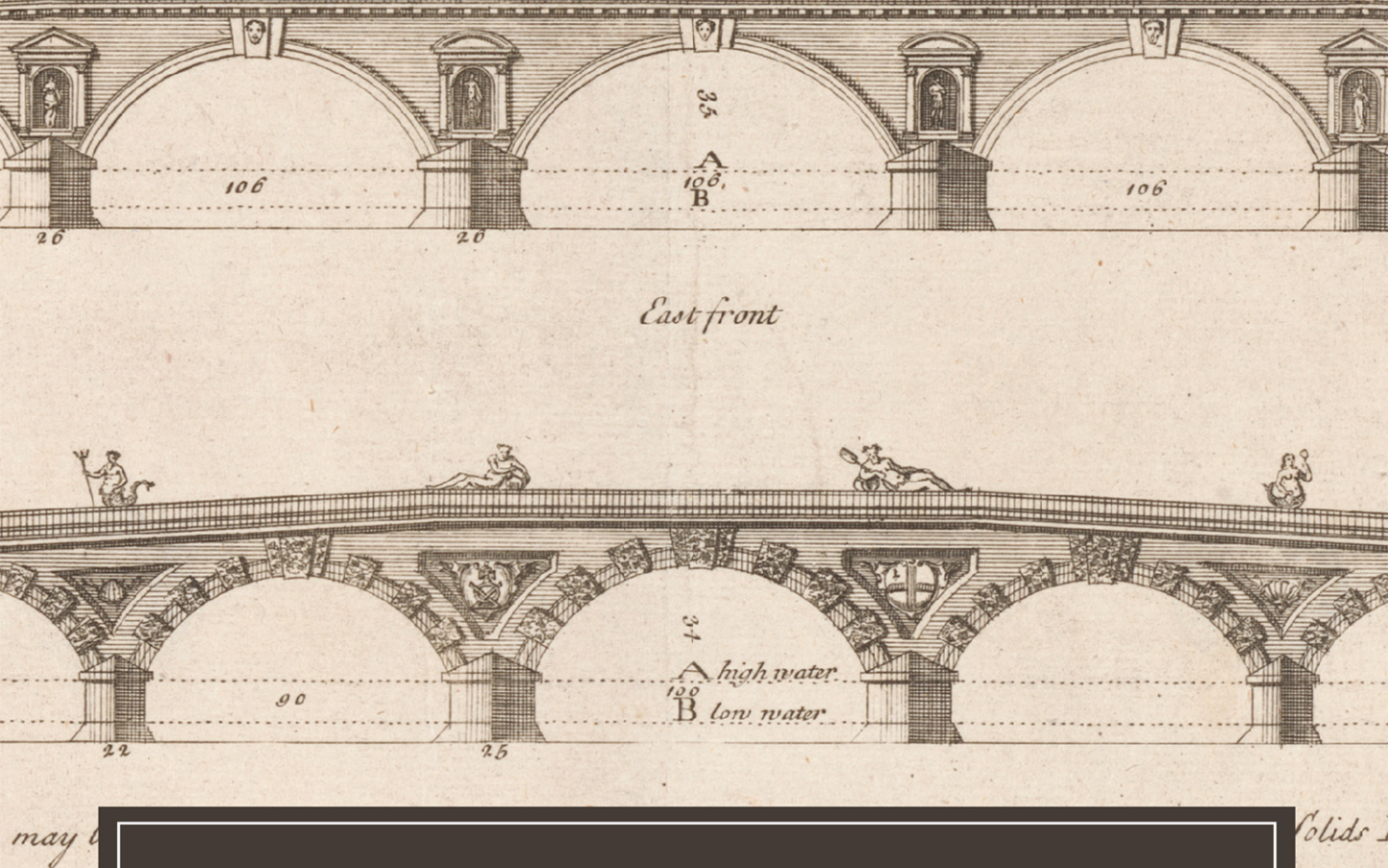


Theory and Applications of Craig Interpolation

Edited by

Balder ten Cate
Jean Christoph Jung
Patrick Koopmann
Christoph Wernhard
Frank Wolter

Theory and Applications of Craig Interpolation

Edited by
Balder ten Cate
Jean Christoph Jung
Patrick Koopmann
Christoph Wernhard
Frank Wolter

Published by
Ubiquity Press Ltd.
Unit 3N, 6 Osborn Street
London, E1 6TD
United Kingdom
https://www.ubiquitypress.com/

Text © the authors, 2026

First published 2026

Cover design by Tom Grady
Cover image: Designs for Blackfriars Bridge, 1756, Benjamin Cole, 1697-1783. Public Domain, Yale Center for British Art, Paul Mellon Collection

Typeset by the authors.
Final preparation of print and digital versions by
Lumina Datamatics Limited

ISBN (Hardback): 978-1-914481-82-6
ISBN (PDF): 978-1-914481-83-3
ISBN (EPUB): 978-1-914481-84-0
ISBN (Mobi): 978-1-914481-85-7

DOI: https://doi.org/10.5334/bdg

This work is licensed under the Creative Commons Attribution 4.0 International License (unless stated otherwise within the content of the work). To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA. This license allows for copying any part of the work for personal and commercial use, providing author attribution is clearly stated. The full text of this book has been peer-reviewed to ensure high academic standards. For full review policies, see https://www.ubiquitypress.com/

Suggested citation: Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard and Frank Wolter (Eds.) 2026. *Theory and Applications of Craig Interpolation.* London: Ubiquity Press. DOI: https://doi.org/10.5334/bdg. License: CC BY 4.0

To read the free, open access version of this book online, visit https://doi.org/10.5334/bdg or scan this QR code with your mobile device:

Editors

Balder ten Cate
University of Amsterdam

Jean Christoph Jung
TU Dortmund University

Patrick Koopmann
Vrije Universiteit Amsterdam

Christoph Wernhard
University of Potsdam

Frank Wolter
University of Liverpool

Dedicated to the memory of
Larisa Lvovna Maksimova

Contents

Acknowledgments ix

Preface xi

Reviewers xiii

Foundations

1 Interpolation in Classical Propositional Logic
Patrick Koopmann, Christoph Wernhard, and Frank Wolter 1–33

2 Interpolation in First-Order Logic
Balder ten Cate and Jesse Comer 35–67

3 Interpolation in Non-Classical Logics
Wesley Fussner 69–100

4 Six Proofs of Interpolation for the Modal Logic K
Nick Bezhanishvili, Balder ten Cate, and Rosalie Iemhoff 101–147

Methods

5 Interpolation in Proof Theory
Iris van der Giessen, Raheleh Jalali, and Roman Kuznets 149–196

6 Proof Complexity and Feasible Interpolation
Amirhossein Akbar Tabatabai 197–243

7 Interpolation in Model Theory
Jouko Väänänen 245–272

8 Interpolation and Amalgamation
George Metcalfe 273–299

9 Uniform Interpolation
Sam van Gool 301–328

10 From Interpolating Formulas to Separating Languages and Back Again
Agi Kurucz, Frank Wolter, and Michael Zakharyaschev 329–369

11 Interpolation via Cyclic Proofs
Bahareh Afshari and Graham E. Leigh 371–413

12 Interpolation with Automated First-Order Reasoning
Christoph Wernhard 415–459

Applications

13 Craig Interpolation in Program Verification
Philipp Rümmer 461–501

14 Interpolation and Query Rewriting
Michael Benedikt 503–537

15 Interpolation in Knowledge Representation
Jean Christoph Jung, Patrick Koopmann, and Matthias Knorr 539–588

16 Definability and Interpolation in Philosophy
Johan van Benthem 589–614

Acknowledgments

For helpful feedback we wish to thank Aurélien Blancal-Simon, Aurélien Boeri, Axel De Masure, Diane Ducrocq, François Gallois, Paul Huber, Corentin Jazeron, and Antonin Loubière. The project would not have been possible without support from various parties, which we gratefully acknowledge: The Evert Willem Beth Foundation; TU Dortmund University; Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495.

Preface

The Craig Interpolation Property, "CIP", has long stood as a central theme in logic, bridging proof theory, model theory, and applications in computer science and philosophy. Since William Craig's seminal theorem in 1957, interpolation has provided a powerful lens through which to examine the expressive and deductive boundaries of logical systems. The property's deceptively simple statement—that whenever a formula of the form $\varphi \to \psi$ is valid, there exists an interpolant using only the common vocabulary of φ and ψ—has inspired decades of research. Beyond its theoretical significance, interpolation has become a cornerstone in applications. Computed with techniques of automated reasoning, interpolants serve as separators in program verification based on model checking and abstraction refinement. In databases and knowledge representation, they provide tools for ontology modularization, concept learning, and query rewriting.

This volume brings together contributions that chart the evolving landscape of Craig Interpolation. The chapters explore both classical results and recent advances. They also highlight how interpolation continues to inform both foundational theory and practical algorithms, underscoring its enduring relevance. The editors hope that this collection will serve as both a reference and an inspiration. For the seasoned researcher, it offers a synthesis of recent progress and open challenges. For the student, it provides a gateway into a rich area where proof theory, complexity, and applications converge. Above all, the volume testifies to the vitality of Craig Interpolation as a subject that continues to connect disciplines, stimulate new methods, and pose deep questions about the nature of logical inference.

The idea of editing a volume on Craig Interpolation—and the closely related Beth Definability—first took shape in September 2023 during a bus ride at the *Fourteenth International Tbilisi Symposium on Logic, Language and Computation (TbiLLC 2023)* in Georgia. Initial chapter proposals were presented at the *Workshop on Theory and Applications of Craig Interpolation and Beth Definability (CIBD 2024)* held in Amsterdam in April 2024, followed by a second workshop in Dortmund in November 2024, where the development of chapters was discussed in greater detail. The year 2025 was then devoted to writing and reviewing of the chapters that comprise this volume.

We dedicate this volume to the memory of Larisa Lvovna Maksimova (1943–2025), whose groundbreaking and beautiful contributions to Craig Interpolation in modal and superintuitionistic logics elevated research in the area to a new level and continue to exert a lasting influence on the field.

April 2026

Balder ten Cate
Jean Christoph Jung
Patrick Koopmann
Christoph Wernhard
Frank Wolter

Reviewers

Michael Benedikt
Johan van Benthem
Nick Bezhanishvili
Balder ten Cate
Jesse Comer
Abhishek De
Wesley Fussner
Han Gao
Iris van der Giessen
Sam van Gool
Rosalie Iemhoff
Jean Christoph Jung
Jędrzej Kołodziejski
Patrick Koopmann
Roman Kuznets
George Metcalfe
Leonardo Pacheco
Philipp Rümmer
Amirhossein Akbar Tabatabai
Jouko Väänänen
Christoph Wernhard
Frank Wolter
Michael Zakharyaschev

Chapter 1

Interpolation in Classical Propositional Logic

Patrick Koopmann
Vrije Universiteit Amsterdam, The Netherlands

Christoph Wernhard
University of Potsdam, Germany

Frank Wolter
University of Liverpool, UK

Abstract

We introduce Craig interpolation and related notions such as uniform interpolation, Beth definability, and theory decomposition in classical propositional logic. We present four approaches to computing interpolants: via quantifier elimination, from formulas in disjunctive normal form, and by extraction from resolution or tableau refutations. We close with a discussion of the size of interpolants and links to circuit complexity.

Contents

1 Introduction **2**
2 Preliminaries **3**
3 Craig Interpolants **4**
4 Uniform Interpolants **6**
5 Beth Definability **8**
6 Generalisations and an Application to Theory Decomposition **9**
7 Interpolants via Quantifier Elimination **11**
8 Interpolants via DNF **12**
9 Interpolants via Resolution **14**
9.1 Uniform Interpolation Using Resolution 15
9.2 Craig Interpolation Using Resolution 17
9.3 Bibliographic and Historic Remarks 21



© Patrick Koopmann, Christoph Wernhard, and Frank Wolter;
licensed under Creative Commons License CC BY 4.0

In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 1, pages 1–33. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.a

10 Craig Interpolants via Analytic Tableaux 21
10.1 A Tableau Proof System 22
10.2 Craig-Lyndon Separator Extraction from a Closed Tableau 23
10.3 Bibliographic and Historic Remarks 27
11 The Size of Interpolants 27
Acknowledgments 30
References 30

1 Introduction

In this chapter, we introduce Craig interpolation in the context of classical propositional logic. We aim to cover the fundamental results and techniques in the field, while still being accessible to readers who have only very basic logic background. We focus on explaining the underpinning logical notions and algorithms. For a discussion of applications of Craig interpolation in areas such as verification, databases, knowledge representation, and philosophy we refer the reader to the respective chapters of this volume.

In detail, we start by introducing the notion of a Craig interpolant between formulas in an implication and the Craig interpolation property, which states that a Craig interpolant exists if the implication is valid. We then introduce uniform interpolants, a strong form of Craig interpolants which do not depend on the right hand side of the implication and which can be regarded as a formalization of forgetting some propositional atoms in a formula. We then link the computation of Craig interpolants to the computation of explicit definitions and the Craig interpolation property to the (projective) Beth definability property. While conceptually, and in applications, definability and interpolation are equally important, because of their mutual reducibility it suffices to discuss one of the two notions in detail. We chose interpolation and next discuss generalizations of Craig interpolation to structured inputs containing more than two formulas and an application to the uniqueness of theory decompositions. Then we present, in the core of this chapter, four ways of computing Craig (and sometimes uniform) interpolants. We start by presenting a simple, but conceptually important, construction that introduces quantified propositional formulas, in which interpolants trivially exist, and then applies quantifier eliminations to obtain propositional logic interpolants. Next we show how interpolants are obtained from formulas in disjunctive normal form by dropping atoms. Our focus is, however, on the introduction of two slightly more sophisticated and also practical methods of computing interpolants, namely their construction from resolution and tableau refutations. In both cases we give rather detailed expositions, assuming no prior understanding of resolution or tableau techniques. We close by discussing the size of interpolants and linking it to open questions in circuit complexity, again assuming hardly any prior knowledge of complexity theory.

2 Preliminaries

Propositional formulas are defined as usual by the grammar

$$\varphi, \psi \quad := \quad p \quad | \quad \top \quad | \quad \bot \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \psi \quad | \quad \varphi \vee \psi,$$

where p ranges over a countably infinite set of *propositional variables* or *atoms*. We set $\varphi \to \psi := \neg\varphi \vee \psi$ and $\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi)$.

A *model* v is a mapping from the set of atoms to $\{0, 1\}$. The *truth value* $v(\varphi) \in \{0, 1\}$ of a formula φ under v is defined as usual by induction using truth tables. We set $v \models \varphi$ and say that φ *is satisfied in* v if $v(\varphi) = 1$. A formula φ is *valid*, in symbols $\models \varphi$, if $v \models \varphi$ for all models v, and φ is *satisfiable* if $v \models \varphi$ for some model v. We say that φ *entails* ψ, in symbols $\varphi \models \psi$, if $v \models \psi$ follows from $v \models \varphi$, for all models v. Observe that $\varphi \models \psi$ iff $\models \varphi \to \psi$. Formulas φ and ψ are *logically equivalent*, in symbols $\varphi \equiv \psi$, if $\varphi \models \psi$ and $\psi \models \varphi$.

The set $sub(\varphi)$ of *subformulas* of a formula φ is defined by induction by setting $sub(p) = \{p\}$, $sub(\top) = \{\top\}$, $sub(\bot) = \{\bot\}$, $sub(\varphi \wedge \psi) = \{\varphi \wedge \psi\} \cup sub(\varphi) \cup sub(\psi)$, $sub(\varphi \vee \psi) = \{\varphi \vee \psi\} \cup sub(\varphi) \cup sub(\psi)$, and $sub(\neg\varphi) = \{\neg\varphi\} \cup sub(\varphi)$. Define the size $|\varphi|$ of a propositional formula φ as the number of its distinct subformulas. This corresponds to representing formulas as directed acyclic graphs (dags) and is the standard measure used in circuit complexity theory. We also call $|\varphi|$ the *dag-size* of φ.

In contrast, if one takes into account the number of different occurrences of subformulas, one obtains the *tree-size* $s(\varphi)$ of φ, defined inductively by setting $s(p) = s(\top) = s(\bot) = 1$, $s(\varphi \wedge \psi) = s(\varphi) + s(\psi) + 1$, $s(\varphi \vee \psi) = s(\varphi) + s(\psi) + 1$, and $s(\neg\varphi) = s(\varphi) + 1$. Clearly, $s(\varphi) \geq |\varphi|$ and it is conjectured that there is a superpolynomial (even exponential) gap between dag-size and tree-size in the sense that there are no functions e mapping every propositional formula φ to a logically equivalent propositional formula $e(\varphi)$ and polynomial function f (that is, a function of the form $f(n) = cn^d$ with c, d natural numbers) such that $s(e(\varphi)) \leq f(|\varphi|)$ for all φ. However, this conjecture remains open [39].

Central for interpolation is the notion of a signature: A *signature* σ is a finite set of atoms. The signature $sig(\varphi)$ of a formula φ is the set of atoms that occur in it. In other words, the signature of φ are the non-logical symbols that occur in φ, whereas logical symbols such as $\wedge$, $\vee$, $\neg$, $\bot$ and $\top$ are not part of the signature of a formula. If $sig(\varphi) \subseteq \sigma$, we call φ a *σ-formula*. Given two formulas φ and ψ, we call ψ a *conservative extension of* φ if $\psi \models \varphi$, $sig(\varphi) \subseteq sig(\psi)$, and for every $sig(\varphi)$-formula χ, $\psi \models \chi$ implies $\varphi \models \chi$.

Our methods for computing interpolants rely on different normal forms which we introduce next. A *literal* ℓ is either an atom or the negation $\neg p$ of an atom p. The *complement* of a literal ℓ, written as $\overline{\ell}$, is $\neg p$ if ℓ is p, and p if ℓ is $\neg p$. A disjunction of literals $C = \ell_1 \vee \cdots \vee \ell_n$ is called a *clause*, often written as $C = \{\ell_1, \ldots, \ell_n\}$ and treated as set. As usual, the empty disjunction is identified with $\bot$. A conjunction of literals $C = \ell_1 \wedge \cdots \wedge \ell_n$ is called a *conjunctive clause*. The empty conjunction is identified with $\top$.

A formula φ is in *conjunctive normal form (CNF)* if it is a conjunction of clauses; it is in *disjunctive normal form (DNF)* if it is a disjunction of conjunctive clauses. Every formula

is logically equivalent to a formula in CNF and DNF. For convenience, we may sometimes identify formulas in CNF/DNF with the corresponding sets of (conjunctive) clauses.

A formula φ is in *negation normal form (NNF)* if it is built from literals, truth value constants, conjunction and disjunction. Formally, NNF formulas are defined by the grammar

$$\varphi, \psi \quad := \quad p \quad | \quad \neg p \quad | \quad \top \quad | \quad \bot \quad | \quad \varphi \wedge \psi \quad | \quad \varphi \vee \psi,$$

where p ranges over the set of propositional atoms. CNF and DNF formulas are both special cases of NNF formulas. While any propositional formula can be converted in linear time to an equivalent NNF formula with an at most linear size increase, the smallest CNF and DNF formulas that are equivalent to a given formula may be exponentially larger.[1] One can, however, construct in polynomial time for every propositional formula φ a formula φ' in CNF that is a conservative extension of φ, using structural transformations [3] such as the Tseitin encoding [46].

3 Craig Interpolants

In this section we give the main definitions of this chapter.

▶ **Definition 1** (Craig Interpolant). *Let φ, ψ be formulas. Then a formula χ is called a* Craig interpolant *for φ, ψ if*

- $\varphi \models \chi$ *and* $\chi \models \psi$*; and*
- $sig(\chi) \subseteq sig(\varphi) \cap sig(\psi)$.

Clearly, if a Craig interpolant for φ, ψ exists, then $\varphi \models \psi$, by transitivity of the entailment relation $\models$. The converse direction, stating that a Craig interpolant for φ, ψ exists whenever $\varphi \models \psi$, is called the *Craig interpolation property (CIP)*. It holds for propositional logic.

▶ **Theorem 2** (Craig Interpolation Property (CIP)). *In propositional logic, if $\varphi \models \psi$, then there exists a Craig interpolant for φ, ψ.*

We will present and discuss several proofs of this result in Sections 7–10. Here we continue with illustrating the definition by examples, basic observations, and variants of the definition.

▶ **Example 3.**

(i) Observe that $p \wedge q_1 \models q_2 \to p$. By CIP a Craig interpolant for $p \wedge q_1$, $q_2 \to p$ exists. Clearly p is such a Craig interpolant. Moreover, up to logical equivalence, p is the only Craig interpolant for $p \wedge q_1$, $q_2 \to p$. Indeed, up to logical equivalence, the only other candidates are $\bot$, $\top$, and $\neg p$, and none of these is a Craig interpolant for $p \wedge q_1$, $q_2 \to p$.

[1] Each formula in DNF equivalent to $\bigwedge_{1 \leq i \leq n}(p_i \vee q_i)$ has at least 2^n conjunctive clauses. Each formula in CNF equivalent to $\bigvee_{1 \leq i \leq n}(p_i \wedge q_i)$ has at least 2^n clauses.

(ii) We have $p \wedge \neg p \models q$ and $\bot$ is a Craig interpolant for $p \wedge \neg p$, q. Up to logical equivalence, $\bot$ is the only Craig interpolant. This example shows that without admitting at least one of $\top$ or $\bot \equiv \neg\top$ as a formula not using any atoms, Craig interpolation would fail for formulas in disjoint signatures.

(iii) Consider atoms $p_1, \dots, p_n$. Then $\bigwedge_{1 \leq i \leq n}(p_i \wedge \neg p_i) \models \bigvee_{1 \leq i \leq n}(p_i \vee \neg p_i)$ and any formula using only atoms from $p_1, \dots, p_n$ is a Craig interpolant.

(iv) Let $\varphi = p \wedge q \wedge r$ and $\psi = s \to (p \vee q)$. Then $\varphi \models \psi$ and the Craig interpolants for φ, ψ are, up to logical equivalence, the formulas $p \wedge q$, p, q, and $p \vee q$.

(v) If φ is a propositional formula that is semantically independent from some atoms $\sigma \subseteq \mathit{sig}(\varphi)$, i.e., φ is equivalent to a propositional formula ψ without atoms in σ, then Craig interpolation can be applied to find such a ψ. Specifically, any Craig interpolant for φ, φ', where φ' is the formula obtained from φ by systematically replacing all atoms in σ with fresh atoms, provides such a formula ψ. For example, let

$$\varphi = (p \to (q \wedge r)) \wedge (p \vee q) \wedge (q \to r).$$

Let $\sigma = \{p\}$ and $\varphi' = (p' \to (q \wedge r)) \wedge (p' \vee q) \wedge (q \to r)$. We obtain $\psi = q \wedge r$, which is equivalent to φ, as Craig interpolant for φ, φ'. ⌟

Example 3.iii shows that the space of Craig interpolants for a fixed pair of formulas can be large. It has, however, a very transparent structure:

▶ **Theorem 4** (Closure of Craig Interpolants under Conjunction and Disjunction). *Let φ and ψ be formulas and $C(\varphi, \psi)$ be the set of Craig interpolants for φ, ψ. Then $C(\varphi, \psi)$ is closed under conjunctions and disjunctions: if $\chi_1, \chi_2 \in C(\varphi, \psi)$, then $\chi_1 \wedge \chi_2 \in C(\varphi, \psi)$ and $\chi_1 \vee \chi_2 \in C(\varphi, \psi)$.*

As $C(\varphi, \psi)$ contains only finitely many formulas up to logical equivalence, it contains a logically strongest formula (that entails all others), given by the conjunction of all formulas in $C(\varphi, \psi)$, and a logically weakest formula (that is entailed by all others), given by the disjunction of all formulas in $C(\varphi, \psi)$.

In many applications and proofs, a different, but equivalent, view of Craig interpolants and CIP is useful.

▶ **Definition 5** (Craig Separator). *Let φ and ψ be formulas. Then a formula χ is a* Craig separator *for φ, ψ if*

- *$\varphi \models \chi$ and $\chi \wedge \psi \models \bot$; and*
- *$\mathit{sig}(\chi) \subseteq \mathit{sig}(\varphi) \cap \mathit{sig}(\psi)$.*

Clearly, χ is a Craig separator for φ, ψ iff χ is a Craig interpolant for $\varphi, \neg\psi$. The CIP is then equivalent to the statement that for any formulas φ, ψ, if $\varphi \wedge \psi$ is not satisfiable, then there exists a Craig separator for φ and ψ.

In the literature on verification, a Craig separator is often termed *reverse Craig interpolant* or even just *Craig interpolant*.

Craig interpolation can be strengthened to take also the *polarity* of atom occurrences into account, which is often called *Craig-Lyndon* interpolation. Let $sig^{\pm}(\varphi)$ denote the set of all pairs $\langle p, pol\rangle$ such that atom p occurs in formula φ with polarity $pol \in \{+, -\}$, where the polarity of an occurrence is $+$ (respectively, $-$) if it is in the scope of an even (respectively, odd) number of negation symbols. We can then define the notion of *Craig-Lyndon interpolant* as follows.

▶ **Definition 6** (Craig-Lyndon Interpolant). *Let φ, ψ be formulas. Then a formula χ is called a* Craig-Lyndon interpolant *for φ, ψ if*

- $\varphi \models \chi$ *and* $\chi \models \psi$*; and*
- $sig^{\pm}(\chi) \subseteq sig^{\pm}(\varphi) \cap sig^{\pm}(\psi)$.

▶ **Example 7.** Let $\varphi = (p \to q) \wedge (r \to (p \vee q)) \wedge t$ and let $\psi = (p \to (q \wedge t)) \wedge ((q \wedge s) \to t)$. Atom p occurs in φ with both polarities but in ψ only negatively. Atom q occurs in ψ with both polarities but in φ only positively. Atom t occurs just positively, in both φ and ψ. Hence, in a Craig-Lyndon interpolant for φ, ψ the atom p is allowed only negatively, and q, t are allowed only positively. Atoms r, s are not allowed at all, since each of them occurs just in one of φ or ψ. Indeed, $\chi = (p \to q) \wedge t$ is a Craig-Lyndon interpolant for φ, ψ. The formula $\chi' = (p \to (q \wedge t)) \wedge (q \to t)$ is another Craig interpolant for φ, ψ, but it is no Craig-Lyndon interpolant as it has a negative occurrence of q. ⌟

A Craig-Lyndon interpolant is sometimes also called a *Lyndon interpolant* or a Craig interpolant with the *Lyndon property*. In propositional logic, the following strengthening of Theorem 2 holds.

▶ **Theorem 8** (Craig-Lyndon Interpolation Property (LIP)). *In propositional logic, if $\varphi \models \psi$, then there exists a Craig-Lyndon interpolant for φ, ψ.*

Since a Craig-Lyndon interpolant is also a Craig interpolant, the LIP (Theorem 8) implies the CIP (Theorem 2). In fact, some of the proofs in Sections 7–10 actually show the LIP. The notion of *Craig separator* also has an analog where polarity is considered.

▶ **Definition 9** (Craig-Lyndon Separator). *Let φ and ψ be formulas. Then a formula χ is a* Craig-Lyndon separator *for φ, ψ if*

- $\varphi \models \chi$ *and* $\chi \wedge \psi \models \bot$*; and*
- $sig^{\pm}(\chi) \subseteq sig^{\pm}(\varphi) \cap sig^{\pm}(\neg\psi)$.

As before, a formula χ is a Craig-Lyndon separator for φ, ψ iff χ is a Craig-Lyndon interpolant for $\varphi, \neg\psi$.

4 Uniform Interpolants

In this section we introduce a special form of Craig interpolant called a *uniform* interpolant. While Craig interpolants are defined in terms of a pair of input formulas φ, ψ with $\varphi \models \psi$, uniform interpolants are defined in terms of a single formula φ and a signature $\sigma \subseteq sig(\varphi)$.

▶ **Definition 10** (Uniform Interpolant). *Let φ be a propositional formula and $\sigma \subseteq sig(\varphi)$. Then a propositional formula χ is called a* uniform σ-interpolant for φ *if*

- $\varphi \models \chi$;
- $sig(\chi) \subseteq \sigma$; *and*
- *for every propositional formula ψ, if $\varphi \models \psi$ and $sig(\psi) \cap sig(\varphi) \subseteq \sigma$, then $\chi \models \psi$.*[2]

Observe that, given φ and $\sigma \subseteq sig(\varphi)$, a uniform σ-interpolant for φ is a Craig interpolant for all pairs φ, ψ such that $\varphi \models \psi$ and $\sigma = sig(\psi) \cap sig(\varphi)$. In contrast to Craig interpolants, uniform interpolants are uniquely determined (up to logical equivalence) by φ and σ. In what follows, we therefore refer to *the* uniform σ-interpolant for a formula φ.

▶ **Example 11.** We consider the uniform σ-interpolants for the formulas φ in the entailments $\varphi \models \psi$ in Example 3, where $\sigma = sig(\varphi) \cap sig(\psi)$. The uniform $\{p\}$-interpolant for $p \wedge q_1$ is given by p and the uniform $\emptyset$-interpolant for $p \wedge \neg p$ is given by $\bot$. Similarly, the uniform $\{p_1, \ldots, p_n\}$-interpolant for $\bigwedge_{1 \leq i \leq n}(p_i \wedge \neg p_i)$ is given by $\bot$. Finally, the uniform $\{p, q\}$-interpolant for $p \wedge q \wedge r$ is $p \wedge q$. ⌟

Note that in Example 11, the uniform σ-interpolant is always the logically strongest Craig interpolant for φ, ψ. This is no accident. To show this in general, assume that χ is the logically strongest Craig interpolant for φ, ψ. Let $\sigma = sig(\varphi) \cap sig(\psi)$ and let χ' be the uniform σ-interpolant for φ. We show that χ and χ' are logically equivalent. Clearly $\chi' \models \chi$ since $\varphi \models \chi$ and $sig(\chi) \subseteq \sigma$. We also have $\chi \models \chi'$ since χ' is a Craig interpolant for φ, ψ and χ is the logically strongest one.

Sometimes, a uniform interpolant as in Definition 10 is also called *right-uniform interpolant*, indicating that it is a Craig interpolant for an implication $\varphi \models \psi$ that is independent of the formula *on the right*. This notion can then be contrasted with that of a *left-uniform interpolant*, which is a Craig interpolant that is independent of the formula on the left (it is a Craig interpolant also for any other entailment $\varphi' \models \psi$):

▶ **Definition 12** (Left-Uniform Interpolant). *Let φ be a propositional formula and $\sigma \subseteq sig(\varphi)$. Then a propositional formula χ is called a* left-uniform σ-interpolant for φ *if*

- $\chi \models \varphi$;
- $sig(\chi) \subseteq \sigma$; *and*
- *for every propositional formula ψ, if $\psi \models \varphi$ and $sig(\psi) \cap sig(\varphi) \subseteq \sigma$, then $\psi \models \chi$.*

For logics with classical negation, right-uniform interpolation can easily be reduced to left-uniform interpolation and vice versa: χ is a right-uniform σ-interpolant for ψ iff $\neg\chi$ is a left-uniform σ-interpolant for $\neg\psi$. For the rest of this chapter, we therefore focus on right-uniform interpolants and call them simply uniform interpolants.

---

[2] Uniform interpolants are not always defined in exactly the same way. For instance, instead of starting from the signature σ of the uniform interpolant, one can start from the signature $sig(\varphi) \setminus \sigma$ that is 'forgotten', or the condition $sig(\psi) \cap sig(\varphi) \subseteq \sigma$ is replaced by the condition $sig(\psi) \subseteq \sigma$. In the context of propositional logic these different definitions are equivalent.

The following result is shown in different ways in Sections 7–9.

▶ **Theorem 13** (Uniform Interpolation Property (UIP)). *In propositional logic, if φ is a propositional formula and $\sigma \subseteq sig(\varphi)$, then a uniform σ-interpolant for φ exists.*

Note that any logic with the uniform interpolation property automatically satisfies the CIP.

5 Beth Definability

In this section, we show that Craig interpolants are closely related to definitions and can be used to compute an explicit definition of an atom that is implicitly defined. We first formalize the notions of explicit and implicit definitions.

▶ **Definition 14** (Explicit Definability). *Let φ be a propositional formula, σ be a set of atoms, and $p \notin \sigma$. Then a propositional formula ψ is a* σ-definition of p under φ *if $sig(\psi) \subseteq \sigma$ and $\varphi \models p \leftrightarrow \psi$. p is called* explicitly σ-definable under φ *if there exists a σ-definition of p under φ.*

The condition $p \notin \sigma$ reflects the intuition that definitions should not be circular. Also note that we are not stipulating a definition of p but ask whether p is logically equivalent to some ψ under the assumption φ.

Alternatively, we may approach definability from a semantic point of view: The atom p is *implicitly σ-definable under φ* if, whenever two models that satisfy φ agree on the values of all atoms in σ, then they also agree on the value of p. Intuitively, the interpretation of the atoms in σ fully determines the interpretation of p. The formal definition is as follows.

▶ **Definition 15** (Implicit Definability). *Let φ be a propositional formula, σ be a set of atoms, and $p \notin \sigma$. Then p is* implicitly σ-definable under φ *if for any models v_1, v_2 satisfying φ, $v_1(q) = v_2(q)$ for all $q \in \sigma$ implies $v_1(p) = v_2(p)$.*

Clearly explicit definability implies implicit definability. The converse is called *projective Beth definability property (BDP)* and does not hold for all logics. It does hold, however, for classical propositional logic.

▶ **Theorem 16** (Projective Beth Definability Property (BDP)). *Propositional logic has the projective Beth definability property.*

Proof. The proof is by reduction to the Craig interpolation property. Assume p is implicitly σ-definable under φ. Let φ' be obtained from φ by systematically replacing all atoms q not in σ by associated fresh atoms q'. Then implicit definability implies

$$\varphi \wedge \varphi' \models p \leftrightarrow p'$$

Hence

$$\varphi \wedge p \models \varphi' \rightarrow p'$$

Let χ be any Craig interpolant for $\varphi \wedge p, \varphi' \to p'$. We show that χ is a σ-definition of p under φ. By definition of Craig interpolants, $sig(\chi) \subseteq \sigma$. From $\varphi \wedge p \models \chi$ we obtain $\varphi \models p \to \chi$. Conversely, from $\chi \models \varphi' \to p'$ we obtain $\varphi' \models \chi \to p'$. Then, since $sig(\chi) \subseteq \sigma$, we obtain $\varphi \models \chi \to p$, as required. ◀

Observe that the proof of Theorem 16 shows how one can construct in polynomial time explicit definitions from Craig interpolants: given φ, a signature σ, and $p \notin \sigma$, a σ-definition of p under φ is obtained by taking any Craig interpolant for propositional formulas χ, χ' computed from φ, σ, and p in linear time.

The special case of the projective Beth definability property where σ is not a parameter but defined as $\sigma = sig(\varphi) \setminus \{p\}$ is known as the *Beth definability property*. Note also that the notions of explicit and implicit definability, and the projective Beth definability property can be easily generalized to definitions of compound propositional formulas in place of atoms p.

▶ **Example 17.**

(i) As a very simple first example for Beth definability consider $\varphi = p \leftrightarrow q$ and $\sigma = \{q\}$. Clearly p is σ-definable under φ. The σ-definition q is a Craig interpolant for $(\varphi \wedge p)$,$(\varphi(p/p') \to p')$. The entailment underlying the Craig interpolation is $(p \leftrightarrow q) \wedge p \models (p' \leftrightarrow q) \to p'$, which can be equivalently expressed as $p \wedge q \models p' \vee q$.

(ii) Let $\varphi = (p \to (q \wedge r)) \wedge ((q \wedge s) \to p) \wedge (r \to s)$ and $\sigma = \{q, r\}$. Since $\varphi \models p \leftrightarrow (q \wedge r)$ atom p is σ-definable under φ. The σ-definition $q \wedge r$ is a Craig interpolant for $(\varphi \wedge p)$,$(\varphi(p/p', s/s') \to p')$.

(iii) Let $\varphi = (p \to (q \wedge r)) \wedge ((q \vee r) \to p)$ and $\sigma = \{q, r\}$. Then p is σ-definable under φ, actually with several σ-definitions: $\varphi \models p \leftrightarrow (q \wedge r)$, $\varphi \models p \leftrightarrow q$, $\varphi \models p \leftrightarrow r$, and $\varphi \models p \leftrightarrow (q \vee r)$. Each of these four definitions is a Craig interpolant for $(\varphi \wedge p)$,$(\varphi(p/p') \to p')$.

(iv) Let $\varphi = p$ and $\sigma = \{\}$. Then p is σ-definable under φ, since $p \models p \leftrightarrow \top$. The σ-definition $\top$ is a Craig interpolant of $(\varphi \wedge p)$,$(\varphi(p/p') \to p')$, i.e., of $(p \wedge p)$,$(p' \to p')$. ⌟

We also give a polynomial time reduction of the computation of Craig interpolants to the computation of explicit definitions.

▶ **Theorem 18** (Reduction of Craig Interpolation to Explicit Definitions). *Assume* $\models \varphi \to \psi$. *Then the following conditions are equivalent for any formula* χ*:*

- χ *is a Craig interpolant for* φ, ψ*;*
- χ *is an explicit* $sig(\varphi) \cap sig(\psi)$*-definition of* ψ *under* $\psi \to \varphi$*.*

Because of the mutual polynomial time reduction between Craig interpolant computation and explicit definition computation shown in this section, in what follows we focus on Craig interpolation without always mentioning again the consequences for explicit definitions.

6 Generalisations and an Application to Theory Decomposition

Craig interpolation has been generalised in many different ways. In particular, in applications of Craig interpolation to program verification various inductive generalisations have been

introduced, including inductive sequences of interpolants [18, 33], tree interpolants [17, 7], and disjunctive interpolation [40, 41], in increasing order of generality. We refer the reader to Chapter 12 for further discussion. In all these generalisations, one considers multiple formulas (for instance, as labels of nodes in a tree), and the interpolants capture the interaction between these formulas in their shared signature.

Here we focus on a generalisation observed by Craig already [11, Lemma 2].

▶ **Theorem 19.** *In propositional logic, if* $\varphi_1 \wedge \ldots \wedge \varphi_k \models \varphi_{k+1} \vee \ldots \vee \varphi_l$*, then there are formulas* $\chi_1, \ldots, \chi_l$ *with*

- $\varphi_i \models \chi_i$ *for* $1 \leq i \leq k$
- $\chi_i \models \varphi_i$ *for* $k+1 \leq i \leq l$
- $sig(\chi_i) \subseteq sig(\varphi_i) \cap \bigcup_{j \in \{1,\ldots,l\}, j \neq i} sig(\varphi_j)$ *for* $1 \leq i \leq l$
- $\chi_1 \wedge \ldots \wedge \chi_k \models \chi_{k+1} \vee \ldots \vee \chi_l$

In applications to program verification, these interpolants have been studied for $l = k+1$ and $\varphi_{k+1} = \bot$ [31, Sect. 5], where they are called *symmetric interpolants*. We focus on a brief discussion of another special case of Theorem 19 called *parallel interpolation*. In what follows, T denotes a finite set of propositional formulas which, intuitively, axiomatises some theory or knowledge base of interest. We often identify T with the conjunction $\bigwedge_{\chi \in T} \chi$; for example, $T \models \varphi$ stands for $\bigwedge_{\chi \in T} \chi \models \varphi$. Now, parallel interpolation is the special case in which $l = k+1$ and $sig(\varphi_i) \cap sig(\varphi_j) = \emptyset$ for $1 \leq i < j \leq k$. So, given $T = \{\varphi_1, \ldots, \varphi_n\}$ with $sig(\varphi_i) \cap sig(\varphi_j) = \emptyset$ for $1 \leq i < j \leq n$ and a formula ψ with $T \models \psi$, there are formulas $\chi_1, \ldots, \chi_n$, called *parallel interpolants*, with

- $\varphi_i \models \chi_i$ for $1 \leq i \leq n$;
- $sig(\chi_i) \subseteq sig(\varphi_i) \cap sig(\psi)$ for $1 \leq i \leq n$; and
- $\{\chi_1, \ldots, \chi_n\} \models \psi$.

Intuitively, the parallel interpolants χ_i capture, in the shared atoms of φ_i and ψ, the contribution of φ_i to the derivation of ψ from T. Although fairly straightforward to derive from Craig interpolation, parallel interpolation has some unexpected applications. Here we use it to show that finest decompositions of theories into axioms with mutually disjoint signatures are uniquely determined (up to logical equivalence). In detail, let T be a set of formulas and let $\sigma_1, \ldots, \sigma_n$ be a partition of $sig(T)$. Then $\sigma_1, \ldots, \sigma_n$ is called a *splitting* of T if there are formulas $\chi_1, \ldots, \chi_n$ such that $sig(\chi_i) \subseteq \sigma_i$ for $1 \leq i \leq n$ and $\{\chi_1, \ldots, \chi_n\}$ axiomatises T, that is to say, $\bigwedge_{1 \leq i \leq n} \chi_i$ and $\bigwedge_{\chi \in T} \chi$ are logically equivalent. Splittings formalise the idea of decomposing a theory T into axioms that speak about mutually disjoint subject matters. We say that a partition $\sigma_1, \ldots, \sigma_n$ of $sig(T)$ is at least as fine as partition $\sigma'_1, \ldots, \sigma'_m \subseteq sig(T)$ if for all σ'_i there exists σ_j with $\sigma_j \subseteq \sigma'_i$.

▶ **Theorem 20.** *Every finite set* T *of propositional formulas has a unique finest splitting.*

Proof. Assume splittings $\sigma_1, \ldots, \sigma_n$ and $\sigma'_1, \ldots, \sigma'_m$ of T are given. We show that the set of non-empty $\sigma_i \cap \sigma'_j$, with $1 \leq i \leq n$ and $1 \leq j \leq m$ is again a splitting of T. The claim then follows directly by assuming that $\sigma_1, \ldots, \sigma_n$ and $\sigma'_1, \ldots, \sigma'_m$ are finest splittings of T. Let

$\varphi_1, \ldots, \varphi_n$ and $\psi_1, \ldots, \psi_m$ be axiomatisations of T witnessing that $\sigma_1, \ldots, \sigma_n$ and $\sigma'_1, \ldots, \sigma'_m$ are splittings of T, respectively. Then take the parallel interpolants χ_{ij} for the entailments $\{\varphi_1, \ldots, \varphi_n\} \models \psi_j$ with $\varphi_i \models \chi_{ij}$, for $1 \leq i \leq n$ and $1 \leq j \leq m$. Then $sig(\chi_{ij}) \subseteq \sigma_i \cap \sigma'_j$ and the χ_{ij} axiomatise T. So they witness that the non-empty $\sigma_i \cap \sigma'_j$ form a splitting of T. ◀

For further results on parallel interpolation and its application to theory decomposition in philosophy and knowledge representation we refer the reader to [26, 25, 35].

7 Interpolants via Quantifier Elimination

Having discussed various generalizations and applications of interpolation, in the next few sections, we discuss different proofs of the Craig interpolation property (CIP) and methods for computing interpolants. We start with the easiest proof, which shows the stronger result that propositional logic has the uniform interpolation property (Theorem 13). CIP then follows. While the proof constructs uniform interpolants (and so Craig interpolants), it does not show how to construct interpolants efficiently. We examine efficient methods in Sections 8–10.

The proof has two steps: (i) we show that uniform interpolants always exist in the language obtained from propositional logic by adding quantifiers over atoms; and (ii) we show that these quantifiers can be always eliminated in the sense that one can always construct a logically equivalent propositional formula that does not use quantifiers. Overall, one then obtains that uniform interpolants always exist.

We first extend propositional logic by adding quantifiers. *QBF (quantified Boolean formulas)* are an extension of propositional logic defined according to the following grammar:

$$\varphi, \psi \quad := \quad p \quad | \quad \top \quad | \quad \bot \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \psi \quad | \quad \varphi \vee \psi \quad | \quad \exists p.\varphi$$

where p ranges over a countably infinite set of atoms. Models v of QBF are defined in the same way as for propositional logic and we define $v \models \varphi$ inductively using the additional condition

- $v \models \exists p.\varphi$ if there is a model v' that coincides with v for all atoms except possibly p such that $v' \models \varphi$.

Validity, satisfiability, and logical equivalence are now defined in the obvious way. The definition of the *signature* $sig(\varphi)$ of a QBF φ extends inductively the definition for propositional logic with $sig(\exists p.\varphi) = sig(\varphi) \setminus \{p\}$.

We are now in a position to give the mentioned proof of uniform interpolation in two steps.

QBF Uniform Interpolants. Assume φ and a signature $\sigma \subseteq sig(\varphi)$ are given. Let $p_1, \ldots, p_n$ be any ordering of the atoms in $sig(\varphi) \setminus \sigma$ and consider the QBF $\chi = \exists p_1 \cdots \exists p_n.\varphi$. Then

- $\varphi \models \chi$ and $sig(\chi) = \sigma$;
- for every propositional formula ψ, if $\varphi \models \psi$ and $sig(\psi) \cap sig(\varphi) \subseteq \sigma$, then $\chi \models \psi$.

and so χ satisfies the conditions for a uniform σ-interpolant for φ, except that it is not a propositional formula.

Quantifier Elimination. Let $\varphi = \exists p_1 \cdots \exists p_n.\psi$ be a QBF with ψ a propositional formula. Then φ is logically equivalent to a propositional formula. Indeed, let $S_n = \{\bot, \top\}^n$ be the set of all sequences of $\bot$ and $\top$ of length n. Then

$$\bigvee_{t_1,\ldots,t_n \in S_n} \psi[p_1/t_1, \ldots, p_n/t_n]$$

is logically equivalent to φ, where $\psi[p_1/t_1, \ldots, p_n/t_n]$ is obtained from ψ by replacing p_i by t_i for $1 \leq i \leq n$. Using quantifier elimination, we can thus easily compute propositional uniform interpolants from QBF uniform interpolants.

Note that it can now also be shown by induction that every QBF is logically equivalent to a propositional formula.

▶ **Example 21.** Let $\varphi = \exists q.(p \to q) \wedge (q \to r)$. By quantifier elimination φ is equivalent to $((p \to \bot) \wedge (\bot \to r)) \vee ((p \to \top) \wedge (\top \to r))$, which simplifies to $p \to r$. ⌟

We conclude this section with a few further observations. Let $\forall p.\varphi := \neg\exists p.\neg\varphi$. When constructing Craig interpolants for formulas φ, ψ, instead of $\exists p_1 \cdots \exists p_n.\varphi$ with $p_1, \ldots, p_n$ any ordering of $sig(\varphi) \setminus sig(\psi)$, we could have also used $\forall q_1 \cdots \forall q_n.\psi$ with $q_1, \ldots, q_n$ any ordering of $sig(\psi) \setminus sig(\varphi)$. The formula $\exists p_1 \cdots \exists p_n.\varphi$ is (up to equivalence) the logically strongest interpolant for φ, ψ (it entails all others) and the formula $\forall q_1 \cdots \forall q_n.\psi$ is (up to equivalence) the logically weakest interpolant for φ, ψ (it is entailed by all others).

Quantifier elimination generalizes deciding satisfiability, which can be seen as follows. For $\exists p_1 \ldots \exists p_n.\varphi$ where $\{p_1, \ldots, p_n\} = sig(\varphi)$ quantifier elimination yields a propositional formula built just from logical symbols. Rewriting with equivalences $\neg\bot \equiv \top$, $\neg\top \equiv \bot$, $\top \wedge \psi \equiv \psi$, $\bot \wedge \psi \equiv \bot$, $\top \vee \psi \equiv \top$, and $\bot \vee \psi \equiv \psi$ yields $\top$ if φ is satisfiable and $\bot$ if φ is unsatisfiable.

8 Interpolants via DNF

The uniform σ-interpolant for a formula φ formalises a natural notion of forgetting the atoms in $sig(\varphi) \setminus \sigma$ from φ. This suggests that one can construct the uniform σ-interpolant by somehow dropping the atoms in $sig(\varphi) \setminus \sigma$ from φ in a syntactic sense. This is clearly not the case in general. For instance, for $\varphi = (p \vee q) \wedge \neg p$ and $\sigma = \{p\}$, we do not obtain the uniform σ-interpolant for φ by just dropping q: the result of dropping q is logically equivalent to $p \wedge \neg p$ and $\varphi \not\models p \wedge \neg p$. If, however, φ is in DNF, then we indeed obtain the uniform σ-interpolant by dropping the atoms in $sig(\varphi) \setminus \sigma$.

Assume φ is given in DNF, say

$$\varphi = \bigvee_{1 \leq i \leq n} \varphi_i, \quad \varphi_i = \bigwedge_{1 \leq j \leq n_i} \ell_{ij}$$

with ℓ_{ij} literals. Let $\sigma \subseteq sig(\varphi)$. We may assume that no conjunctive clause φ_i contains an atom and its negation. Then obtain $\varphi^{-\sigma}$ from φ by dropping all literals ℓ_{ij} not in σ from every φ_i.

▶ **Lemma 22.** *Let $p_1, \ldots, p_n$ be an ordering of $sig(\varphi) \setminus \sigma$. Then $\varphi^{-\sigma}$ is logically equivalent to $\exists p_1 \cdots \exists p_n.\varphi$.*

Proof. The lemma can be proven as follows with a syntactic argument. We can move the existential quantifiers in $\exists p_1 \cdots \exists p_n.\varphi$ inward, preserving equivalence, until all quantifications are of the form $\exists p_i.p_i$ or $\exists p_i.\neg p_i$ (we assume that the conjunctive clauses of φ contain no duplicate literals). We can do this due to the following equivalences: 1) $\exists p_1 \ldots \exists p_n.\varphi \vee \psi \equiv \exists p_1 \ldots \exists p_n.\varphi \vee \exists p_1 \ldots \exists p_n.\psi$; 2) if $p \notin sig(\psi)$, then $\exists p.\varphi \wedge \psi \equiv (\exists p.\varphi) \wedge \psi$; and 3) if $p \notin sig(\varphi)$, then $\exists p.\varphi \equiv \varphi$.

Then we apply our quantifier elimination method from Section 7 to these quantified literals. For positive as well as negative literals this yields $\top$, i.e., $\exists p_i.p_i \equiv p_i[p_i/\bot] \vee p_i[p_i/\top] \equiv \top$ and $\exists p_i.\neg p_i \equiv \neg p_i[p_i/\bot] \vee \neg p_i[p_i/\top] \equiv \top$. By removing these redundant subformulas $\top$ from the conjunctive clauses, we obtain exactly $\varphi^{-\sigma}$. ◀

We can generalize this idea of computing uniform interpolants to the computation of Craig interpolants, for which we take into account the logical strength of the right hand side of the entailment. It is more convenient to look at Craig separators here (the construction of Craig interpolants would assume the right-hand side to be in CNF rather than DNF). Assume that

$$\varphi = \bigvee_{1 \le i \le n} \varphi_i, \quad \varphi_i = \bigwedge_{1 \le j \le n_i} \ell_{ij}, \quad \psi = \bigvee_{1 \le i' \le m} \psi_{i'}, \quad \psi_{i'} = \bigwedge_{1 \le j' \le m_{i'}} \ell'_{i'j'}$$

and that $\varphi \wedge \psi$ is not satisfiable. We may assume that no φ_i nor ψ_i contains an atom and its negation. We construct a Craig separator as follows. Let $\sigma = sig(\varphi) \cap sig(\psi)$. Because $\varphi \wedge \psi$ is not satisfiable, for any pair i, i' we find j, j' s.t. $\ell_{ij} \wedge \ell'_{i'j'}$ unsatisfiable. Take any collection P of such pairs $(\ell_{ij}, \ell'_{i'j'})$. Then P can be regarded as a *proof* that $\varphi \wedge \psi$ is not satisfiable. Observe that each pair in P consists of an atom in σ and its negation. Now, let φ_P be the DNF formula obtained from φ by dropping all literals $\ell_{i,j}$ that have no matching partner in P, i.e., that do not occur as first component of some pair in P. It is easy to check that φ_P satisfies all properties of a Craig separator: $\varphi \models \varphi_P$, $sig(\varphi_P) \subseteq \sigma$, and $\varphi_P \wedge \psi$ is unsatisfiable since the pairs in P still show that $\varphi_P \wedge \psi$ is unsatisfiable. Indeed, it is even a Craig-Lyndon separator: Any literal in φ_P occurs as a literal ℓ_{ij} in φ, while its complement occurs as a literal $\ell'_{i'j'}$ in ψ. We obtain the following theorem, which is at the same time our first proof of Theorem 8, namely that propositional logic has the LIP.

▶ **Theorem 23.** *φ_P is a Craig-Lyndon separator for φ, ψ.*

▶ **Example 24.** Let $\varphi = p \wedge q \wedge r$ and $\psi = s \to (p \vee q)$, as in Example 3.iv, and let $\sigma = sig(\varphi) \cap sig(\psi) = \{p, q\}$. Formula φ is already in DNF, a single conjunctive clause. As

$\varphi^{-\sigma}$ we obtain $p \wedge q$ by dropping atom r from this conjunctive clause. The formula $p \wedge q$ is the unique uniform $\{p, q\}$-interpolant for φ. Now consider $\psi' = s \wedge \neg p \wedge \neg q$, the negation of ψ in DNF, also a single conjunctive clause. A collection of pairs $(\ell_{ij}, \ell'_{i'j'})$ as described above is $\{(\ell_{11}, \ell'_{12})\} = \{(p, \neg p)\}$. From this pair, we obtain p as Craig separator for φ, ψ' (and as Craig interpolant for φ, ψ). Another suitable set of pairs is $\{(\ell_{12}, \ell'_{13})\} = \{(q, \neg q)\}$. Based on this we obtain q as another Craig separator. ⌟

9 Interpolants via Resolution

The quantifier elimination method described in Section 7 gives an elegant proof of the CIP, and a simple method for computing Craig and uniform interpolants. A downside however is that the interpolants computed in this way are formulas whose size is always exponential in the number of eliminated atoms. In contrast, the method described in Section 8 does not increase the size of the involved formula. However, it requires these formulas to be in DNF, and transforming arbitrary formulas into DNF can lead to an exponential increase in the size of the formula. Other methods for interpolation try to compute interpolants in a more goal-oriented manner, with the aim of achieving efficiency in many practical cases, even though so far no method is known that can compute interpolants whose size is guaranteed to be sub-exponential, and there might not even exist one (this is discussed in more detail in Section 11).

One such method uses resolution. Resolution is a well-known method for deciding satisfiability of propositional formulas in CNF [4, 23]. As mentioned in Section 2, it is possible to transform any propositional formula into a conservative extension in CNF in linear time. In particular, many structure-preserving transformations such as the well-known Tseitin encoding produce formulas that are not only equi-satisfiable, but also conservative extensions [3, 46], which thanks to the following result is sufficient for computing interpolants. Since we will focus on Craig separators rather than Craig interpolants in this section, we directly formulate it for uniform interpolants and Craig separators.

▶ **Lemma 25.** *Let φ, ψ be formulas such that $\varphi \wedge \psi \models \bot$, let $\sigma \subseteq sig(\varphi)$, and let φ' and ψ' be conservative extensions of φ and ψ that share no atoms that are not also shared by φ and ψ. Then, every uniform σ-interpolant for φ' is also a uniform σ-interpolant for φ, and every Craig separator for φ', ψ' is also a Craig separator for φ, ψ.*

Proof. The first claim follows directly from the definitions. Now let χ be a Craig separator of φ', ψ', which means that $sig(\chi) \subseteq sig(\varphi') \cap sig(\psi')$, $\varphi' \models \chi$ and $\chi \wedge \psi' \models \bot$. By our assumption on the signatures of φ', ψ', we have $sig(\chi) \subseteq sig(\varphi) \cap sig(\psi)$. By the definition of conservative extension, this implies $\varphi \models \chi$ and $\psi \models \neg\chi$, which means that χ is also a Craig separator for φ, ψ. ◀

Resolution for propositional logic operates on formulas in CNF, which in this context are seen as sets of clauses. Its central inference rule is the resolution rule:

$$\frac{C_1 \vee p \qquad C_2 \vee \neg p}{C_1 \vee C_2}$$

Since we represent clauses as sets, we can write this rule also as follows, highlighting the fact that the order of literals is not relevant and duplicates are silently removed, and where we assume that $p \in C_1$ and $\neg p \in C_2$:

$$\frac{C_1 \qquad C_2}{(C_1 \setminus \{p\}) \cup (C_2 \setminus \{\neg p\})}$$

The conclusion of the resolution rule is called *resolvent* of the *parent clauses* C_1 and C_2, and we say that it was derived by *applying resolution upon* p *to* C_1 *and* C_2, the atom that has been eliminated in the resolvent.

It is well-known that resolution is sound and *refutationally complete*, which means that we can derive the *empty clause* from a set N of clauses by a sequence of resolution steps iff N is unsatisfiable. This also holds for a restricted version called *ordered resolution*: here, we assume a linear order $\prec$ on atoms, and apply resolution only upon literals whose atom is maximal wrt. $\prec$ within the respective clauses.

Resolution can also be used to compute Craig interpolants and uniform interpolants, and in both cases, one may argue that resolution operates in a more goal-oriented fashion than the approaches we have seen so far. We first discuss *uniform interpolation*, as this method is easier.

9.1 Uniform Interpolation Using Resolution

Our method for computing uniform interpolants using resolution is based on the SCAN approach introduced for first-order logic in [15]. For formulas in CNF, uniform interpolants can be computed by step-wise application of the following theorem.

▶ **Theorem 26.** *Let* Φ *be a formula in CNF,* p *an atom, and* Ψ *be obtained from* Φ *by*
1. *applying resolution exhaustively upon* p *to all clauses in* Φ*,*
2. *removing all clauses that contain* p*.*

Then, $\Psi \equiv \exists p.\Phi$*.*

Before we prove the theorem, we illustrate on an example how it is used to compute uniform interpolants.

▶ **Example 27.** Consider the formula

$$\varphi = \neg(d \wedge e) \wedge (a \rightarrow d) \wedge (a \vee c) \wedge e.$$

We want to compute a uniform $\{a, d\}$-interpolant for φ. We first compute the CNF Φ equivalent to φ, resulting in the following clauses:

1. $\neg d \vee \neg e$ *input clause*
2. $\neg a \vee d$ *input clause*
3. $a \vee c$ *input clause*
4. e *input clause*

We first compute a propositional formula equivalent to $\exists d.\Phi$ by applying resolution upon d:

5. $\neg a \vee \neg e$ *resolvent of 1, 2 upon d*

$\exists d.\Phi$ is equivalent to the conjunction of Clauses 3–5, the clauses that do not contain d. We continue by computing the propositional formula equivalent to $\exists a.\exists d.\Phi$, for which we now have to apply resolution upon a to the remaining clauses:

6. $c \vee \neg e$ *resolvent of 3, 5 upon a*

After removing all clauses that contain a, we obtain the uniform $\{a, d\}$-interpolant for φ, which is the conjunction of Clause 4 and 6, namely $e \wedge (c \vee \neg e) \equiv e \wedge c$. ⌟

Proof of Theorem 26. We can prove the theorem in two ways. While the first proof is more straightforward, the second one uses an idea that is also used for more expressive logics, but is much easier to follow in the case of propositional logic (e.g. see Chapter 15).

Variant 1: Using quantifier elimination. Assume that no clause contains both p and $\neg p$. (Since such clauses are tautologies, we can safely remove them.) By factoring out occurrences of p and $\neg p$ respectively, we can transform Φ into an equivalent formula of the following form, where Φ_1, Φ_2 and Φ_3 are formulas in CNF that contain no occurrences of p:

$$\Phi_1 \wedge (\Phi_2 \vee p) \wedge (\Phi_3 \vee \neg p) \tag{1}$$

If we use the quantifier elimination technique from Section 7 on p in this formula, we obtain the following formula that is equivalent to $\exists p.\Phi$

$$\begin{aligned}
&(\Phi_1 \wedge (\Phi_2 \vee \top) \wedge (\Phi_3 \vee \neg\top)) \vee (\Phi_1 \wedge (\Phi_2 \vee \bot) \wedge (\Phi_3 \vee \neg\bot)) \\
\equiv &(\Phi_1 \wedge \top \wedge \Phi_3) \vee (\Phi_1 \wedge \Phi_2 \wedge \top) \\
\equiv\ &\Phi_1 \wedge (\Phi_3 \vee \Phi_2)
\end{aligned}$$

We can transform $(\Phi_3 \vee \Phi_2)$ back into CNF using distributivity: the resulting formula is a conjunction over all clauses that can be obtained by combining a clause from Φ_3 with another clause from Φ_2. Note that those clauses are exactly those that we obtain when applying resolution upon p in the original formula, so that the resulting set of clauses is indeed the CNF formula Ψ that is computed as in the theorem, so that indeed $\Psi \equiv \exists p.\Phi$. ◀

Variant 2: Using ordered resolution. For this proof, we use refutational completeness of ordered resolution, where we use an ordering $\prec_p$ that makes p maximal. We need to show that Ψ is a uniform Σ-interpolant of Φ for $\Sigma = sig(\Phi) \setminus \{p\}$. The first two conditions in Definition 10 hold because 1) the resolvent is always entailed by its parent clauses, and 2) we explicitly removed all occurrences of p. It remains to show that for any formula φ that does not contain p, $\Psi \models \varphi$ if $\Phi \models \varphi$. Since $\Phi \models \varphi$ iff $\Phi \wedge \neg\varphi \models \bot$, we can do so by showing that for any formula Θ in CNF s.t. $p \notin sig(\Theta)$, $\Phi \cup \Theta \models \bot$ implies $\Psi \cup \Theta \models \bot$. We can decide $\Phi \cup \Theta \models \bot$ using ordered resolution with the ordering $\prec_p$. In particular, this means that we first compute all resolvents upon p (which only occurs in Φ, but not in Θ), and then ignore occurrences of p while we continue. Effectively, we thus first compute Ψ, and then continue on $\Psi \cup \Theta$. Thus, if the empty clause can be derived from $\Phi \cup \Theta$, it can also be derived from $\Psi \cup \Theta$. ◀

9.2 Craig Interpolation Using Resolution

It is more convenient to look at Craig separators rather than Craig interpolants. To compute a Craig separator using resolution, we follow the method by Huang [20]. Let Φ and Ψ be sets of clauses s.t. $\Phi \cup \Psi \models \bot$. The idea is to construct a Craig separator for Φ, Ψ by following the clauses used in the *resolution proof for* $\Phi \cup \Psi \models \bot$, by which we simply refer to the sequence of resolution steps that are used to derive the empty clause from $\Phi \cup \Psi$. For this, we annotate each clause φ in the proof—including the input clauses—with a formula $\chi[\varphi]$ that serves as an *intermediate separator*.

▶ **Definition 28.** *Let θ be a clause. A formula χ is an* intermediate separator of Φ and Ψ relative to θ *if*

1. $sig(\chi) \subseteq sig(\Phi) \cap sig(\Psi)$,
2. $\Phi \models \theta \vee \chi$, *and*
3. $\Psi \models \theta \vee \neg\chi$.

The following corollary follows directly from the definition:

▶ **Corollary 29.** *χ is a Craig separator for Φ and Ψ iff it is an intermediate separator of Φ and Ψ relative to the empty clause.*

The clauses in the resolution proof are annotated inductively, starting from the input clauses, and then following each resolution step. This means that the intermediate separator for each derived clause is based on the intermediate separators for the clauses that we apply resolution to. How we build the next intermediate separator depends on the atom upon which we resolve. Intuitively, the atom has to go into the formula if it is from the common signature of Φ and Ψ. Otherwise, we have to combine the annotations of the parent clauses in a clever way.

We first annotate the clauses in $\Phi \cup \Psi$. We annotate every $\theta \in \Phi$ with $\chi[\theta] = \bot$, and every $\theta \in \Psi$ with $\chi[\theta] = \top$. If a clause θ is a resolvent of the parent clauses θ_1 and θ_2 upon p,

where θ_1 and θ_2 are already annotated, we annotate θ with $\chi[\theta]$, which is defined based on the origin of p:

1. if $p \in sig(\Phi) \cap sig(\Psi)$, then $\chi[\theta] = (p \vee \chi[\theta_1]) \wedge (\neg p \vee \chi[\theta_2])$;
2. if $p \in sig(\Phi) \setminus sig(\Psi)$, then $\chi[\theta] = \chi[\theta_1] \vee \chi[\theta_2]$;
3. if $p \in sig(\Psi) \setminus sig(\Phi)$, then $\chi[\theta] = \chi[\theta_1] \wedge \chi[\theta_2]$.

We illustrate the technique with an example.

▶ **Example 30.** We use again the formula

$$\varphi = \neg(d \wedge e) \wedge (a \to d) \wedge (a \vee c) \wedge e$$

from Example 27. It turns out that $\varphi \models \psi$, where

$$\psi = (b \to c) \wedge (d \to f)$$

To compute a Craig interpolant for φ, ψ, we compute a Craig separator for $\varphi, \neg\psi$. We first need to transform these formulas into CNF. We already produced the CNF Φ of φ. For $\neg\psi$, its CNF Ψ is obtained as follows:

$$\begin{aligned} & \neg\big((b \to c) \wedge (d \to f)\big) \\ \equiv\ & (b \wedge \neg c) \vee (d \wedge \neg f) \\ \equiv\ & (b \vee d) \wedge (b \vee \neg f) \wedge (\neg c \vee d) \wedge (\neg c \vee \neg f) \end{aligned}$$

We derive the empty clause from $\Phi \cup \Psi$, and annotate each clause with an intermediate separator, which for readability we simplify as we go along.

clause θ	origin	annotation $\chi[\theta]$
1. $\neg d \vee \neg e$	Φ	$\bot$
2. $\neg a \vee d$	Φ	$\bot$
3. $a \vee c$	Φ	$\bot$
4. e	Φ	$\bot$
5. $b \vee d$	Ψ	$\top$
6. $b \vee \neg f$	Ψ	$\top$
7. $\neg c \vee d$	Ψ	$\top$
8. $\neg c \vee \neg f$	Ψ	$\top$
9. $a \vee d$	resolvent of 3, 7	$(c \vee \bot) \wedge (\neg c \vee \top) \equiv c$
10. d	resolvent of 9, 2	$(c \vee \bot) \equiv c$
11. $\neg e$	resolvent of 10, 1	$(d \vee c) \wedge (\neg d \vee \bot) \equiv (c \wedge \neg d)$
12. $\bot$	resolvent of 4, 11	$(c \wedge \neg d) \vee \bot \equiv (c \wedge \neg d)$

We obtain that the Craig separator for Φ, Ψ, and hence the Craig interpolant for φ, ψ is $c \wedge \neg d$, and indeed, one can confirm that

$$\begin{aligned} & \neg(d \wedge e) \wedge (a \to d) \wedge (a \vee c) \wedge e \\ \models\ & c \wedge \neg d \\ \models\ & (b \to c) \wedge (d \to f). \end{aligned}$$

⌟

It remains to show that the method is also correct. By Corollary 29, we only need to show that the annotations really correspond to intermediate separators.

▶ **Lemma 31.** *For every clause θ in the resolution proof, $\chi[\theta]$ is an intermediate separator for Φ and Ψ relative to θ.*

Proof. Our construction ensures that all formulas are within the signature $sig(\Phi) \cap sig(\Psi)$, which means we only need to show the other two conditions in Definition 28. For clauses directly occuring in Φ and Ψ, our construction makes these conditions trivially true: In particular, if $\theta \in \Phi$, then $\Phi \models \theta \vee \bot$ and $\Psi \models \theta \vee \neg\bot$, and if $\theta \in \Psi$, then $\Phi \models \theta \vee \top$ and $\Psi \models \theta \vee \neg\top$.

For the derived clauses, we show Conditions 2 and 3 by induction. Let $\theta = \theta'_1 \vee \theta'_2$ be the resolvent of two clauses $\theta_1 = \theta'_1 \vee p$ and $\theta_2 = \theta'_2 \vee \neg p$, which are annotated by formulas $\chi[\theta_1]$ and $\chi[\theta_2]$ for which we have already established that they are intermediate interpolants. We distinguish the different cases in the construction of $\chi[\theta]$, and show that in each case, $\Phi \models \theta \vee \chi[\theta]$ and $\Psi \models \theta \vee \neg\chi[\theta]$.

1. $p \in sig(\Phi) \cap sig(\Psi)$. Then, $\chi[\theta] = (p \vee \chi[\theta_1]) \wedge (\neg p \vee \chi[\theta_2])$. We first show that $\Phi \models \theta \vee \chi[\theta]$. Let v be a model s.t. $v \models \Phi$ and $v \not\models \theta$. By our inductive hypothesis, $v \models \theta'_1 \vee p \vee \chi[\theta_1]$ and $v \models \theta'_2 \vee \neg p \vee \chi[\theta_2]$. Because $v \not\models \theta$ and $\theta = \theta'_1 \vee \theta'_2$, neither θ'_1 nor θ'_2 can be satisfied in v, so that we obtain $v \models p \vee \chi[\theta_1]$ and $v \models \neg p \vee \chi[\theta_2]$. This directly gives us $v \models \chi[\theta]$, which establishes that for all models v s.t. $v \models \Phi$, $v \models \theta \vee \chi[\theta]$, and thus $\Phi \models \theta \vee \chi[\theta]$.
   We next show $\Psi \models \theta \vee \neg\chi[\theta]$, for which we again take a model v s.t. $v \models \Psi$ and $v \not\models \theta$. Our inductive hypothesis now gives us $v \models \theta'_1 \vee p \vee \neg\chi[\theta_1]$ and $v \models \theta'_2 \vee \neg p \vee \neg\chi[\theta_2]$, and we can again use $v \not\models \theta$ to obtain

   $$v \models p \vee \neg\chi[\theta_1] \tag{2}$$
   $$v \models \neg p \vee \neg\chi[\theta_2] \tag{3}$$

   We need to show $v \not\models (p \vee \chi[\theta_1]) \wedge (\neg p \vee \chi[\theta_2])$. Assume that v does satisfy the first conjunct, that is,

   $$v \models p \vee \chi[\theta_1] \tag{4}$$

   From this, we can use Equation (2) to obtain $v \models p$, which together with Equation (3) gives us $v \models \neg\chi[\theta_2]$. This implies that $v \not\models \neg p \vee \chi[\theta_2]$. Since this was a consequence of our assumption in Equation (4), we obtain $v \not\models (p \vee \chi[\theta_1]) \wedge (\neg p \vee \chi[\theta_2]) = \chi[\theta]$, and thus $\Psi \models \theta \vee \neg\chi[\theta]$.

2. $p \in sig(\Phi) \setminus sig(\Psi)$. By the inductive hypothesis, $\Phi \models \theta_1 \vee \chi[\theta_1]$ and $\Phi \models \theta_2 \vee \chi[\theta_2]$. We can weaken these disjunctions to $\theta_1 \vee (\chi[\theta_1] \vee \chi[\theta_2])$ and $\theta_2 \vee (\chi[\theta_1] \vee \chi[\theta_2])$, so that we can factor out $\chi[\theta_1] \vee \chi[\theta_2] = \chi[\theta]$, resulting in $\Phi \models (\theta_1 \wedge \theta_2) \vee \chi[\theta]$. Because θ was inferred from θ_1 and θ_2 through resolution, we have $\theta_1 \wedge \theta_2 \models \theta$, so that we obtain $\Phi \models \theta \vee \chi[\theta]$. Next, we show $\Psi \models \theta \vee \neg\chi[\theta]$. By inductive hypothesis, $\Psi \models \theta'_1 \vee p \vee \neg\chi[\theta_1]$ and $\Psi \models \theta'_2 \vee \neg p \vee \neg\chi[\theta_1]$. Since p does not occur in Ψ, this implies $\Psi \models \theta'_1 \vee \neg\chi[\theta_1]$ and $\Psi \models \theta'_2 \vee \neg\chi[\theta_2]$. We can weaken θ'_1 and θ'_2 to $\theta'_1 \vee \theta'_2 = \theta$, to obtain $\Psi \models \theta \vee \neg\chi[\theta_1]$ and $\Psi \models \theta \vee \neg\chi[\theta_2]$. Factoring out θ, we obtain $\Psi \models \theta \vee (\neg\chi[\theta_1] \wedge \neg\chi[\theta_2]) = \theta \vee \neg(\chi[\theta_1] \vee \chi[\theta_2]) = \theta \vee \neg\chi[\theta]$.
3. $p \in sig(\Psi) \setminus sig(\Phi)$. This case is shown as the previous one, with the roles of Φ and Ψ reversed. ◀

▶ **Lemma 32.** *Let Φ, Ψ be some set of clauses s.t. $\Phi \cup \Psi \models \bot$, and Θ be a sequence of clauses deriving the empty clause from $\Phi \cup \Psi$, where each clause is annotated as above. Then, the empty clause is annotated with a formula $\chi[\bot]$ which is a Craig separator for Φ and Ψ.*

A nice property of the construction is that we can construct from a given resolution proof for $\Phi \cup \Psi \models \bot$ a Craig separator whose size is polynomial in the number of inference steps in the proof. This is a desirable property of proof systems in general, that is commonly referred to as *feasible interpolation*. A more detailed discussion on feasible interpolation is provided in Chapter 6. The interested reader might also consult [27, 28, 36].

▶ **Theorem 33.** *Given a resolution proof for $\Phi \cup \Psi \models \bot$, we can in polynomial time construct a Craig separator for Φ, Ψ whose size is linearly bounded by the size of the proof.*

Proof. By induction on the number of steps in the proof, we obtain that each clause is annotated by a formula whose size is polynomial in the number of proof steps that lead to this clause. For the input clauses, this holds trivially. By analyzing the different cases for how we annotate new resolvents, we see that each intermediate separator introduces at most five new subformulas. Consequently, the (dag-)size of the computed separator is at most five times the number of inference steps, and thus linearly bounded in the size of the proof. ◀

Because of the case where we resolve upon some $p \in sig(\Phi) \cap sig(\Psi)$, which introduces both p and $\neg p$ into the Craig interpolant, the interpolants computed with this method do not satisfy the Lyndon property. There are however alternative approaches, following the same idea, but using a different way of annotating clauses, that do preserve the Lyndon property. One such example is the interpolation system by McMillan [30, 32]. Here, each input clause $\theta \in \Phi$ is annotated with the clause $\chi[\theta]$ that is obtained from θ by removing literals that use atoms from $sig(\Phi) \setminus sig(\Psi)$, that is, keeping only those literals that are in the target signature. Clauses in Ψ are still annotated with $\top$, and clauses θ obtained through resolution upon p on clauses θ_1, θ_2 are annotated as before, except for the case where $p \in sig(\Phi) \cap sig(\Psi)$, in which case we set $\chi[\theta] = \chi[\theta_1] \wedge \chi[\theta_2]$ (as in the case for $p \in sig(\Psi) \setminus sig(\Phi)$).

9.3 Bibliographic and Historic Remarks

Resolution was introduced in 1965 by J. Alan Robinson [38] as a machine-suited proof method for first-order logic. For propositional logic, it was, in the dual form for DNF, already established since 1955 as the *consensus method* by Willard Van Orman Quine [37], which is applied to convert a given DNF to a form that consists of all its prime implicants (minimal conjunctive clauses without complementary literals that imply the overall formula). The first appearance of propositional resolvent/consensus construction is attributed to Archie Blake in 1937 [6, 9]. Our resolution-based method for uniform interpolation from Section 9.1 has ancestors in a more general method for second-order quantifier elimination on the basis of first-order logic developed by Wilhelm Ackermann in 1935 [1], which was rediscovered in 1992 by Dov M. Gabbay and Hans Jürgen Ohlbach [15]. In its dual variation, our method for propositional logic is attributed to Edward W. Samson and Rolf K. Mueller in 1955 [42] (see [24, Sect. 7.2.2.2]). They address the tautology problem by eliminating all atoms from a given propositional DNF one by one with consensus. The system for Craig interpolation shown in Section 9.2 is called *HKPYM* by Maria Paola Bonacina and Moa Johansson in their survey on ground interpolation systems [8], reflecting the initials of several authors who apparently discovered it independently, Guoxiang Huang in 1995 [20] (actually for first-order clauses), Jan Krajícek [28] in 1997, and Pavel Pudlák [36] also in 1997, as well as Greta Yorsh and Madanlal Musuvathi [47] for contributing in 2005 a completeness proof. In their survey, Bonacina and Johansson also present a second resolution-based propositional interpolation system by Kenneth L. McMillan from 2003 [30, 32] in a form that allows easy comparison.

A comprehensive presentation of resolution in propositional logic is included in the monograph *Propositional Logic – Deduction and Algorithms* by Hans Kleine Büning and Theodor Lettmann [23]. As introductory presentations of resolution in general we recommend the classical textbook by Chin-Liang Chang and Richard Char-Tung Lee [10], and the handbook article by Leo Bachmair and Harald Ganzinger [4].

10 Craig Interpolants via Analytic Tableaux

In this section we provide a further proof of the CIP, or actually the stronger LIP, via a *tableau* proof system. As for resolution, we adopt the *separator* view, since tableau systems are most commonly modeled as *refutational* systems that construct a tree-structured proof, a *closed tableau*, which certifies unsatisfiability of a given formula. Again, as for resolution, computation of a Craig separator χ for φ, ψ proceeds in two phases: first, a proof of the unsatisfiability of $\varphi \wedge \psi$ is obtained, which is here the tableau; second, a separator χ is extracted from the proof. As we will see, extraction from a tableau is a straightforward induction over its tree structure. We describe our proof system in Section 10.1 and the separator extraction in Section 10.2. The sources of the underlying technique are the monographs by Fitting [13] and Smullyan [44]. Its variation for sequent systems is known as *Maehara's method* (see Chapter 5).

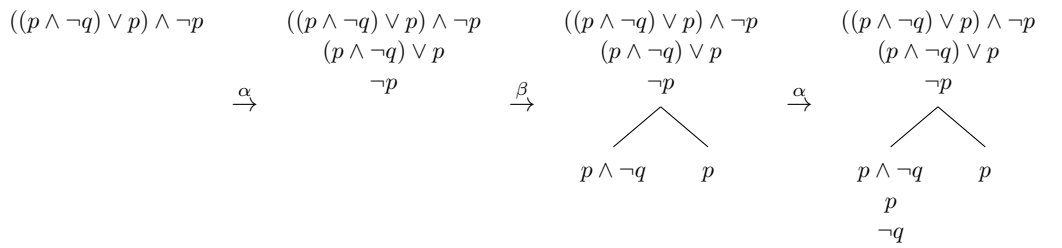
10.1 A Tableau Proof System

We consider a very simple analytic tableau system that operates on propositional formulas in NNF. It constructs a *tableau*, a tree whose nodes are labeled by formulas. For our system, this is an ordered binary tree. As common for tableau systems, it is a refutation system, that is, it aims at showing unsatisfiability of a given formula φ. A tableau is said to be *for* this given formula φ. The tableau construction starts with a single-node tableau labeled with φ. A tableau branch is *closed* if it contains a node labeled with $\bot$ or contains two nodes labeled with complementary literals, otherwise it is called *open*. A tableau is called *closed* if all of its branches are closed. The tableau construction repeatedly extends an open branch of the tableau by applying one of the following two *tableau expansion rules*.

$$\frac{\varphi_1 \wedge \varphi_2}{\begin{array}{c}\varphi_1\\ \varphi_2\end{array}}\ \alpha \qquad\qquad \frac{\varphi_1 \vee \varphi_2}{\varphi_1 \mid \varphi_2}\ \beta$$

Applying rule α means selecting an open branch and a conjunction $\varphi_1 \wedge \varphi_2$ on the branch, and then adding to the end of the branch a node labeled with φ_1 and another node with φ_2 as child of the latter one. Applying rule β means selecting an open branch and a disjunction $\varphi_1 \vee \varphi_2$ on the branch, and then adding to the end of the branch a left child labeled with φ_1 and a right child labeled with φ_2. The objective of the system is to construct a tableau for φ that is closed, which indicates that φ is unsatisfiable.

▶ **Example 34.** We apply our tableau system to construct a closed tableau for the formula $\varphi = ((p \wedge \neg q) \vee p) \wedge \neg p$, the negation of $((p \to q) \to p) \to p$ as NNF.



⌟

Our tableau system is *sound*, that is, if it constructs a closed tableau for φ, then φ is indeed unsatisfiable. One way to see this is via considering tableau construction as an equivalence-preserving formula transformation, similar to DNF transformation: We associate with a tableau a specific formula, the disjunction of, for each branch, the conjunction of the branch's formula labels. For the initial single-node tableau, the associated formula is then just the given formula φ. The associated formula is preserved up to equivalence by applications of the rules α and β. If the tableau is closed, then each branch is closed, that is, contains a

pair of complementary literals or $\bot$. Hence, for each branch the conjunction of its formula labels is then equivalent to $\bot$. Thus, also the overall formula associated with the tableau, the disjunction of the formulas for the branches, is equivalent to $\bot$. It follows that if a tableau for φ can be constructed that is closed, then $\varphi \equiv \bot$, i.e., φ is indeed unsatisfiable.

The system is also *complete*, that is, if φ is unsatisfiable, then it constructs a closed tableau for φ. For proofs we refer to the monographs by Smullyan [44] or Fitting [13], where completeness of similar systems is discussed.

10.2 Craig-Lyndon Separator Extraction from a Closed Tableau

A closed tableau for a given conjunction $\varphi \wedge \psi$ allows to extract a *Craig-Lyndon separator* χ for the conjuncts φ, ψ. To enable this, we enhance our tableau system by a means for keeping track of provenance information, whether a formula in the tableau stems from φ or from ψ. This is represented by wrapping formulas stemming from φ in $L(_)$ (suggesting *left side*) and formulas stemming from ψ in $R(_)$ (suggesting *right side*). We call formulas decorated with such provenance information *biased formulas*. The tableau expansion rules α, β now come in *biased* versions, where provenance information is propagated.

$$\frac{L(\varphi_1 \wedge \varphi_2)}{\begin{array}{c}L(\varphi_1)\\ L(\varphi_2)\end{array}}\ \alpha_L \qquad \frac{R(\varphi_1 \wedge \varphi_2)}{\begin{array}{c}R(\varphi_1)\\ R(\varphi_2)\end{array}}\ \alpha_R \qquad \frac{L(\varphi_1 \vee \varphi_2)}{L(\varphi_1) \mid L(\varphi_2)}\ \beta_L \qquad \frac{R(\varphi_1 \vee \varphi_2)}{R(\varphi_1) \mid R(\varphi_2)}\ \beta_R$$

The provenance indicators L and R are ignored in determining whether a branch is closed. Thus, a branch can be closed in the following ways: by a pair $L(\ell), L(\overline{\ell})$, by a pair $R(\ell), R(\overline{\ell})$, by a pair $L(\ell), R(\overline{\ell})$, by $L(\bot)$, and by $R(\bot)$.

Let S be a set of biased formulas. We say that a formula χ *separates* S, symbolically expressed as $S \overset{sep}{\rightarrow} \chi$, iff χ is a Craig-Lyndon separator for $\bigwedge_{L(\theta)\in S} \theta,\ \bigwedge_{R(\theta)\in S} \theta$. The following lemma gathers the properties of $\overset{sep}{\rightarrow}$ that underly separator extraction.

▶ **Lemma 35.** *Let S be a set of biased formulas. It then holds for literals ℓ and formulas $\varphi_1, \varphi_2, \chi, \chi_1, \chi_2$ that*

- **(i)** $S \cup \{L(\ell)\} \cup \{L(\overline{\ell})\} \overset{sep}{\rightarrow} \bot$
- **(ii)** $S \cup \{R(\ell)\} \cup \{R(\overline{\ell})\} \overset{sep}{\rightarrow} \top$
- **(iii)** $S \cup \{L(\ell)\} \cup \{R(\overline{\ell})\} \overset{sep}{\rightarrow} \ell$
- **(iv)** $S \cup \{L(\bot)\} \overset{sep}{\rightarrow} \bot$
- **(v)** $S \cup \{R(\bot)\} \overset{sep}{\rightarrow} \top$
- **(vi)** *If* $S \cup \{L(\varphi_1), L(\varphi_2)\} \overset{sep}{\rightarrow} \chi$, *then* $S \cup \{L(\varphi_1 \wedge \varphi_2)\} \overset{sep}{\rightarrow} \chi$
- **(vii)** *If* $S \cup \{R(\varphi_1), R(\varphi_2)\} \overset{sep}{\rightarrow} \chi$, *then* $S \cup \{R(\varphi_1 \wedge \varphi_2)\} \overset{sep}{\rightarrow} \chi$
- **(viii)** *If* $S \cup \{L(\varphi_1)\} \overset{sep}{\rightarrow} \chi_1$ *and* $S \cup \{L(\varphi_2)\} \overset{sep}{\rightarrow} \chi_2$, *then* $S \cup \{L(\varphi_1 \vee \varphi_2)\} \overset{sep}{\rightarrow} (\chi_1 \vee \chi_2)$
- **(ix)** *If* $S \cup \{R(\varphi_1)\} \overset{sep}{\rightarrow} \chi_1$ *and* $S \cup \{R(\varphi_2)\} \overset{sep}{\rightarrow} \chi_2$, *then* $S \cup \{R(\varphi_1 \vee \varphi_2)\} \overset{sep}{\rightarrow} (\chi_1 \wedge \chi_2)$

Lemma 35 is easy to verify. For example, Lemma 35.iii expresses that, for arbitrary sets S of biased formulas, the literal ℓ is a Craig-Lyndon separator for $\bigwedge_{L(\theta)\in S}\theta \wedge \ell$, $\bigwedge_{R(\theta)\in S}\theta \wedge \overline{\ell}$. In case ℓ is a positive literal p, this holds, since $\bigwedge_{L(\theta)\in S}\theta \wedge p \models p$, $\bigwedge_{R(\theta)\in S}\theta \wedge \neg p \wedge p \models \bot$, and $sig^{\pm}(p) = \{\langle p, +\rangle\} \subseteq sig^{\pm}(\bigwedge_{L(\theta)\in S}\theta \wedge p) \cap sig^{\pm}(\neg(\bigwedge_{R(\theta)\in S}\theta \wedge \neg p))$. The case for ℓ a negative literal $\neg p$ holds analogously.

As basis for Craig-Lyndon separator extraction, we build a closed biased tableau *for* φ, ψ with the biased rules, starting from the initial biased tableau

$L(\varphi)$
$R(\psi)$

Lemma 35 then straightforwardly suggests a recursive separator extraction procedure. We start with the closed tableau and determine as base cases for each of its branches a separator, according to one of the Lemmas 35.i–35.v. Since each branch is closed, it must match with at least one of these lemmas.

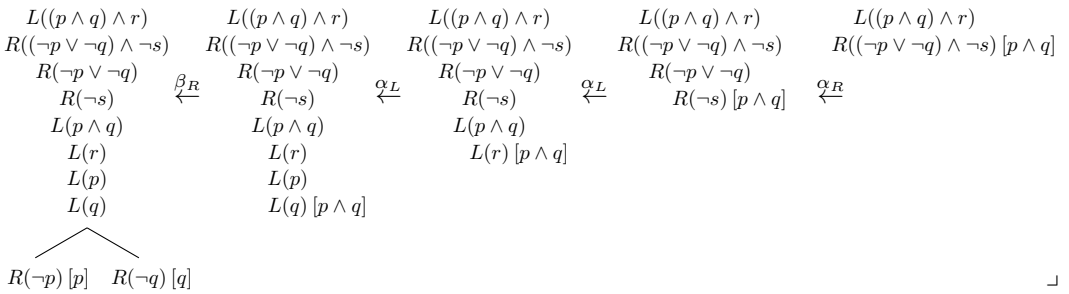
We then proceed by considering each application of an expansion rule in the tableau construction in the *reverse order* of their application. If the last applied rule was α_L, the branch after rule application matches the if-condition of Lemma 35.vi and the shorter branch before rule application matches the conclusion of the lemma. Hence, the formula χ, which separates the branch after rule application, also separates the shorter branch before rule application. Analogously, α_R corresponds to Lemma 35.vii. If the last applied rule was β_L, we have two branches, matching the two if-conditions of Lemma 35.viii, and the branch before rule application matches the conclusion of the lemma. Hence, from the two formulas χ_1 and χ_2, each separating one of the two branches after rule application, we build the disjunction $\chi_1 \vee \chi_2$, which separates the shorter common branch before rule application. Analogously, β_R corresponds to Lemma 35.ix, where the separator for the shorter branch before rule application is the conjunction $\chi_1 \wedge \chi_2$.

Finally, after having processed the very first rule application, we obtain a formula χ that separates $\{L(\varphi), R(\psi)\}$, the initial branch with which we started the tableau construction. In other words, χ is the desired Craig-Lyndon separator for φ, ψ, and the extraction is completed.

If we have a given closed tableau for $\varphi \wedge \psi$ that is "unbiased", obtained with the plain proof system of Section 10.1, it can easily be preprocessed to a *biased* tableau for φ, ψ: The initial application of α that leads to φ followed by ψ is dropped, such that the proof only starts with φ followed by ψ. If this occurrence of φ is replaced by $L(\varphi)$ and the occurrence of ψ by $R(\varphi)$, then all rule applications can be replaced by biased versions that propagate the provenance information downwards.

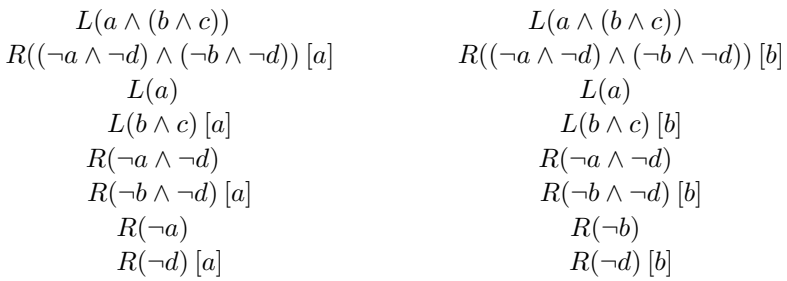
▶ **Example 36.** To illustrate Craig-Lyndon separator extraction, we start from a closed biased tableau for φ, ψ with $\varphi = (p \wedge q) \wedge r$ and $\psi = (\neg p \vee \neg q) \wedge \neg s$, where separator values for closed branches according to Lemma 35.iii are annotated at the leaves. Then we undo the extension steps that led to the closed tableau, propagating the separator values upwards

according to the respective cases of Lemma 35. We finally arrive at the initial tableau with two nodes $L(\varphi)$ and $R(\psi)$ and obtain $\chi = p \wedge q$ as separator for φ, ψ.



▶ **Example 37.** Figure 1 shows a more complex example of the separator extraction. Differently from Example 36, we do not exhibit the undone tableau extension steps but simultaneously annotate all nodes that were leaves during the tableau construction with the respective separators. The given formulas are $\varphi = d \wedge (\neg d \vee (a \wedge (b \vee c)))$ and $\psi = (b \wedge (\neg b \vee (\neg a \wedge \neg e))) \vee (\neg b \wedge \neg c)$. The obtained separator is $\bot \vee ((\top \wedge a) \wedge (b \vee c))$, which simplifies to $a \wedge (b \vee c)$. ⌟

▶ **Example 38.** Two different proofs of the unsatisfiability of $\varphi \wedge \psi$ can lead to different extracted separators. Here is a simple example. The given formulas are $\varphi = a \wedge (b \wedge c)$ and $\psi = (\neg a \wedge \neg d) \wedge (\neg b \wedge \neg d)$. As separators we obtain a and b, depending on the closed biased tableau for φ, ψ. As in Example 37, we simultaneously annotate all nodes that were leaves during the tableau construction with the respective separators.



We can now summarize the results of this section in a proof statement of the Craig-Lyndon Interpolation Property (LIP, Theorem 8), expressed in terms of Craig-Lyndon separation.

Proof (Craig-Lyndon Interpolation Property (LIP)). Let φ, ψ be propositional formulas such that $\varphi \wedge \psi$ is unsatisfiable. We assume that both formulas are in NNF, which is w.l.o.g. as conversion to an equivalent NNF is always possible in linear time. Since the tableau proof system from Section 10.1 is complete, it yields a closed tableau for $\varphi \wedge \psi$. By adding provenance indicators it can be converted to a biased tableau for φ, ψ. Based on Lemma 35, from the biased tableau we can extract a formula χ such that $\{L(\varphi), R(\psi)\} \overset{sep}{\to} \chi$, that is, χ is a Craig-Lyndon separator for φ, ψ. ◀

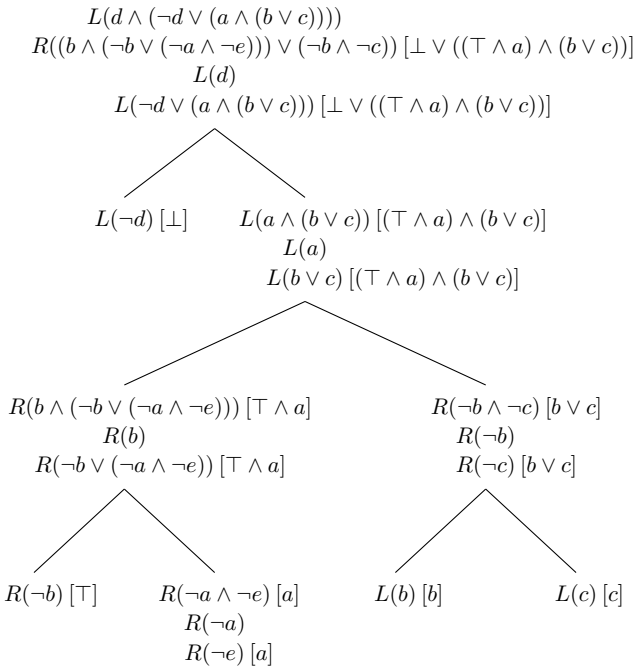


Figure 1 A more complex example of Craig-Lyndon separator extraction, discussed in Example 37.

This proof permits the following corollary about computational effort as well as the shape and size of the separator in relation to the size of the underlying proof, i.e., the tableau.

▶ **Corollary 39.** *The time for extracting a Craig-Lyndon separator from a closed tableau is polynomial in the number of tableau nodes. The separator is in NNF and its size is at most linear in the number of tableau nodes.*

That the separator extraction is in polynomial time can be seen as follows. The extraction is performed in steps that each correspond to a dedicated tableau node: to a leaf for each closed branch (Lemmas 35.i–35.v), and to an inner node for each application of an extension rule (Lemmas 35.vi–35.ix). Within each of these steps, a separating formula is determined according to the respective lemma, which is clearly polynomial in the number of tableau nodes, as it involves matching of only one or two branches with the preconditions of the respective lemma, and the effort for constructing the separating formula is at most building the disjunction or conjunction of two given formulas.

The NNF shape and the size bound follow since the extracted separator is built by disjunction and conjunction from exactly one occurrence of a literal or truth value constant for each tableau leaf.

10.3 Bibliographic and Historic Remarks

Analytic tableaux became widely known with Raymond Smullyan's compact monograph on first-order logic from 1968 [44]. They are related to earlier systems by, among others, Jaakko Hintikka 1955 [19] and Evert W. Beth 1959 [5]. For a comprehensive historic account, see Melvin Fitting's introduction [14] to the *Handbook of Tableau Methods* [12]. A monograph by Fitting [13] and a handbook article by Reiner Hähnle [16] give presentations of tableaux that take aspects of automated reasoning into account. Our interpolant extraction from analytic tableaux is based on the monograph by Fitting from 1996 [13] and the earlier monograph by Smullyan [44]. Fitting specifies it by means of a rule system. Similar interpolation methods were presented on the basis of semantic tableaux by Beth in 1959 [5] and on the basis of sequent calculi by Shoji Maehara in 1960 [29, 45] and also by Smullyan in 1968 [44].

11 The Size of Interpolants

The algorithms computing Craig interpolants discussed above all output interpolants of exponential size in the size of the input formulas, in the worst case. The only exception is the DNF based algorithm which constructs interpolants by dropping subformulas from the left-hand side of the input entailment. However, since the transformation of an arbitrary formula into a logically equivalent formula in DNF can lead to an exponential increase in the size of the formula, also the DNF based algorithm outputs exponential size interpolants when applied to arbitrary input formulas. In fact, no algorithm computing Craig interpolants of polynomial size is known and it is widely conjectured that none exists. Proving this conjecture is hard, however, and would solve longstanding open problems in complexity theory. Even for uniform interpolants, no exponential (and not even superpolynomial) lower bounds are known. In this section, we discuss results that formalise this claim. We also refer the reader to Chapter 6, where many of the concepts used in this section are introduced in detail as part of a discussion of the role of Craig interpolation in proof complexity.

Recall that we denote by $|\varphi|$ and $s(\varphi)$ the dag- and, respectively, tree-size of a propositional formula φ. It is conjectured that there is a superpolynomial gap between the dag-size and the tree-size of propositional formulas. This conjecture remains open [39]. The following observation states a close link between this conjecture and the tree-size of Craig interpolants. We say that propositional logic has *polynomial tree-size Craig interpolants* if there is a polynomial function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for any φ, ψ with $\varphi \models \psi$ there exists a Craig interpolant χ for φ, ψ with $s(\chi) \leq f(|\varphi|, |\psi|)$.

▶ **Theorem 40.** *If there is a superpolynomial gap between dag-size and tree-size, then propositional logic does not have polynomial tree-size Craig interpolants.*

Proof. Assume there is a sequence of propositional formulas $(\varphi_n)_{n\in\mathbb{N}}$ such that the tree-size of any equivalent propositional formulas grows superpolynomially compared to the dag-size of the formulas φ_n. Introduce for every subformula ψ of φ_n an atom p_ψ and let ψ_n be the

conjunction of all

$$p_q \leftrightarrow q, \quad p_\top \leftrightarrow \top, \quad p_\bot \leftrightarrow \bot, \quad p_{\psi\wedge\chi} \leftrightarrow p_\psi \wedge p_\chi, \quad p_{\psi\vee\chi} \leftrightarrow p_\psi \vee p_\chi, \quad p_{\neg\psi} \leftrightarrow \neg p_\psi$$

with q ranging over atoms in φ_n, and $\psi \wedge \chi$, $\psi \vee \chi$, and $\neg\psi$ ranging over subformulas of φ_n. Let $\sigma_n = sig(\varphi_n)$. Then $\psi_n \models p_{\varphi_n} \leftrightarrow \varphi_n$ and so we have explicit σ_n-definitions of p_{φ_n} under ψ_n. There cannot be polynomial tree-size explicit σ_n-definitions of p_{φ_n} under ψ_n since these would be propositional formulas equivalent to φ_n of polynomial tree-size. It follows from the proof of Theorem 16 that Craig interpolants can also not be of polynomial tree-size. ◀

In what follows, we only consider the dag-size of a formula. So we say that propositional logic has *polysize Craig interpolants* if there is a polynomial function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for any φ, ψ with $\varphi \models \psi$ there exists a Craig interpolant χ for φ, ψ with $|\chi| \leq f(|\varphi|, |\psi|)$. We say that propositional logic has *polysize uniform interpolants* if there is a polynomial function $f : \mathbb{N} \to \mathbb{N}$ such that for any φ and ordering $\mathbf{p} = p_1, \ldots, p_n$ of any subset of $sig(\varphi)$, there exists a propositional formula χ equivalent to $\exists\mathbf{p}.\varphi$ with $|\chi| \leq f(|\varphi|)$, where $\exists\mathbf{p}.\varphi$ stands for $\exists p_1 \cdots \exists p_n.\varphi$.

We start with uniform interpolants and show the following equivalence.

▶ **Theorem 41.** *Propositional logic has polysize uniform interpolants iff* $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$.

Before proving Theorem 41, we explain the notation and concepts used in its formulation. We hope to convince a reader not familiar with computational complexity, in particular circuit complexity, that it is essentially a complexity theoretic reformulation of the definition of uniform interpolants. Note that the inclusion $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ (stating that every problem in the complexity class **NP** is also in the complexity class $\mathbf{P}_{/\mathbf{poly}}$) is a long-standing open problem in complexity theory. We comment on the status of this problem below, once the notation is introduced.

We introduce the relevant complexity classes, in particular those defined using computations with Boolean circuits. We refer the reader to [2] for further details. For our purposes, we can identify a Boolean circuit with a propositional formula. Let $\mathbf{p} = p_1, \ldots, p_n$ be an ordering of the atoms in a propositional formula φ. Then φ computes the function $C_\varphi : \{0,1\}^n \to \{0,1\}$ defined by setting $C_\varphi(v(p_1), \ldots, v(p_n)) = v(\varphi)$, with v any model mapping the atoms in $\mathbf{p}$ to $\{0,1\}$. Call n the *arity* of φ.

As usual for decision problems in computational complexity, we assume these are encoded as membership problems for subsets S of $\{0,1\}^*$, the set of words of arbitrary length over the alphabet $\{0,1\}$. For the definition of the complexity classes **P**, **NP**, and **coNP** via Turing machines we refer the reader to [2]. To define the complexity of $S \subseteq \{0,1\}^*$ in terms of computations with Boolean circuits, we require families $(\varphi_n)_{n\in\mathbb{N}}$ of propositional formulas, where φ_n has arity n. Then $(\varphi_n)_{n\in\mathbb{N}}$ *decides* S if for all $n \in \mathbb{N}$ and $t_1 \cdots t_n \in \{0,1\}^n$, $C_{\varphi_n}(t_1, \ldots, t_n) = 1$ iff $t_1 \cdots t_n \in S$. Note that the classes **P**, **NP**, and **coNP** are defined using a single Turing machine that accepts inputs of arbitrary length. In contrast, the complexity classes defined using Boolean circuits require families of circuits, one for each arity $n \in \mathbb{N}$.

Now S is in the complexity class $\mathbf{P}_{/\mathbf{poly}}$ if there exists a family $(\varphi_n)_{n\in\mathbb{N}}$ of formulas deciding S and a polynomial function $f : \mathbb{N} \to \mathbb{N}$ such that $|\varphi_n| \leq f(n)$ for all $n \in \mathbb{N}$. To define the class $\mathbf{NP}_{/\mathbf{poly}}$ we require quantified Boolean formulas of the form $\varphi = \exists q_1 \cdots \exists q_m.\psi$ with ψ a propositional formula. Any such quantified Boolean formula with additional non-quantified atoms $p_1, \ldots, p_n$ computes the function $C_\varphi : \{0,1\}^n \to \{0,1\}$ defined by setting $C_\varphi(v(p_1), \ldots, v(p_n)) = v(\varphi)$ with v any model mapping the atoms $p_1, \ldots, p_n$ to $\{0,1\}$. The *arity* of φ is n. Intuitively, to decide whether $t_1 \cdots t_n \in \{0,1\}^n$ is in a set S we guess a value $u_1 \cdots u_m \in \{0,1\}^m$ (a certificate) and check whether $v \models \psi$ for the model v with $v(p_1) = t_1, \ldots, v(p_n) = t_n, v(q_1) = u_1, v(q_m) = u_m$. We can now generalise the definitions for $\mathbf{P}_{/\mathbf{poly}}$ to $\mathbf{NP}_{/\mathbf{poly}}$ in the obvious way. A family $(\varphi_n)_{n\in\mathbb{N}}$ with $\varphi_n = \exists q_1 \cdots \exists q_{m_n}.\psi_n$ *decides* S if for all $n \in \mathbb{N}$ and $t_1 \cdots t_n \in \{0,1\}^n$, $C_{\varphi_n}(t_1, \ldots, t_n) = 1$ iff $t_1 \cdots t_n \in S$. S is in the complexity class $\mathbf{NP}_{/\mathbf{poly}}$ if there exists such a family $(\varphi_n)_{n\in\mathbb{N}}$ deciding S and a polynomial function $f : \mathbb{N} \to \mathbb{N}$ such that $|\psi_n| \leq f(n)$ for all $n \in \mathbb{N}$. $\mathbf{coNP}_{/\mathbf{poly}}$ is the complement of $\mathbf{NP}_{/\mathbf{poly}}$. The following inclusions hold [2].

▶ **Theorem 42.** $\mathbf{P} \subseteq \mathbf{P}_{/\mathbf{poly}}$, $\mathbf{NP} \subseteq \mathbf{NP}_{/\mathbf{poly}}$, *and* $\mathbf{coNP} \subseteq \mathbf{coNP}_{/\mathbf{poly}}$.

We return to Theorem 41. As mentioned above, the inclusion $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ is a long-standing open problem. It is regarded as very unlikely that this inclusion holds. In fact, by the Karp-Lipton Theorem, if $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$, then the polynomial hierarchy collapses at the second level which is regarded as unlikely [22, 21, 2]. Conversely, if we could prove $\mathbf{NP} \not\subseteq \mathbf{P}_{/\mathbf{poly}}$, then $\mathbf{NP} \neq \mathbf{P}$ would follow since $\mathbf{P} \subseteq \mathbf{P}_{/\mathbf{poly}}$. So we would have solved one of the most famous open questions in computer science.

We are now in a position to prove Theorem 41. Assume first that propositional logic has polysize uniform interpolants. Let $S \subseteq \{0,1\}^*$ be in $\mathbf{NP}$. By Theorem 42, S is in $\mathbf{NP}_{/\mathbf{poly}}$. Then, by definition, we find a family $(\exists \mathbf{q}_n.\psi_n)_{n\in\mathbb{N}}$ deciding S with $|\psi_n|$ bounded by a polynomial function. Since we assume that polysize uniform interpolants exist we find polysize χ_n logically equivalent to $\exists \mathbf{q}_n.\psi_n$. But then $(\chi_n)_{n\in\mathbb{N}}$ decides S and so S is in $\mathbf{P}_{/\mathbf{poly}}$.

Conversely, we show that $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ implies that uniform interpolants of polynomial size exist. Let $\exists P_n$ denote the set of formulas of the form $\varphi = \exists q_1 \cdots \exists q_m.\psi$ with ψ a propositional formula and $sig(\varphi) = \{p_1, \ldots, p_n\}$ the remaining atoms in φ distinct from $q_1, \ldots, q_m$. It suffices to show that $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ implies that formulas in $\exists P = \bigcup_{n\in\mathbb{N}} \exists P_n$ are equivalent to polysize propositional formulas. Let $w_\varphi \in \{0,1\}^*$ be an encoding of formulas $\varphi \in \exists P$. We regard every $w \in \{0,1\}^n$ as a model with $w(p_i) = w_i$ for $1 \leq i \leq n$. Consider the set

$$S_n = \{w_\varphi w' \in \{0,1\}^* \mid w' \models \varphi, w' \in \{0,1\}^n, \varphi \in \exists P_n\}$$

Then it follows from the definition that $S = \bigcup_{n\in\mathbb{N}} S_n$ is in $\mathbf{NP}$. From $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ we obtain a polynomial size sequence of propositional formulas $(\chi_k)_{k\in\mathbb{N}}$ with χ_k deciding $S \cap \{0,1\}^k$. To obtain the propositional formula equivalent to $\varphi = \exists q_1 \cdots \exists q_m.\psi \in \exists P_n$ simply let k be the length of w_φ and consider χ_{k+n}. Let the atoms of χ_{k+n} be $r_1, \ldots, r_k, p_1, \ldots, p_n$ with r_i

used for the i-th component of w_φ. Then the formula obtained from χ_{k+n} by replacing r_i by $\top$ if the i-th component of w_φ is 1 and by $\bot$ otherwise is as required. This finishes the proof of Theorem 41.

We next consider the size of Craig interpolants. Note that since uniform interpolants are also Craig interpolants, any lower bound on the size of Craig interpolants implies the same lower bound on the size of uniform interpolants. The following result is due to Mundici [34].

▶ **Theorem 43.** *If propositional logic has polysize Craig interpolants, then* $\mathbf{NP}_{/\mathbf{poly}} \cap \mathbf{coNP}_{/\mathbf{poly}} \subseteq \mathbf{P}_{/\mathbf{poly}}$*, and so also* $\mathbf{NP} \cap \mathbf{coNP} \subseteq \mathbf{P}_{/\mathbf{poly}}$*.*

Proof. Theorem 42 implies the "also"-part. The main part is shown by using the definitions in a straightforward way. Assume propositional logic has polysize Craig interpolants. Also assume that $S \subseteq \{0,1\}^*$ is in $\mathbf{NP}_{/\mathbf{poly}} \cap \mathbf{coNP}_{/\mathbf{poly}}$. Then, since S is in $\mathbf{NP}_{/\mathbf{poly}} \cap \mathbf{coNP}_{/\mathbf{poly}}$, we find families $(\exists \mathbf{q}_n.\varphi_n)_{n\in\mathbb{N}}$ and $(\exists \mathbf{q}'_n.\varphi'_n)_{n\in\mathbb{N}}$ of polysize deciding S and $\{0,1\}^* \setminus S$, respectively. Then, by definition, $\exists \mathbf{q}_n.\varphi_n \models \neg\exists \mathbf{q}'_n.\varphi'_n$. Hence $\varphi_n \models \neg\varphi'_n$ and so we find polysize Craig interpolants χ_n for $\varphi_n, \neg\varphi'_n$. But then $(\chi_n)_{n\in\mathbb{N}}$ decides S and so S is in $\mathbf{P}_{/\mathbf{poly}}$. ◀

The question of whether the inclusion $\mathbf{NP} \cap \mathbf{coNP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ holds is open, similar to the stronger inclusion $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$. Again, it is regarded very unlikely that the inclusion holds. However, if we could prove $\mathbf{NP} \cap \mathbf{coNP} \not\subseteq \mathbf{P}_{/\mathbf{poly}}$, then again $\mathbf{NP} \neq \mathbf{P}$ would follow.

Further complexity-theoretic consequences of the assumption that propositional logic has polysize Craig interpolants are shown in [43]. For instance, it would follow that $\mathbf{UP} \subseteq \mathbf{P}_{/\mathbf{poly}}$, where $\mathbf{UP}$ is the class of problems in $\mathbf{NP}$ accepted by non-deterministic Turing machines with at most one computation path on every input.

Interestingly, much more is known about the size of interpolants in the monotone fragment of propositional logic where only the connectives $\wedge$, $\vee$, $\top$, and $\bot$ are used. This fragment of propositional logic also has the Craig interpolation property, but now monotone formulas φ, ψ are known for which Craig interpolants in the monotone fragment are of exponential size. We refer the reader to Chapter 6 for a discussion of the monotone fragment and also of *feasible interpolation* where one does not measure the size of interpolants in the size of the input formulas but in the length of the proof of the entailment between them.

Acknowledgments

The authors thank Jean Christoph Jung, Amirhossein Akbar Tabatabai as well as Sam van Gool and his seminar students for their valuable comments and suggestions. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495.

References

1 Wilhelm Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Math. Ann.*, 110:390–413, 1935. doi:10.1007/BF01448035.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

3 Matthias Baaz, Uwe Egly, and Alexander Leitsch. Normal form transformations. In Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 1, chapter 5, pages 273–333. Elsevier, 2001. doi:10.1016/B978-044450813-3/50007-2.

4 Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 1, chapter 2, pages 19–99. Elsevier, 2001. doi:10.1016/B978-044450813-3/50004-7.

5 Evert Willem Beth. *The Foundations of Mathematics. A Study in the Philosophy of Science*. North-Holland, 1959.

6 Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, Univ. Chicago, Dept. of Math., 1937.

7 Régis Blanc, Ashutosh Gupta, Laura Kovács, and Bernhard Kragl. Tree interpolation in vampire. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *LPAR-19*, volume 8312 of *LNCS*, pages 173–181. Springer, 2013. doi:10.1007/978-3-642-45221-5_13.

8 Maria Paola Bonacina and Moa Johansson. Interpolation systems for ground proofs in automated deduction: a survey. *J. Autom. Reasoning*, 54(4):353–390, 2015. doi:10.1007/s10817-015-9325-5.

9 Frank Markham Brown and Sergiu Rudeanu. A functional approach to the theory of prime implicants. *Publications de l'Institut Mathématique*, 40(54):23–32, 1986.

10 Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Automated Theorem Proving*. Academic Press, 1973.

11 William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. Symb. Log.*, 22(3):269–285, 1957. doi:10.2307/2963594.

12 Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors. *Handb. of Tableau Methods*. Kluwer Academic Publishers, 1999. doi:10.1007/978-94-017-1754-0.

13 Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 2nd edition, 1996. doi:10.1007/978-1-4612-2360-3.

14 Melvin Fitting. Introduction. In Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors, *Handb. of Tableau Methods*, pages 1–43. Kluwer Academic Publishers, 1999. doi:10.1007/978-94-017-1754-0_1.

15 Dov Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second-order predicate logic. In *KR'92*, pages 425–435. Morgan Kaufmann, 1992.

16 Reiner Hähnle. Tableaux and related methods. In Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 1, chapter 3, pages 101–178. Elsevier, 2001. doi:10.1016/b978-044450813-3/50005-9.

17 Matthias Heizmann, Jochen Hoenicke, and Andreas Podelski. Nested interpolants. In Manuel V. Hermenegildo and Jens Palsberg, editors, *POPL 2010*, pages 471–482. ACM, 2010. doi:10.1145/1706299.1706353.

18 Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Kenneth L. McMillan. Abstractions from proofs. In Neil D. Jones and Xavier Leroy, editors, *POPL 2004*, pages 232–244. ACM, 2004. doi:10.1145/964001.964021.

19 Jaakko Hintikka. Form and content in quantification theory. *Acta philosophica Fennica*, 8:7–55, 1955.

20 Guoxiang Huang. Constructing Craig interpolation formulas. In Ding-Zhu Du and Ming Li, editors, *COCOON '95*, volume 959 of *LNCS*, pages 181–190. Springer, 1995. doi:10.1007/BFB0030832.

21 Richard Karp. Turing machines that take advice. *Enseign. Math.*, 28:191–209, 1982.

22 Richard M Karp and Richard J Lipton. Some connections between nonuniform and uniform complexity classes. In Raymond E. Miller, Seymour Ginsburg, Walter A. Burkhard, and Richard J. Lipton, editors, *STOC'80*, pages 302–309. ACM, 1980. doi:10.1145/800141.804678.

23 Hans Kleine Büning and Theodor Lettmann. *Propositional Logic – Deduction and Algorithms*. Cambridge University Press, 1999.

24 Donald E. Knuth. *The Art of Computer Programming*, volume 4, fascicle 6, Satisfiability. Addison-Wesley, 2015.

25 Boris Konev, Carsten Lutz, Denis K. Ponomaryov, and Frank Wolter. Decomposing description logic ontologies. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *KR 2010*. AAAI Press, 2010. URL: http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1356.

26 George Kourousias and David Makinson. Parallel interpolation, splitting, and relevance in belief change. *J. Symb. Log.*, 72(3):994–1002, 2007. doi:10.2178/JSL/1191333851.

27 Jan Krajícek. Lower bounds to the size of constant-depth propositional proofs. *J. Symb. Log.*, 59(1):73–86, 1994. doi:10.2307/2275250.

28 Jan Krajícek. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. Symb. Log.*, 62(2):457–486, 1997. doi:10.2307/2275541.

29 Shoji Maehara. On the interpolation theorem of Craig (Japanese). *Sugaku*, 12:235–237, 1960.

30 Kenneth L. McMillan. Interpolation and SAT-based model checking. In Warren A. Hunt Jr. and Fabio Somenzi, editors, *CAV 2023*, volume 2725 of *LNCS*, pages 1–13. Springer, 2003. doi:10.1007/978-3-540-45069-6_1.

31 Kenneth L. McMillan. Applications of Craig interpolants in model checking. In Nicolas Halbwachs and Lenore D. Zuck, editors, *TACAS 2005*, volume 3440 of *LNCS*, pages 1–12. Springer, 2005. doi:10.1007/978-3-540-31980-1_1.

32 Kenneth L. McMillan. An interpolating theorem prover. *Theor. Comput. Sci.*, 345(1):101–121, 2005. doi:10.1016/J.TCS.2005.07.003.

33 Kenneth L. McMillan. Lazy abstraction with interpolants. In Thomas Ball and Robert B. Jones, editors, *CAV 2006*, volume 4144 of *LNCS*, pages 123–136. Springer, 2006. doi:10.1007/11817963_14.

34 Daniele Mundici. Tautologies with a unique Craig interpolant, uniform vs. nonuniform complexity. *Ann. Pure Appl. Log.*, 27(3):265–273, 1984. doi:10.1016/0168-0072(84)90029-0.

35 Rohit Parikh. Beth definability, interpolation and language splitting. *Synth.*, 179(2):211–221, 2011. doi:10.1007/S11229-010-9778-3.

36 Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symb. Log.*, 62(3):981–998, 1997. doi:10.2307/2275583.

37 Willard Van Orman Quine. A way to simplify truth functions. *American Mathematical Monthly*, 59(9):627–631, 1955.

38 J. A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23–41, 1965. doi:10.1145/321250.321253.

39 Benjamin Rossman. Formulas versus circuits for small distance connectivity. *SIAM J. Comput.*, 47(5):1986–2028, 2018. doi:10.1137/15M1027310.

40 Philipp Rümmer, Hossein Hojjat, and Viktor Kuncak. Disjunctive interpolants for horn-clause verification. In Natasha Sharygina and Helmut Veith, editors, *CAV 2013*, volume 8044 of *LNCS*, pages 347–363. Springer, 2013. doi:10.1007/978-3-642-39799-8_24.

41 Philipp Rümmer, Hossein Hojjat, and Viktor Kuncak. On recursion-free Horn clauses and Craig interpolation. *Formal Methods Syst. Des.*, 47(1):1–25, 2015. doi:10.1007/S10703-014-0219-7.

42 Edward W. Samson and Rolf K. Mueller. Circuit minimization: sum to one process for irredundant sums. Technical Report AFCRC-TR-55-118, Air Force Cambridge Research Center, August 1955.

43 Uwe Schöning and Jacobo Torán. A note on the size of Craig interpolants. In Thomas Schwentick, Denis Thérien, and Heribert Vollmer, editors, *Circuits, Logic, and Games – Dagstuhl Seminar*, volume 6451 of *DagSemProc*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2006. doi:10.4230/DagSemProc.06451.3.

44 Raymond M. Smullyan. *First-Order Logic*. Springer, 1968. Also republished with corrections by Dover publications, 1995.

45 Gaisi Takeuti. *Proof Theory*. North-Holland, second edition, 1987.

46 Grigori Samuilovitsch Tseitin. On the complexity of derivation in propositional calculus. In *Structures in Constructive Mathematics and Mathematical Logic*, pages 115–125. Steklov Mathematical Institute, 1968.

47 Greta Yorsh and Madanlal Musuvathi. A combination method for generating interpolants. In Robert Nieuwenhuis, editor, *CADE 20*, volume 3632 of *LNCS (LNAI)*, pages 353–368. Springer, 2005. doi:10.1007/11532231_26.

Chapter 2

Interpolation in First-Order Logic

Balder ten Cate
University of Amsterdam, The Netherlands

Jesse Comer
University of Pennsylvania, USA

Abstract

In this chapter we give a basic overview of known results regarding Craig interpolation for first-order logic as well as for fragments of first-order logic. Our aim is to provide an entry point into the literature on interpolation theorems for first-order logic and fragments of first-order logic, and their applications. In particular, we cover a range of known refinements of the Craig interpolation theorem, we discuss several important applications of interpolation in logic and computer science, we review known results about interpolation for important syntactic fragments of first-order logic, and we discuss the problem of computing interpolants.

Contents

1 Introduction 36

2 Interpolation Theorems for First-Order Logic 36
2.1 Craig Interpolation 37
2.2 Variants of the Interpolation Theorem 37
2.3 Negative Results 43

3 Applications of Interpolation for First-Order Logic 44
3.1 $\exists\mathrm{SO} \cap \forall\mathrm{SO} = \mathrm{FO}$ 44
3.2 Preservation Theorems 45
3.3 The Beth Definability Theorem 46
3.4 Combining Knowledge Bases 50
3.5 Formal Verification 50

4 Interpolation for Decidable Fragments of First-Order Logic 52
4.1 Decidable Fragments 52
4.2 Repairing Interpolation 53
4.3 Alternative Approaches 54



© Balder ten Cate and Jesse Comer;
licensed under Creative Commons License CC BY 4.0

In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 2, pages 35–67. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.b

5 Interpolation Relative to a Theory 55
6 Proof Methods and Algorithmic Aspects 56
6.1 A Proof of Craig interpolation via Semantic Tableaux 57
6.2 Computational Complexity and Size Bounds 61
Acknowledgments 63
References 63

1 Introduction

In this chapter we give a basic overview of known results regarding Craig interpolation for first-order logic and many of its well-known fragments. Our aim is to provide an easy entry point into the literature. We focus on providing a global picture, intuition through examples, and pointers to further chapters in this book as well as other related literature. Other useful articles on this topic include [30], which provides a survey of the history and refinements of the theorem, [15], which outlines important perspectives on the theorem and its significance, and [62], describing the significance of the theorem in the study of abstract logics.

In Section 2, we recall the Craig interpolation theorem for first-order logic (FO), as well as its variants. In Section 3, we briefly survey some applications of the theorem, with references to other chapters of this book for additional details. In Section 4, we describe what is known about the Craig interpolation property among fragments of FO. Section 5 discusses interpolation in the context of background theories. Finally, Section 6 discusses how the Craig interpolation property can be proven, and how proof theory permits the effective construction of interpolants.

2 Interpolation Theorems for First-Order Logic

In this section, we review several prominent interpolation theorems that have been established for FO. We first briefly recall the syntax of first-order logic in order to fix our notation. A *signature* is a collection of relation, constant, and function symbols, where each relation symbol R and each function symbol f has a specified arity (denoted $\mathrm{arity}(R)$ or $\mathrm{arity}(f)$). We assume first-order formulas are generated by the following grammar:

$$\phi ::= R(t_1, \ldots, t_n) \mid t_1 = t_2 \mid \top \mid \phi \wedge \psi \mid \neg\phi \mid \exists x \phi$$

where R is an n-ary relation symbol, and $t_1, \ldots, t_n$ are terms built up from variables and constant symbols using function symbols. We treat disjunction, implication, and universal quantification as shorthand notations. We write $\mathrm{sig}(\phi)$ to denote the set of relation symbols, constants, and function symbols occurring in ϕ. We write $\mathrm{fvar}(\phi)$ to denote the set of free variables of ϕ. A *sentence* is a formula without free variables.

All results stated in this section hold for first-order logic both with and without equality. In the case without equality, it is important for the interpolation theorems that $\top$ is explicitly

included in the syntax. In the case with equality, $\top$ could be treated as a shorthand for $\forall x(x = x)$. We defer discussion of techniques used to prove the interpolation theorems to Section 6.

2.1 Craig Interpolation

We begin with the Craig interpolation theorem (1957 [25, 24]).

▶ **Definition 2.1** (Craig interpolant). *Let ϕ, ψ be* FO *formulas such that $\models \phi \to \psi$. A* Craig interpolant *for $\models \phi \to \psi$ is an* FO*-formula ϑ such that the following hold:*

1. $\models \phi \to \vartheta$,
2. $\models \vartheta \to \psi$,
3. $\mathrm{sig}(\vartheta) \subseteq \mathrm{sig}(\phi) \cap \mathrm{sig}(\psi)$,
4. $\mathrm{fvar}(\vartheta) \subseteq \mathrm{fvar}(\phi) \cap \mathrm{fvar}(\psi)$.

▶ **Theorem 2.2** (Craig interpolation [25, 24]). *Every valid* FO *implication has a Craig interpolant.*

▶ **Example 2.3.** Consider the formulas $\phi = \exists x Cat(x) \wedge \forall x(Cat(x) \to (Big(x) \wedge Green(x)))$ and $\psi = \exists x(Big(x) \wedge (Cat(x) \vee Dog(x)))$. Then

$$\models \phi \to \psi$$

In this case $\mathrm{sig}(\phi) = \{Cat, Big, Green\}$ and $\mathrm{sig}(\psi) = \{Cat, Dog, Big\}$. Therefore, a Craig interpolant for this valid entailment may only contain the relation symbols Cat and Big. Examples of Craig interpolants for this implication include:

- $\vartheta_1 = \exists x(Big(x) \wedge Cat(x))$, and
- $\vartheta_2 = \exists x(Cat(x)) \wedge \forall x(Cat(x) \to Big(x))$

This also shows that Craig interpolants are not unique in general. ⌟

2.2 Variants of the Interpolation Theorem

The *Lyndon interpolation theorem* refines the Craig interpolation theorem by distinguishing positive occurrences and negative occurrences of relation symbols. We write $\mathrm{relsig}^+(\phi)$ and $\mathrm{relsig}^-(\phi)$ to denote the set of relation symbols occurring *positively*, respectively, *negatively*, in ϕ. Here, an occurrence of a relation symbol is positive if it is in the scope of an even number of negation symbols, and negative if it is in the scope of an odd number of negation symbols; we refer to these terms as the *polarity* of the occurrence.

▶ **Definition 2.4** (Lyndon interpolant). *A* Lyndon interpolant *for $\models \phi \to \psi$ is a Craig interpolant ϑ such that, in addition, the following hold:*

1. $\mathrm{relsig}^+(\vartheta) \subseteq \mathrm{relsig}^+(\phi) \cap \mathrm{relsig}^+(\psi)$,
2. $\mathrm{relsig}^-(\vartheta) \subseteq \mathrm{relsig}^-(\phi) \cap \mathrm{relsig}^-(\psi)$,

▶ **Theorem 2.5** (Lyndon interpolation [42]). *Every valid* FO *implication has a Lyndon interpolant.*

Thus, the Lyndon interpolation theorem refines the Craig interpolation theorem by interpolating jointly over positive and negative occurrences of relation symbols. It is worth noting that, for *constants and function symbols*, no distinction is made between positive and negative occurrences.

▶ **Example 2.6.** Let us revisit Example 2.3. In this case, $\text{relsig}^+(\phi) = \{Cat, Big, Green\}$, $\text{relsig}^-(\phi) = \{Cat\}$, $\text{relsig}^+(\psi) = \{Big, Cat, Dog\}$ and $\text{relsig}^-(\psi) = \emptyset$. It follows that a Lyndon interpolant for $\models \phi \to \psi$ may only contain positive occurrences of Cat and Big and no negative occurrences. In Example 2.3, we listed two Craig interpolants. The first of these is also a Lyndon interpolant, while the second is not. ⌟

▶ **Example 2.7.** This example illustrates why relation symbols are treated differently from constants and function symbols in the Lyndon interpolation theorem. Let $\phi = P(c)$ and let $\psi = \neg\exists x(x = c \wedge \neg P(x))$. Clearly $\models \phi \to \psi$ (indeed, ϕ and ψ are logically equivalent). The constant symbol c occurs only positively (i.e., under an even number of negations) in ϕ and occurs only negatively (i.e., under an odd number of negations) in ψ. It is easy to see that there does not exist a Craig interpolant for $\models \phi \to \psi$ that does not contain an occurrence of c. In other words, if we were to require that the interpolant contains only constants that occur in ϕ and ψ with the same polarity, then such an interpolation does not in general exist for a valid FO implication. The same example can be easily adjusted to use a function symbol instead of a constant. ⌟

Several further refinements were subsequently obtained. For instance, Feferman [28] and Stern [61] proved interpolation theorems in the setting of first-order formulas over a many-sorted signature, which also interpolates over sorts. We omit the details here since it would require introducing many-sorted signatures, and also because Feferman's and Stern's interpolation theorems are subsumed by an even more powerful interpolation theorem that can be stated without reference to many-sorted signatures – namely, the Otto interpolation theorem. For a set of unary predicates $\mathcal{U}$, we say that an FO-formula ϕ is *$\mathcal{U}$-relativized* if every quantifier in ϕ is relativized, i.e., is of the form $\exists x(U(x) \wedge \cdots)$ or $\forall x(U(x) \to \cdots)$, for some $U \in \mathcal{U}$.

▶ **Theorem 2.8** (Otto interpolation [46]). *For every pair of* FO *formulas ϕ, ψ such that $\models \phi \to \psi$, and for all sets $\mathcal{U}$ of unary predicates, if ϕ and ψ are both $\mathcal{U}$-relativized, then there is a Lyndon interpolant ϑ that is also $\mathcal{U}$-relativized.*

Observe that a formula is $\emptyset$-relativized if and only if it is quantifier-free. Therefore, as a consequence of Otto interpolation, every valid quantifier-free entailment has a quantifier-free Lyndon interpolant. This also implies Lyndon interpolation for propositional logic, although it is a very inefficient route to obtain the latter.

▶ **Example 2.9.** The following implication is valid:

$$\models (\exists x(Px \wedge Rxx) \wedge \forall x(Px \to Qx)) \to (\exists x(Qx \wedge \neg Tx) \vee \exists x(Tx \wedge Rxx)).$$

Both formulas are $\mathcal{U}$-relativized with $\mathcal{U} = \{P, Q, T\}$. By the Otto interpolation theorem, there is a $\mathcal{U}$-relativized interpolant. This interpolant may furthermore only contain positive occurrences of Q and positive occurrences of R (and no occurrences of P or T whatsoever). An example of such an interpolant is $\exists x(Qx \wedge Rxx)$. ⌟

▶ **Example 2.10.** This example shows that Theorem 2.8 cannot be strengthened to interpolate on the relativizers (i.e., to require that all relativizers in the interpolant must occur as relativizers in both ϕ and ψ). Consider the valid FO implication

$$\models \forall x(Px \to \neg Qx) \to \forall x(Qx \to \neg Px).$$

Both formulas here are $\mathcal{U}$-relativized sentences, for $\mathcal{U} = \{P, Q\}$. However, they do not have any relativizers in common. If all relativizers in some interpolant occurred as relativizers in both ϕ and ψ, then that interpolant would be an $\emptyset$-relativized sentence. As we have already observed, an FO formula is $\emptyset$-relativized if and only if it is quantifier-free. Therefore, the interpolant would have to be a quantifier-free sentence. The only candidates, up to logical equivalence, are $\top$ and $\bot$, and it is easy to see that neither is an interpolant. ⌟

The last interpolation theorem we mention here is the *access interpolation theorem* [14, 12], a further refinement of Otto interpolation. It is discussed in detail also in Chapter 14. The main idea of the access interpolation is that we want to consider not only which relation symbols occur in a formula, but also the "way" in which the relations in question are "accessed". Before we give a formal definition of an access method, we illustrate the concept with an example.

▶ **Example 2.11.** Consider the infinite ternary *multiplication* relation $R_\times$, consisting of all triples (x, y, z) of real numbers such that $z = x \cdot y$. A pocket calculator, intuitively, provides certain forms of access to this relation. Given specific values for x and y, the calculator can be used to produce the set $\{z \mid R_\times(x, y, z)\}$ (by using the "$\times$" button – note that this set is always a singleton). Similarly, given specific values for x and z, the calculator can produce the set of all corresponding values for y (by using the "$\div$" button — note that this set is either a singleton or is empty). On the other hand, the pocket calculator does not provide a way to compute, for a given value z, the set of all matching pairs (x, y). Indeed, this set would be infinite. To be precise, the pocket calculator provides three forms of access to $R_\times$: (i) find triples $(x, y, z) \in R_\times$ that have specified values for x and y, (ii) find triples $(x, y, z) \in R_\times$ that have specified values for x and z, and (iii) find triples $(x, y, z) \in R_\times$ that have specified values for y and z. In each case, the "specified values" must be provided by the user as input. Of course, in this particular example, (ii) and (iii) are, in some sense, equivalent. ⌟

The access interpolation theorem, intuitively, refines Craig interpolation by controlling the way in which relations may be accessed by the interpolant.

▶ **Definition 2.12.** *An* access method *is a pair* (R, X) *where* R *is a* n*-ary relation symbol and* $X \subseteq \{1, \ldots, n\}$.

Intuitively, an access method (also known in the logic programming literature as a *mode*) (R, X) signifies a way to access the relation R, namely, by providing specific values for the arguments in X as input, and retrieving the set of all matching tuples. The access method $(R, \emptyset)$ does not require specifying any values as input and therefore allows us to retrieve the full content of the relation R. An access method (R, X) with $X = \{i_1, \ldots, i_m\}$ a non-empty set, on the other hand, only allows us to retrieve, for a specific tuple of values $a_1, \ldots, a_m$, the set of tuples in R that have specified value a_k in position i_k for $k \leq m$. In particular, the access method $(R, \{1, \ldots, \text{arity}(R)\})$ only allows testing if a specific tuple belongs to the relation R. There is a natural partial order on access methods: $(R, X) \preceq (R, Y)$ if $X \subseteq Y$. Intuitively, in this case, the former access method is "at least as powerful as" (or, "can be used to implement") the latter access method. For any set S of access methods, we denote by $S^{\uparrow}$ the upwards closure of S w.r.t. the partial order $\preceq$.

▶ **Example 2.13.** Consider the FO sentence $\forall x(Px \to Qx)$. We can determine whether this sentence holds in a structure A using only the access methods $(P, \emptyset)$ (which allows us to retrieve the entire content of P) and $(Q, \{1\})$ (which allows us to test for any specific value whether it belongs to Q). Namely, we first retrieve the entire set P^A and then we test, for each $a \in P^A$, whether or not $a \in Q^A$. Similarly, we can determine whether the sentence holds in a structure A using access methods $(P, \emptyset)$ and $(Q, \emptyset)$ by retrieving the entire sets P^A and Q^A and testing if the former is a subset of the latter. On the other hand, having only the access methods $(P, \{1\})$ and $(Q, \{1\})$ is not enough, since we are not given any information about the domain of A. ⌟

The above example suggests a notion of answerability using access methods. We say that a formula $\phi(\mathbf{x})$ is *answerable for membership* (or simply, *answerable*) using a set of access methods S if given a structure A and a tuple $\mathbf{a} = a_1, \ldots, a_n$, we can effectively test whether $A \models \phi(\mathbf{a})$ by a procedure that only accesses A using the access methods in S. We will refrain here from giving a formal definition, but one can be found in Chapter 14 based on the notion of an *RA-plan* (cf. also [12, Section 3.1]). Note: there are FO formulas that are not answerable using *any* set of access methods. Examples include $\forall x P(x)$ and $\exists x \neg P(x)$. Indeed, an FO *sentence* is answerable by a set of access methods only if it is *domain-independent* (i.e., its truth value in a structure depends only on the denotation of the relation symbols and not on the underlying domain of the structure, or, in other words, it "ignores domain elements that do not participate in any relation").[1]

[1] This can be derived from Corollary 2.16 below, by choosing S to be the set of all access methods.

Whether or not an FO formula is answerable using a given set of access methods has a lot to do with the syntactic shape of the formula in question. Given an FO formula ϕ, we can associate to ϕ a set of access methods as follows:

$$
\begin{array}{lcl}
\mathrm{BindPatt}(\top) = \mathrm{BindPatt}(x = y) & = & \emptyset, \\
\mathrm{BindPatt}((R(x_1, \ldots, x_n)) & = & \{(R, \{1, \ldots, n\})\}, \\
\mathrm{BindPatt}(\phi \wedge \psi) & = & \mathrm{BindPatt}(\phi) \cup \mathrm{BindPatt}(\psi), \\
\mathrm{BindPatt}(\phi \vee \psi) & = & \mathrm{BindPatt}(\phi) \cup \mathrm{BindPatt}(\psi), \\
\mathrm{BindPatt}(\neg\phi) & = & \mathrm{BindPatt}(\phi), \\
\mathrm{BindPatt}(\exists \mathbf{x}(R(y_1, \ldots, y_n) \wedge \phi) & = & \mathrm{BindPatt}(\phi) \cup \{(R, X)\} \\
 & & \quad \text{where } X = \{i \leq n \mid y_i \notin \mathbf{x}\}, \\
\mathrm{BindPatt}(\forall \mathbf{x}(R(y_1, \ldots, y_n) \to \phi) & = & \mathrm{BindPatt}(\phi) \cup \{(R, X)\} \\
 & & \quad \text{where } X = \{i \leq n \mid y_i \notin \mathbf{x}\}, \\
\mathrm{BindPatt}(\phi) & = & \text{undefined for all } \phi \text{ not of the above forms}
\end{array}
$$

In the quantifier clauses above, $\mathbf{x}$ denotes a tuple of variables (i.e., we allow for quantifying multiple variables at once). The binding atom $R(y_1, \ldots, y_n)$ is not required to contain all free variables of ϕ (i.e., it is not required to be a "guard" in the sense of the guarded fragment [2]), but it must contain every variable in $\mathbf{x}$.

We will say that a FO formula is *S-bounded* (for a set S of access methods) if $\mathrm{BindPatt}(\phi)$ is well-defined and $\mathrm{BindPatt}(\phi) \subseteq S^{\uparrow}$.[2] Every S-bounded FO formula is answerable, in the sense described above, using the access methods S. Indeed, Corollary 2.16 below will link S-boundedness in a precise way to answerability using the access modes in S.

▶ **Example 2.14.** To verify the truth of $\phi = \forall xy(R(x, y) \to S(x, y))$ in a structure A using access methods, we need full access to the relation R^A, but we only need limited access to S^A which allows us to test membership of specific tuples. This is reflected by the fact that $\mathrm{BindPatt}(\phi) = \{(R, \emptyset), (S, \{1, 2\})\}$. ⌟

For simplicity, the following results are stated only for relational signatures (i.e., without constant symbols and function symbols). However, the results also hold true (suitably adapted) for signatures that include constant symbols.

▶ **Theorem 2.15** (Access interpolation [12])**.** *Fix any relational signature and set S of access methods. For every pair of S-bounded* FO *formulas ϕ, ψ such that $\models \phi \to \psi$ there is an S-bounded Lyndon interpolant ϑ.*[3]

---

[2] It is not difficult to see that a FO formula ϕ with $\mathrm{BindPatt}(\phi) \subseteq S^{\uparrow}$ can always be rewritten into an equivalent formula ϕ' with $\mathrm{BindPatt}(\phi') \subseteq S$. However, this rewriting may require the use of the equality symbol (for instance, if $S = \{(R, \emptyset)\}$, then it may be necessary to rewrite $R(x, y)$ to $\exists uv(R(u, v) \wedge u = x \wedge v = y)$). To ensure that all theorems stated in this section hold also in the equality-free case, it is necessary to phrase the definition of *S-bounded* in terms of $S^{\uparrow}$ instead of S.

[3] The original statement in [12] is slightly stronger, we simplified it here.

As a corollary, we obtained a semantic characterization of S-boundedness in the form of a preservation theorem. In order to state this characterization, fix a set S of access methods. For a given structure A and tuple $\mathbf{a}$, let us denote by $AccPart_S(A, \mathbf{a})$ the "part of A that is accessible from $\mathbf{a}$ using S", i.e., the smallest set $D \subseteq Dom(A)$ that contains $\mathbf{a}$ and is closed under the following rule:

$$\text{If } (a_1, \dots a_n) \in R^A, (R, X) \in S \text{ and } \{a_i \mid i \in X\} \subseteq D, \text{ then } \{a_1, \dots, a_n\} \subseteq D$$

A FO-formula $\phi(\mathbf{x})$ is *access-determined* by S if for all structures A and tuples $\mathbf{a}$, $A \models \phi(\mathbf{a})$ iff $A|D \models \phi(\mathbf{a})$, where $D = AccPart_S(A, \mathbf{a})$ and where $A|D$ is the substructure of A induced by D. Access determinacy can be viewed as a refined form of *invariance for generated submodels* (cf. [3]) parameterized by a set of access methods. Access-determinacy is also a necessary precondition for answerability using a given set of access methods.

▶ **Corollary 2.16.** *Let ϕ be any* FO *formula over a relational signature and let S be any set of access methods.*

1. *If ϕ is S-bounded then ϕ is access-determined by S.*
2. *If ϕ is access-determined by S then ϕ is equivalent to an S-bounded* FO*-formula.*

Proof sketch. The first item can be proved by a formula induction. The second item follows from Theorem 2.15 as follows: assume that $\phi(x_1, \dots, x_m)$ is access-determined by S. Let D be a fresh unary predicate. We can construct an $S \cup \{(D, \emptyset)\}$-bounded FO-formula $\chi_D(x_1, \dots, x_m)$ that, intuitively speaking, expresses that the denotation of D includes the accessible part of the structure (with respect to S and $x_1, \dots, x_m$). It follows by the definition of access-determinacy that $\models (\chi(D) \wedge \phi^D) \rightarrow (\chi(D') \rightarrow \phi^{D'})$ where ϕ^D is the syntactic relativization of ϕ by D (i.e., the formula obtained from ϕ by relativizing every quantifier to D). It can then be shown that an interpolant for this valid entailment, as produced by Theorem 2.15, is an S-bounded FO-formula that is equivalent to ϕ. A detailed version of this argument can be found in Section 6 of Chapter 14. ◀

The Otto interpolation theorem (for relational signatures) follows from the access interpolation theorem, because an FO-formula ϕ is $\mathcal{U}$-relativized if and only if ϕ is S-bounded for $S = \{(U, \emptyset) \mid U \in \mathcal{U}\} \cup \{(R, \{1, \dots, \text{arity}(R)\}) \mid R \in \text{sig}(\phi)\}$.

One may wonder if the above interpolation theorem could be further strengthened to

$$\text{BindPatt}(\vartheta) \subseteq \text{BindPatt}(\phi) \cap \text{BindPatt}(\psi).$$

The answer is negative, as follows from Example 2.10.

So far, we have discussed Craig interpolation and several of its refinements. A number of other closely related notions exist that can be viewed as weak forms of Craig interpolation, including Δ-*interpolation* and *Beth definability*, discussed in Sections 3.1 and 3.3, respectively.

2.3 Negative Results

We now present two negative results. The first pertains to *finite model theory*, i.e., the study of first-order logic when restricted to finite structures.

▶ **Theorem 2.17** (Failure of Craig interpolation in the finite [35]). *There is an* FO *implication (over a relational signature) that is valid in the finite, i.e.,*

$$\models_{fin} \phi \to \psi,$$

and for which there is no FO *formula* ϑ *with* $sig(\vartheta) \subseteq sig(\phi) \cap sig(\psi)$ *satisfying* $\models_{fin} \phi \to \vartheta$ *and* $\models_{fin} \vartheta \to \psi$.

Proof (sketch). Let ϕ be the FO sentence with $\mathrm{sig}(\phi) = \{R\}$ expressing that R is an equivalence relation and every equivalence class of R contains precisely two elements. Similarly, let ψ be the FO sentence with $\mathrm{sig}(\psi) = \{S\}$ expressing that S is an equivalence relation and that every equivalence class of S contains precisely two elements except for one equivalence class, which contains three elements. Every finite model of ϕ has a domain of even cardinality, while every finite model of ψ has a domain of odd cardinality. Therefore, $\models_{\mathrm{fin}} \phi \to \neg\psi$. A Craig interpolant must have an empty signature and must express (over finite structures) that the domain has even cardinality. It is well known (and can be shown using standard techniques such as Ehrenfeucht-Fraïssé games), that no such formula exists. ◀

Example 3.8 also shows that several important consequences of Craig interpolation fail in the finite as well. The second negative result is concerned with *uniform interpolation*, a topic discussed in detail in Chapter 9.

▶ **Definition 2.18** (Uniform interpolant). *An* FO*-formula* ϑ *is a* uniform interpolant *for an* FO*-formula* ϕ *w.r.t. a signature* $\sigma \subseteq \mathrm{sig}(\phi)$ *if* $\models \phi \to \vartheta$, $\mathrm{sig}(\vartheta) \subseteq \sigma$, *and for all* FO*-formulas* ψ *satisfying* $\models \phi \to \psi$ *and* $\mathrm{sig}(\phi) \cap \mathrm{sig}(\psi) \subseteq \sigma$, *it holds that* $\models \vartheta \to \psi$.

In other words, a uniform interpolant for an FO-formula ϕ is a formula ϑ which is a Craig interpolant for *all* valid implications $\models \phi \to \psi$, where ϑ depends only on ϕ and the signature σ. In the literature, the above notion of "uniform interpolant" is more commonly referred to as *uniform post-interpolant*. There is also a dual notion of *uniform pre-interpolant*. For simplicity, we only consider uniform post-interpolants here, and we will call them simply "uniform interpolant". Propositional logic, intuitionistic logic, and certain modal logics admit uniform interpolants. Uniform interpolation fails, however, for FO [36][4].

[4] An interesting fact of history is that, by William Craig's own account, as told during an event in Berkeley celebrating his 90th birthday, the Craig interpolation theorem was obtained as the result of a failed attempt to prove uniform interpolation for first-order logic ([15] and *Ph. Kolaitis, personal communication*). For more history of the Craig interpolation theorem, see also [26, 27].

▶ **Theorem 2.19** (Failure of uniform interpolation for FO, folklore)**.** *There is an* FO *formula* ϕ *and a signature* $\sigma \subseteq \mathrm{sig}(\phi)$ *for which there is no uniform interpolant.*

Proof. Let ϕ be the FO-sentence in one relation symbol R, expressing that R is a total linear order without end-points. Let σ furthermore be the empty signature. Finally, let ϕ_n (for $n > 1$) be the FO-sentence $\exists x_1 \dots x_n \bigwedge_{1 \leq i < j \leq n} x_i \neq x_j$, expressing the existence of n distinct objects. Clearly, $\models \phi \to \phi_n$ for all n. A uniform interpolant would have to be a FO-sentence ϑ that is implied by ϕ and that implies ϕ_n for all $n > 1$. Thus ϑ would be a sentence true in all and only infinite structures, which, by a simple compactness argument, cannot exist. ◀

3 Applications of Interpolation for First-Order Logic

In this section, we review a few ways in which interpolation theorems for FO can be applied.

3.1 $\exists$SO $\cap$ $\forall$SO = FO

The Craig interpolation theorem implies something interesting about the relationship between first-order logic (FO) and second-order logic (SO). Recall that second-order logic extends first-order logic with second-order variables (of arbitrary arity) and second-order quantifiers. Every second-order formula over a signature τ can be written in prenex form as

$$Q_1 X_1 \dots Q_n X_n . \phi$$

where $Q_1, \dots, Q_n \in \{\forall, \exists\}$, $X_1, \dots, X_n$ are second-order variables, and ϕ is a first-order formula over the signature $\tau \cup \{X_1, \dots, X_n\}$. We write $\exists$SO (resp. $\forall$SO) to denote *existential second-order logic* (resp. *universal second-order logic*), that is, the fragment of SO consisting of formulas in the above prenex form where $Q_1 = \dots = Q_n = \exists$ (resp. $Q_1 = \dots = Q_n = \forall$).

▶ **Example 3.1.** The following $\exists$SO sentence defines the class of (possibly infinite) directed graphs that are *3-colorable*:

$$\exists X_1 \exists X_2 \exists X_3 \big(\forall x (X_1(x) \vee X_2(x) \vee X_3(x)) \wedge \forall xy (R(x,y) \to \bigwedge_{i \leq 3} (X_i(x) \to \neg X_i(y))) \big)$$

Similarly, the following $\forall$SO sentence defines the class of (possibly infinite) *strongly connected* directed graphs:

$$\forall X \big(\exists y (X(y)) \wedge \forall yz (X(y) \wedge Ryz \to X(z)) \to \forall y (X(y)) \big)$$

As a final example, the following $\forall$SO sentence defines the class of *finite* structures:

$$\forall R \big((\forall x \exists y R(x,y) \wedge \forall xyz (R(x,y) \wedge R(y,z) \to R(x,z))) \to \exists x R(x,x) \big).$$

None of these classes is FO-definable. ⌟

▶ **Theorem 3.2** (Δ-interpolation). *Let K be a class of structures that is definable by a* ∃SO*-sentence as well as by a* ∀SO*-sentence. Then K is already definable by an* FO*-sentence.*

Proof. Let $\phi = \exists R_1 \dots R_n.\phi'$ be the ∃SO-sentence defining K, and let $\psi = \forall S_1 \dots S_m.\psi'$ be the ∀SO-sentence defining K. We may assume without loss of generality that $\{R_1, \dots, R_n\} \cap \{S_1, \dots, S_m\} = \emptyset$. From the fact that ϕ and ψ are equivalent, it follows that $\models \phi' \to \psi'$. Let ϑ be any Craig interpolant for this valid FO-implication. In particular, $\models \phi' \to \vartheta$ and $\models \vartheta \to \psi'$. Since $R_1 \dots R_n$ and $S_1 \dots S_m$ do not occur in ϑ, it follows that $\models \phi \to \vartheta$ and $\models \vartheta \to \psi$. Since ϕ and ψ are equivalent, it follows that ϑ is equivalent to ϕ and ψ as well. Hence, ϑ defines the class K. ◀

The same argument used here shows something stronger, namely: *whenever $K \subseteq K'$ with K an* ∃SO*-definable class of structures and K' a* ∀SO*-definable class of structures, then there is an* FO*-definable class of structures K'' with $K \subseteq K'' \subseteq K'$.* This stronger "FO-separability" property is in fact, in a precise sense, equivalent to Craig interpolation (see the discussion in Chapter 7 on the *separation property*).

The name "Δ-interpolation" originates from a notation system commonly used for denoting logical quantifier prenex hierarchies. More precisely, Σ^1_n and Π^1_n are fragments of second-order logic defined as follows (for $n \geq 0$): $\Sigma^1_0 = \Pi^1_0 = \text{FO}$; Σ^1_{n+1} consists of all second-order formulas of the form $\exists R_1 \dots R_n \phi$ for $\phi \in \Pi^1_n$ while Π^1_{n+1} consists of all second-order formulas of the form $\forall R_1 \dots R_n \phi$ for $\phi \in \Sigma^1_n$. A class of structures is said to be in Σ^1_n, or in Π^1_n, if it can be defined by a Σ^1_n-sentence, respectively, by a Π^1_n-sentence. Finally, a class of structures in Δ^1_n is in both Σ^1_n and Π^1_n (where Δ stands for *Durchschnitt*, the German term for intersection). Following this notational convention, ∃SO $= \Sigma^1_1$ and ∀SO $= \Pi^1_1$, and the above Δ-interpolation states that $\Delta^1_1 = \text{FO}$. Δ-interpolation is also known as *Souslin-Kleene interpolation*, owing to the similarity it bears to the Souslin-Kleene theorem ([59, Chapter 20]), as well as to Post's theorem ([59, Chapter 14]), in recursion theory and descriptive set theory. For further reading on Souslin-Kleene interpolation, see the discussion in Chapter 7.

3.2 Preservation Theorems

Another important application of interpolation theorems is that they imply preservation theorems. We already saw an example of this (Corollary 2.16), but we will discuss this in some more detail here. A preservation theorem is a theorem that characterizes a semantic property of FO formulas in terms of a syntactic property. We will illustrate this with a concrete example. Let us say that an FO formula ϕ is *upward monotone* in a relation symbol R if the following holds for all structures A and B: if A and B have the same domain and the same interpretation for every symbol other than R, and $R^A \subseteq R^B$, and $A, g \models \phi$ for some variable assignment g, then also $B, g \models \phi$. The following preservation theorem follows from the Lyndon interpolation theorem:

▶ **Theorem 3.3** (Lyndon preservation [43]). *For all* FO *formulas* ϕ *and relation symbols* R*, the following are equivalent:*

1. ϕ *is upward monotone in* R*.*
2. ϕ *is equivalent to an* FO *formula* ϕ' *with* $R \notin \mathrm{relsig}^-(\phi)$*.*

Proof. We only treat the interesting direction, namely from 1 to 2. Let ϕ be upward monotone in R. It follows that the following implication is valid:

$$\models \phi \to (\forall \mathbf{x}(R\mathbf{x} \to R'\mathbf{x}) \to \phi[R/R'])$$

where R' is a fresh relation symbol of the same arity as R, and $\phi[R/R']$ is the formula obtained by replacing every occurrence of R in ϕ by R'. Let ϑ be a Lyndon interpolant for this valid FO implication. Since R occurs only positively in the consequent of the implication (in the scope of two implications), it follows that $R \notin \mathrm{relsig}^-(\vartheta)$. Furthermore, it is easy to see that ϑ is equivalent to ϕ. ◀

The same general technique can be used to derive many other preservation theorems. For instance, the *Łos-Tarski preservation theorem* and the *homomorphism preservation theorem* (cf. [37]) can both be derived from the Otto interpolation theorem. Similarly, Feferman's preservation theorems characterizing the FO formulas that are *persistent for outer extensions* and the FO formulas that are *invariant for outer extensions* (a.k.a. the formulas *preserved by*, respectively, *invariant for generated substructures*) [29, 3] follow from the access interpolation theorem.

3.3 The Beth Definability Theorem

We introduce the basic idea of the Beth definability theorem through an example.

▶ **Example 3.4.** Consider the binary relation "taller-than" and the unary relation "tallest". There are a number of reasonable axioms that one could write down, governing the relationship between these two concepts. These could include, for instance, axioms expressing that "taller-than" is a linear order, the axiom $\forall xy(\text{taller-than}(y, x) \to \neg\text{tallest}(x))$, or the axiom $\exists x(\text{tallest}(x))$. Let Σ be the FO theory consisting of these axioms. It can then be seen that tallest is definable in terms of taller-than, relative to Σ. Indeed,

$$\Sigma \models \forall x(\text{tallest}(x) \leftrightarrow \neg\exists y(\text{taller-than}(y, x)))$$

Using terminology that we will introduce below, we call the above an *explicit definition of tallest in terms of taller-than relative to* Σ.

What about the converse: is taller-than explicitly definable in terms of tallest relative to Σ? It seems not. We can give a formal argument as follows: consider the following pair of structures

- A is the structure with domain $\{a, b, c\}$ where taller-than$^A = \{(a, b), (b, c), (a, c)\}$ and tallest$^A = \{a\}$.
- B is the structure with domain $\{a, b, c\}$ where taller-than$^B = \{(a, c), (c, b), (a, b)\}$ and tallest$^B = \{a\}$.

Both structures satisfy Σ. Furthermore, both structures have the same domain, and the tallest relation has the same denotation in both structures. If taller-than were explicitly definable in terms of tallest, the taller-than relation would therefore also have to have the same denotation in both structures. But this is not the case. Therefore, we can conclude that taller-than is *not* explicitly definable in terms of tallest, relative to Σ. ⌟

This technique for showing non-definability (by providing a pair of structures that agree on their domain and on the base relations but that disagree on the relation to be defined) is known as Padoa's method [47]. The Beth definability theorem [16], intuitively, states that Padoa's method is a sound and complete method for proving non-definability: whenever a relation R is *not* definable in terms of relations $S_1, \ldots, S_n$ relative to a first-order theory Σ, then this can be shown using Padoa's method. Formally:

▶ **Definition 3.5.** *Let Σ be an* FO *theory, let $R \in \mathrm{sig}(\Sigma)$ and let $\tau \subseteq \mathrm{sig}(\Sigma)$.*

1. *An* explicit definition *of R in terms of τ relative to Σ is a FO-formula $\phi(x_1, \ldots, x_n)$, with $n = arity(R)$, such that $\Sigma \models \forall x_1 \ldots x_n (R(x_1, \ldots, x_n) \leftrightarrow \phi(x_1, \ldots, x_n))$.*
2. *Σ* implicitly defines *R in terms of τ if for all models A and B of Σ having the same domain, $S^A = S^B$ for all $S \in \tau$ implies $R^A = R^B$.*

▶ **Example 3.6.** Consider the theory $\Sigma = \{\forall xy(V_n(x, y) \leftrightarrow \phi_n(x, y)) \mid n > 1\}$, where $\phi_n(x, y)$ expresses the existence of a directed R-path of length n from x to y. Clearly, $\Sigma \models V_4(x, y) \leftrightarrow \exists u\,(V_2(x, u) \wedge V_2(u, y))$. In other words, V_4 is explicitly definable in terms of V_2 relative to Σ. On the other hand, V_4 is not explicitly definable in terms of V_3. Following Padoa's method, this can be proved by showing that Σ does not implicitly define V_4 in terms of V_3 (we omit the details). What is less obvious is that V_5 *is* explicitly definable in terms of V_3 and V_4. We leave it as an exercise to the reader to find the explicit definition (cf. [1] for a solution). ⌟

▶ **Theorem 3.7** (Projective Beth definability)**.** *Let Σ be an* FO *theory, $R \in \mathrm{sig}(\Sigma)$ and $\tau \subseteq \mathrm{sig}(\Sigma)$. There is an explicit definition of R in terms of τ relative to Σ if and only if Σ implicitly defines R in terms of τ.*

Proof. We will only discuss the more interesting right-to-left direction. To simplify the exposition, we will restrict attention to relational signatures. Assume Σ implicitly defines R in terms of τ. For each $S \in \mathrm{sig}(\Sigma)$, let S' be a fresh relation symbol of the same arity as S. Let Σ' be the theory obtained by replacing each relation symbol in Σ by its primed copy. From the fact that Σ implicitly defines R in terms of τ, it follows that

$$\Sigma \cup \Sigma' \models \left(\bigwedge_{S \in \tau} \forall \mathbf{x}\,(S\mathbf{x} \leftrightarrow S'\mathbf{x}) \rightarrow (R\mathbf{y} \rightarrow R'\mathbf{y}) \right)$$

By compactness, there exist finite $\Sigma_0 \subseteq \Sigma$ and $\Sigma'_0 \subseteq \Sigma'$ such that

$$\Sigma_0 \cup \Sigma'_0 \models \left(\bigwedge_{S \in \tau} \forall \mathbf{x} \, (S\mathbf{x} \leftrightarrow S'\mathbf{x}) \right) \rightarrow (R\mathbf{y} \rightarrow R'\mathbf{y})$$

Rearranging the formulas a bit, we obtain

$$\models \left(\left(\bigwedge \Sigma_0 \right) \wedge R\mathbf{y} \right) \rightarrow \left(\left(\bigwedge \Sigma'_0 \right) \wedge \left(\bigwedge_{S \in \tau} \forall \mathbf{x} \, (S\mathbf{x} \leftrightarrow S'\mathbf{x}) \right) \rightarrow R'\mathbf{y} \right)$$

Let ϑ be any Craig interpolant. Observe that $\mathrm{sig}(\vartheta) \subseteq \tau$. It is not difficult to show that

$$\Sigma \models \forall \mathbf{y}(R\mathbf{y} \rightarrow \vartheta(\mathbf{y}))$$

and

$$\Sigma' \cup \{\forall \mathbf{x}(S\mathbf{x} \leftrightarrow S'\mathbf{x}) \mid S \in \tau\} \models \forall \mathbf{y}(\vartheta(\mathbf{y}) \rightarrow R'\mathbf{y})$$

Since first-order validity is preserved when we uniformly substitute a relation symbol by another relation symbol, it follows (by replacing all primed relation symbols by their unprimed version) that $\Sigma \models \forall \mathbf{y}(\vartheta(\mathbf{y}) \rightarrow R\mathbf{y})$, and therefore

$$\Sigma \models \forall \mathbf{y}(R\mathbf{y} \leftrightarrow \vartheta(\mathbf{y}))$$

In other words, ϑ is an explicit definition for R in terms of τ relative to Σ. ◀

In the literature, this is known as the *projective* Beth definability theorem. The non-projective Beth definability theorem (as originally proved in [16]) is the special case where $\tau = \mathrm{sig}(\Sigma) \setminus \{R\}$.

▶ **Example 3.8** (Order-invariant FO(<) = FO)**.** This example shows the projective Beth definability theorem in action by using it as a stepping stone towards a further application. Fix a signature τ. We say that an FO sentence ϕ over $\tau \cup \{<\}$ is *order-invariant* if for all τ-structures A and for all linear orders $<_1, <_2$ over the domain of A, it holds that $(A, <_1) \models \phi$ iff $(A, <_2) \models \phi$. If ϕ is an order-invariant FO sentence over $\tau \cup \{<\}$ and A is a τ-structure, we may simply write $A \models \phi$ to denote that $(A, <) \models \phi$ for an arbitrary linear order $<$ (because the choice of linear order does not matter). By *order-invariant first-order logic* (notation: $\mathrm{FO}(<)_{\mathrm{inv}}$) over a signature τ we mean the collection of order-invariant FO sentences over the signature $\tau \cup \{<\}$. $\mathrm{FO}(<)_{\mathrm{inv}}$ can be viewed as a language for describing τ-structures. From this perspective, FO is a fragment of $\mathrm{FO}(<)_{\mathrm{inv}}$, because, if a FO sentence does not contain any occurrences of $<$, it is trivially order-invariant. It is conceivable, however, that $\mathrm{FO}(<)_{\mathrm{inv}}$ would be strictly more expressive than FO. Indeed, this turns out to be the case *in the finite*: there are $\mathrm{FO}(<)$ sentences that are order-invariant over finite structures, but that are not equivalent, over finite structures, to an FO sentence in the signature without the order (cf. [58]). Over arbitrary (finite and infinite) structures, however, $\mathrm{FO}(<)_{\mathrm{inv}}$ has the same expressive power as FO, and the projective Beth definability theorem can be used to prove this, as we will explain in a moment.

Excursus. A beautiful example of the expressive power of $\mathrm{FO}(<)_{\mathrm{inv}}$ on finite structures is given in [49] (cf. also [58]): consider *finite binary trees* as $\{E, D\}$-structures where E is the edge relation and D is the descendant relation, i.e., the transitive closure of E. Let us furthermore restrict to *perfect* finite binary trees, meaning that every non-leaf node has exactly two children, and that all leaves have equal distance from the root.

There is an order-invariant $\mathrm{FO}(E, D, <)$-sentence ϕ such that for every such perfect finite binary tree T, we have that $T \models \phi$ iff every leaf of T has an even distance from the root. It can be shown using standard techniques (e.g., Ehrenfeucht-Fraïsse games) that there is no $\mathrm{FO}(E, D)$-sentence ψ with the same property. We first define the shorthand notations

$$\begin{aligned} \mathrm{leaf}(x) &:= \neg\exists y E(x, y), \text{ and} \\ \mathrm{firstchild}(x) &:= \neg\exists yz(E(y, x) \wedge E(y, z) \wedge z < x), \text{ and} \\ \mathrm{secondchild}(x) &:= \exists yz(E(y, x) \wedge E(y, z) \wedge z < x). \end{aligned}$$

The formulas defining $\mathrm{firstchild}(x)$ and $\mathrm{secondchild}(x)$ use the ordering on the structure to label each non-root node as the first or second child of its parent. The core idea behind the construction is expressed by the formula

$$\begin{aligned} &\exists z(\mathrm{leaf}\,(z) \wedge \mathrm{firstchild}\,(z) \wedge \\ &\qquad \forall xy\,(E\,(x, y) \wedge D\,(y, z) \rightarrow (\mathrm{firstchild}\,(y) \leftrightarrow \neg\mathrm{firstchild}\,(x)))), \end{aligned}$$

which expresses the existence of a leaf node n, such that the path from the root to n is of the form as described by the regular expression "$(\mathrm{firstchild} \cdot \mathrm{secondchild})^* \cdot \mathrm{firstchild}$". It follows that the above formula is true in perfect trees of even depth and false in perfect trees of odd depth. This is not a complete argument for $\mathrm{FO} \subsetneq \mathrm{FO}(<)_{\mathrm{inv}}$: more work is needed to ensure that the formula is order-invariant over all structures (not only over perfect binary trees). Nevertheless, we hope that the above explanation gives the basic intuition.

As we will now show, it follows from the projective Beth definability theorem that, over arbitrary (i.e., finite and infinite) structures, every $\mathrm{FO}(<)_{\mathrm{inv}}$ sentence is equivalent to an FO sentence. Indeed, fix an arbitrary FO sentence φ over the signature $\tau \cup \{<\}$, and let Σ be the FO theory over the signature $\tau \cup \{<, Z\}$ expressing that $<$ is a linear order and that $Z \leftrightarrow \phi$, where Z is a 0-ary relation symbol not occurring in τ. Then ϕ is order-invariant if and only if all structures with the same domain which agree on relations in τ (but not necessarily the order $<$) also agree on Z. In other words, ϕ is order invariant precisely when Σ implicitly defines Z in terms of τ. When this holds, by the projective Beth definability theorem, there is an explicit definition of Z in terms of τ, i.e., $\Sigma \models Z \leftrightarrow \psi$ for some FO sentence ψ over the original signature τ (i.e., without the linear order). It follows that ϕ and ψ are equivalent in

the sense that both define the same class of τ-structures. ⌟

There is much more to say about Beth definability. In particular, Chapter 16 discusses relevant connections to various topics in philosophy.

3.4 Combining Knowledge Bases

In the field of *knowledge representation*, one important set of problems that have been studied in depth pertains to *knowledge-base management*. This set of problems includes *combining* knowledge bases that have partially overlapping domains, as well as, for instance, *restricting* a knowledge base to a subdomain. Many of these tasks turn out to be intimately related to variants of Craig interpolation or uniform interpolation. See Chapter 15 for an in-depth exposition of these connections. Here, we will merely illustrate this through one basic result. In general, a knowledge base can be thought of as a theory specified in a suitable fragment of FO logic. For simplicity, let us consider, here, a knowledge base to simply be an FO theory. Then the following theorem, which follows immediately from the Craig interpolation theorem, tells us that two knowledge bases can be safely combined with each other, as long as they do not contradict each other on statements in their common signature.

▶ **Theorem 3.9** (Robinson joint consistency [57]). *For all* FO *theories* Σ_1 *and* Σ_2*, the following are equivalent:*

1. $\Sigma_1 \cup \Sigma_2$ *is consistent,*
2. *There is no* FO *sentence* ϕ *with* $\mathrm{sig}(\phi) \subseteq \mathrm{sig}(\Sigma_1) \cap \mathrm{sig}(\Sigma_2)$*, such that* $\Sigma_1 \models \phi$ *and* $\Sigma_2 \models \neg\phi$*.*

Proof. The direction from 1 to 2 is immediate. For the converse direction, if $\Sigma_1 \cup \Sigma_2$ is inconsistent, then, by compactness, there are finite subsets $\Sigma_1' \subseteq \Sigma_1$ and $\Sigma_2' \subseteq \Sigma_2$ such that $\Sigma_1' \cup \Sigma_2'$ is inconsistent. It follows that

$$\models (\bigwedge \Sigma_1') \to \neg(\bigwedge \Sigma_2').$$

and for any Craig interpolant φ for this valid FO implication, we have that $\Sigma_1 \models \phi$ and $\Sigma_2 \models \neg\phi$. ◀

3.5 Formal Verification

As will be discussed in depth in Chapter 13, variants of Craig interpolation have been fruitfully applied in the area of *formal verification*. We illustrate this use case here by means of a simple example.

▶ **Example 3.10.** Consider the following computer program.

```
function f(int x):  int {
  // Input:  an integer that is assumed to be divisible by 3
  // Output:  an integer that is divisible by 3.
```

```
    // assert:  x is divisible by 3
    x := x+1;
    // assert:  ?
    x := x+2;
    // assert:  x is divisible by 3
    return x;
}
```

If the initial assertion holds before the program execution then the final assertion holds afterwards. In order to prove this, we would insert an intermediate assertion in the spot marked by ? and show that each instruction preserves the truth of the surrounding assertions (that is, the pre-condition of each instruction implies that the post-condition holds after executing the instruction). What would be the right assertion to put in place of the ?? The answer is obvious, namely: *x is congruent to 1 modulo 3*. However, one would like to have a method that allows us to find this answer automatically. As it turns out, Craig interpolation provides such a method.

Clearly, the reasoning here requires some arithmetic. We can think of it as reasoning relative to a first-order theory such as Peano arithmetic. In fact, for this example it suffices to use Presburger arithmetic (PrA), i.e., the first-order theory of the integers with addition and without multiplication. Note that *x is divisible by 3* can be expressed in the language of PrA as $\exists u(x = u + u + u)$ (which we will abbreviate, in what follows, as $\exists u(x = 3u)$). Next, in order to make the reasoning more precise, let us introduce variables x_0, x_1, x_2 to distinguish the values of x at each stage of the computation. We can then formalize the correctness claim that we wish to prove as follows:

$$\mathrm{PrA} \models (\exists u\,(x_0 = 3u) \wedge x_1 = x_0 + 1 \wedge x_2 = x_1 + 2) \rightarrow \exists v\,(x_2 = 3v)$$

Rearranging the formula a bit, we get:

$$\models \left(\left(\bigwedge \mathrm{PrA}\right) \wedge (\exists u\,(x_0 = 3u) \wedge x_1 = x_0 + 1)\right) \rightarrow (x_2 = x_1 + 2 \rightarrow \exists v\,(x_2 = 3v))$$

We can now see that the only shared free variable between the antecedent and the consequent is x_1. Therefore, there should exist some interpolant with x_1 as its only free variable. Indeed, one such interpolant is $\exists u(x_1 = 3u + 1)$. It follows that we can put $\exists u(x = 3u + 1)$ for the ?. In fact, we can even simplify this assertion a bit further, using the fact that Presburger arithmetic admits quantifier elimination. This finally leads us to the solution we wanted, namely the assertion that x is congruent to 1 modulo 3. See also Section 5 where we consider interpolation relative to a FO theory in more detail. ⌟

Similar techniques can be fruitfully applied for the formal verification of more complicated code involving loops or recursion (see Chapter 13).

4 Interpolation for Decidable Fragments of First-Order Logic

In this section, we discuss the situation regarding Craig interpolation for a number of important decidable fragments of first-order logic. See Chapter 7 for a similar overview of results about interpolation for extensions of first-order logic. This section is largely based on [20].

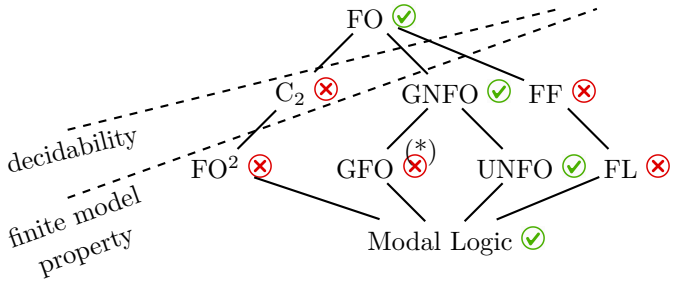
4.1 Decidable Fragments

The study of decidable fragments of FO is a topic with a long history, dating back to the early 1900s ([44, 60], cf. also [18]), and more actively pursued since the 1990s. Inspired by Vardi [63], who asked "what makes modal logic so robustly decidable?" and Andréka et al. [2], who asked "what makes modal logic tick?" many decidable fragments have been introduced and studied over the last 25 years that take inspiration from modal logic (ML), which itself can be viewed as a fragment of FO that features a restricted form of quantification. These include the following fragments, each of which naturally generalizes modal logic in a different way: the *two-variable fragment* (FO^2) [45], the *guarded fragment* (GFO) [2], and the *unary negation fragment* (UNFO) [22]. Further decidable extensions of these fragments were subsequently identified, including the *two-variable fragment with counting quantifiers* (C_2) [34] and the *guarded negation fragment* (GNFO) [6]. The latter can be viewed as a common generalization of GFO and UNFO. Many decidable logics used in computer science and AI, including various description logics and rule-based languages, can be translated into GNFO and/or C_2. In this sense, GNFO and C_2 are convenient tools for explaining the decidability of other logics. Extensions of GNFO have been studied that push the decidability frontier even further (for instance with fixed-point operators [10, 9] and using clique-guards), but these fall outside the scope of this paper.

In an earlier line of investigation, Quine identified the decidable *fluted fragment* (FL) [56], the first of several *ordered logics* which have been the subject of recent interest [52, 53, 54, 55, 50]. The idea behind ordered logics is to restrict the order in which variables are allowed to occur in atomic formulas and quantifiers. Another recently introduced decidable fragment that falls in this family is the *forward fragment* (FF), whose syntax strictly generalizes that of FL. Both FL and FF have the finite model property (FMP) [50] and embed ML [39], but are incomparable in expressive power to GFO [51], FO^2, and UNFO. The recently-introduced *adjacent fragment* (AF) generalizes both FF and FL while also retaining decidability [8].

▶ **Example 4.1.** The FO-sentence $\exists xy(R(x,y) \wedge R(y,x))$ belongs to GFO, FO^2 and UNFO, but is not expressible in FF, since the structure consisting of two points with symmetric edges and the structure $(\mathbb{Z}, S)$ with S the successor relation, are "infix bisimilar" (see [7]). ⌟

Ideally, an FO-fragment is not only decidable, but also model-theoretically well behaved. A particularly important model-theoretic property of logics is the *Craig Interpolation Property* (CIP). It turns out that, although GFO is in many ways model-theoretically well-behaved [2], it lacks CIP [38]. Likewise, FO^2 lacks CIP [23] and the same holds for C_2 ([40, Example 2]



FO	First-order logic	GNFO	Guarded-negation fragment
FO^2	Two-variable fragment	UNFO	Unary-negation fragment
C_2	Two-variable fragment with counting	FF	Forward fragment
GFO	Guarded fragment	FL	Fluted fragment

Figure 1 Landscape of decidable fragments of FO with (✓) and without (✗) CIP. The inclusion marked (∗) holds only for sentences and self-guarded formulas. Figure taken from [20].

yields a counterexample). Both FF and FL lack CIP [7]. On the other hand, UNFO and GNFO have CIP [22, 5]. Figure 1 summarizes these known results. Note that we restrict attention to relational signatures without constant symbols and function symbols. Some of the results depend on this restriction. Other known results not reflected in Figure 1 (to avoid clutter) are that the intersection of GFO and FO^2 (also known as GFO_2) has CIP [38]. Similarly, the intersection of FF with GFO and the intersection of FL with GFO (known as GFF and GFL, respectively) have CIP [7].

Additional results are known for UNFO and GNFO beyond the fact that they have CIP. In particular, CIP holds for the fixed-point extension of UNFO, while the weak "modal" form of Craig interpolation, mentioned above, holds for the fixed-point extension of GNFO [13, 9]. Furthermore, interpolants for UNFO and GNFO can be constructed effectively, and tight bounds are known on the size of interpolants and the computational complexity of computing them [11].

4.2 Repairing Interpolation

When a logic L lacks CIP, the question naturally arises as to whether there exists a more expressive logic L' that has CIP. If such an L' exists, then, in particular, interpolants for valid L-implications can be found in L'. This line of analysis is sometimes referred to as *Repairing Interpolation* [4]. Another alternative is to find a logic L' which contains interpolants for all valid implications in L (see the Craig(L, L')) property in Chapter 7 and in Chapter 10.

If L is an FO-fragment, and our aim is to repair interpolation by extension, then there is a trivial solution: FO itself is an extension of L satisfying CIP. More interesting is the following question: can a natural extension L' of L be identified which satisfies CIP while retaining decidability? In [20], this question is answered in the negative: FO is the smallest

extension of FO^2 and of FF satisfying CIP. With some simple assumptions regarding the effective computability of the translation between the extension and the logic that it extends, it follows that no extension of FO^2 or FF satisfying CIP is decidable. It is also shown that the smallest logic extending GFO that has CIP is GNFO.

These results give us a clear sense of where, in the larger landscape of decidable fragments of FO, we may find logics that enjoy CIP. What makes the above results remarkable is that, from the definition of the Craig interpolation property, it doesn't appear to follow that a logic without CIP would have a unique minimal extension with CIP. Note that a valid implication may have many possible interpolants, and the Craig interpolation property merely requires the existence of one such interpolant. Nevertheless, the above results show that, in the case of FO^2, GFO, and FF, such a unique minimal extension indeed exists (assuming suitable closure properties). It is worth noting that, in the case of these logics, their Δ-closure (which is the smallest logic $\Delta(L)$ extending a logic L and satisfying the Δ-interpolation property) already satisfies the stronger Craig interpolation property, and hence is also a minimal extension satisfying Craig interpolation. For more on this, see Chapter 7.

The aforementioned line of work can be viewed as an instance of *abstract model theory* for fragments of FO. One large driving force behind the development of abstract model theory was the identification of *extensions* of FO which satisfy desirable model-theoretic properties, such as the compactness theorem, the Löwenheim-Skolem, and Craig interpolation. One takeaway from this line of research is that CIP is scarce among many "reasonable" FO-extensions. For an overview of results regarding interpolation for extensions of FO, see Chapter 7.

Some prior work exists that studies CIP for (extensions of) fragments of FO using the lens of abstract model theory. It was shown in [19] that the smallest extension of modal logic with the difference operator (ML_D) which satisfies CIP is full first-order logic. Additionally, in [33], the authors identified minimal extensions of various fragments of propositional linear temporal logic with CIP. Repairing interpolation has also been pursued in the context of quantified modal logics, which typically lack CIP; in [4], the authors showed that CIP can be repaired for such logics by adding nominals, @-operators and the $\downarrow$-binder.

4.3 Alternative Approaches

Several other approaches have been proposed for dealing with logics that lack CIP. One approach is to weaken CIP. For example, it was shown in [38] that GFO satisfies a weak, "modal" form of Craig interpolation, where, roughly speaking, only the relation symbols that occur in non-guard positions in the interpolant are required to occur both in the premise and the conclusion. As it turns out, this weakening of CIP is strong enough to entail the (non-projective) *Beth Definability Property*, which is one important use case of CIP. We have discussed such weakenings of CIP in Sections 3.1 and 3.3.

Another recent approach [40] is to develop algorithms for testing whether an interpolant exists for a given entailment. That is, rather than viewing Craig interpolation as a property of logics, the existence of interpolants is studied as an algorithmic problem at the level of

individual entailments. The interpolant existence problem turns out to be indeed decidable (although of higher complexity than the satisfiability problem) for both GFO and FO^2 [40]. See Chapter 10 for more details.

5 Interpolation Relative to a Theory

Example 3.10 involved Craig interpolation under an FO theory. In this section we briefly explore this in a bit more detail.

▶ **Definition 5.1** (Craig interpolant under a theory)**.** *Let* Σ *be an* FO *theory and* ϕ, ψ FO *formulas such that* $\Sigma \models \phi \to \psi$*.*

1. *A* weak Craig interpolant for $\phi \to \psi$ under Σ *is a* FO *formula* ϑ *such that* $\Sigma \models \phi \to \vartheta$*,* $\Sigma \models \vartheta \to \psi$ *and* $\mathrm{sig}(\vartheta) \subseteq (\mathrm{sig}(\phi) \cap \mathrm{sig}(\psi)) \cup \mathrm{sig}(\Sigma)$ *and* $\mathrm{fvar}(\vartheta) \subseteq \mathrm{fvar}(\phi) \cap \mathrm{fvar}(\psi)$*.*
2. *If, furthermore,* $\mathrm{sig}(\vartheta) \subseteq \mathrm{sig}(\phi) \cap \mathrm{sig}(\psi)$*, then we say that* ϑ *is a* strong Craig interpolant for $\phi \to \psi$ under Σ*.*

Intuitively, weak Craig interpolants can contain non-logical symbols from the ambient theory Σ, even if they don't occur in ϕ or ψ, while strong Craig interpolants are strictly limited to using those symbols in both ϕ and ψ. As the next example shows, strong Craig interpolants do not always exist.

▶ **Example 5.2.** Let PA be the theory of Peano arithmetic (which is a theory of the natural numbers with addition, multiplication, and order). Let $\phi = \exists x(Px \wedge Qx \wedge \exists y(x = y + 1))$ and let $\psi = \exists x(Qx \wedge (x > 0 \vee R(x)))$. Then $\mathrm{PA} \models \phi \to \psi$. A weak Craig interpolant in this case is $\exists x(Qx \wedge x > 0)$. Since $>$ is not a shared symbol of ϕ and ψ, this is not a strong Craig interpolant. Indeed, it can easily be seen that there is no strong Craig interpolant in this case. Intuitively, the antecedent uses the symbols $+$ and 1 to force that x is positive, while the consequent uses the symbols $>$ and 0 to express that x is positive. A Craig interpolant can be found using either of these but cannot be constructed without using *any* arithmetical symbols. ⌟

The next theorem shows that weak Craig interpolants always exist, and provides a sufficient condition for the existence of strong Craig interpolants. Let us say that an FO theory Σ is (σ, τ)*-splittable* if it can be decomposed as $\Sigma = \Sigma_1 \cup \Sigma_2$ with $(\mathrm{sig}(\Sigma_1) \cup \sigma) \cap (\mathrm{sig}(\Sigma_2) \cup \tau) \subseteq \sigma \cap \tau$. For instance, it is not hard to see that if each formula in Σ involves only one symbol (i.e., if $|\mathrm{sig}(\phi)| = 1$ for each $\phi \in \Sigma$), then Σ is (σ, τ)-splittable for all choices of σ and τ.

▶ **Theorem 5.3.** *Let* Σ *be a* FO *theory and* ϕ, ψ FO *formulas such that* $\Sigma \models \phi \to \psi$*.*

1. *There is a weak Craig interpolant for* $\phi \to \psi$ *under* Σ*.*
2. *If* Σ *is* $(\mathrm{sig}(\phi), \mathrm{sig}(\psi))$*-splittable, then there is a strong Craig interpolant for* $\phi \to \psi$ *under* Σ*.*

Proof. 1. By compactness, we may assume without loss of generality that Σ is finite. If $\Sigma \models \phi \to \psi$, then $\models ((\bigwedge \Sigma) \wedge \phi) \to \psi$. Let ϑ be a Craig interpolant for this valid implication. Then, clearly, $\Sigma \models \phi \to \vartheta$ and $\Sigma \models \vartheta \to \psi$. Furthermore, $\mathrm{sig}(\vartheta) \subseteq (\mathrm{sig}(\phi) \cup \mathrm{sig}(\Sigma)) \cap \mathrm{sig}(\psi) \subseteq (\mathrm{sig}(\phi) \cap \mathrm{sig}(\psi)) \cup \mathrm{sig}(\Sigma)$.

2. By compactness, we may assume without loss of generality that Σ is finite. If $\Sigma \models \phi \to \psi$, then $\models ((\bigwedge \Sigma_1) \wedge \phi) \to ((\bigwedge \Sigma_2) \to \psi)$, where $\Sigma_1 \cup \Sigma_2$ is a $(\mathrm{sig}(\phi), \mathrm{sig}(\psi))$-splitting of Σ. Let ϑ be a Craig interpolant for this valid implication. Then, clearly, $\Sigma \models \phi \to \vartheta$ and $\Sigma \models \vartheta \to \psi$. Furthermore, $\mathrm{sig}(\vartheta) \subseteq (\mathrm{sig}(\phi) \cup \mathrm{sig}(\Sigma_1)) \cap (\mathrm{sig}(\psi) \cup \mathrm{sig}(\Sigma_2)) = \mathrm{sig}(\phi) \cap \mathrm{sig}(\psi)$. ◀

An interesting question is whether restricted fragments of FO allow weak Craig interpolation under certain first-order theories. We will give two examples of results on this:

1. The *modal* fragment of FO has weak Craig interpolation under every first-order theory consisting of universal Horn sentences (or more generally, relative to any first-order theory that is closed under bisimulation products). In particular, this implies that many relevant modal logics, such as K4 and S4, have Craig interpolation. See Chapter 4 for more details.
2. The *quantifier-free* fragment of FO has (a restricted form of) weak Craig interpolation under every first-order theory that admits quantifier elimination. See Chapter 8 for more details.

Other examples can be found in knowledge representation (see Chapter 15). For instance, the concept language $\mathcal{ALC}$ has weak Craig interpolation under $\mathcal{ALC}$-TBoxes [21].

Similarly, one may ask when FO itself admits weak Craig interpolation relative to a *second-order* theory. For instance, the class of finite structures is definable by a (finite) second-order theory (cf. Example 3.1), and Theorem 2.17 implies that FO lacks weak Craig interpolation relative to this second-order theory. Similarly, the class of ω-words over a finite alphabet is definable by a (finite) second-order theory, and FO over this class of structures (or, equivalently by Kamp's theorem [41], *Linear Temporal Logic*), lacks Craig interpolation [48, 33] (see also Chapter 10).

6 Proof Methods and Algorithmic Aspects

There are two main styles of techniques for proving interpolation theorems for FO, namely *model theoretic* and *proof theoretic*. At a high level, model theoretic proofs proceed along the following lines. Suppose, for the sake of a contradiction, that some valid implication $\models \phi \to \psi$ does *not* have a suitable interpolant. For simplicity, assume that ϕ and ψ are sentences. Using compactness, it can then be shown that there exist structures $A \models \phi$ and $B \models \neg\psi$ such that A and B have the same FO-theory in the signature $\mathrm{sig}(\phi) \cap \mathrm{sig}(\psi)$. Next, A and B are then "amalgamated" into a single structure (typically using ω-saturated ultrapowers) that simultaneously satisfies ϕ and $\neg\psi$, a contradiction. Note that such proofs are inherently *non-constructive*: they only show that interpolants exist but do not provide a way to construct

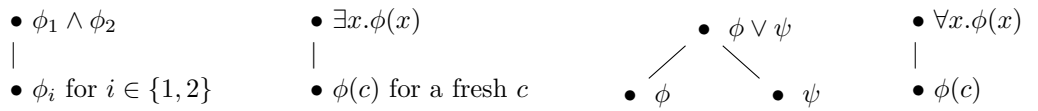
an interpolant. Chapter 7 contains such a model-theoretic proof of Craig interpolation for first-order logic.

On the other hand, *proof theoretic* techniques construct an interpolant for $\models \phi \to \psi$ from a proof of $\phi \to \psi$ in a suitable proof system. Below, we show how proofs in semantic tableaux can be used to construct interpolants. A detailed discussion on constructing interpolants from sequent-calculus proofs can be found in Chapter 5 and Chapter 11, and a discussion of more practical algorithms for computing interpolants can be found in Chapter 12 and Chapter 13. We note that semantic tableaux, as a proof system for first-order logic, are closely related to sequent calculi. Indeed, a closed tableau for a formula, when read "upside-down", corresponds to a cut-free sequent calculus proof for its negation.

6.1 A Proof of Craig interpolation via Semantic Tableaux

We will now show how to prove a Craig interpolation theorem via semantic tableaux. To keep the proof as simple as possible, we will restrict attention to equality-free FO sentences over signatures without function symbols: this restriction reduces the number of tableau rules, and allows us to avoid some technicalities regarding substitutability of variables by other variables. The proof we present, however, can be extended to the general case, and all the interpolation theorems for FO that we discussed in Section 2 can be proved using the same approach. More detailed but accessible presentations of constructive proofs of interpolation theorems for FO via semantic tableaux can be found in the textbooks [31] and [12].

A *semantic tableau calculus* (or, simply, a *tableau calculus*) is defined by a set of "expansion rules" and "closure rules". Typically, each rule pertains to a different connective. A tableau is then a tree whose nodes are labeled by sentences, and where each non-leaf node corresponds to the application of an expansion rule, while each leaf node corresponds to the application of a closure rule. We will now present a concrete tableau calculus for equality-free FO sentences over signatures without function symbols. We will assume that the input sentences are in negation normal form. The tableau calculus consists of four expansion rules and two closure rules. The expansion rules are:



where, in the second and the fourth rule, $\phi(c)$ is obtained from ϕ by replacing all free occurrences of x by the constant symbol c.

Each rule should be read as follows: in order to fulfill the upper requirement, it is necessary to fulfill the lower requirements (or, in case of the third rule, one of the lower requirements). For instance, the first rule says that, in order to satisfy a conjunction, we must satisfy both conjuncts. Similarly, the second rule says that, to satisfy $\exists x\phi(x)$, we must satisfy $\phi(c)$ for

some fresh constant c (i.e., a constant symbol not yet occurring on the tableau branch). The third rule is a "branching" rule. It says that, to satisfy a disjunction, we must satisfy one of the disjuncts. The fourth rule differs subtly from the second one: to satisfy $\forall x\phi(x)$ it is necessary to satisfy $\phi(c)$ for every constant c, not only for a fresh constant c. This rule, unlike the other three, may sometimes need to be applied multiple times to the same formula (for different constants c).

The closure rules are the following two rules:

$$\begin{array}{ll} \bullet\ \bot & \bullet\ \alpha, \neg\alpha \\ | & | \\ \times & \times \end{array}$$

Intuitively, these rules say that a branch may be closed if either $\bot$ occurs on the branch or a formula and its negation both occur on the branch. The $\times$ symbol is used to mark branches that are closed (i.e., branches ending in a leaf node).

If Σ is a finite set of FO sentences, then a *closed tableau for* Σ is a finite tree whose root is labeled by Σ, where each non-leaf node corresponds to applying one of the expansion rules to a formula that has already been obtained (either in the previous step or at some earlier ancestor of the node in question), and where each branch ends in the application of a closure rule. Note that, while the root node of the tableau is labeled by a finite set of formulas, every other node is labeled by a single formula.

We can think of a tableau as a way to systematically search for a model that satisfies the input formulas. A closed tableau shows that no such model exists. Thus, a closed tableau constitutes a proof of unsatisfiability.

▶ **Example 6.1.** This example is adapted from [12, Example 2.2]. Consider the set of FO sentences

$$\Sigma = \{\exists x(A(x) \wedge \neg B(x) \wedge C(x)), \forall y((\neg A(y) \wedge E(y)) \vee B(y))\}\ .$$

We show that Σ is unsatisfiable by giving, in Figure 2, a closed tableau for Σ. The annotation using "L" and "R" in this figure can be ignored for now, and will be explained later. Intuitively, the tableau describes a systematic search for a model of Σ. Starting with the set Σ, we repeatedly break down the requirements that need to be satisfied in order to make Σ true. When encountering a disjunction, we explore both possibilities. Since each possibility leads to a contradiction (the tableau is closed), we can conclude that it is not possible to satisfy Σ. ⌟

▶ **Theorem 6.2** (Soundness and completeness [17])**.** *Let Σ be a set of equality-free* FO *sentences in negation normal form, over a signature without function symbols. Then Σ is unsatisfiable if and only if there is a closed tableau for Σ.*

We refer to [31, 12] for a proof of Theorem 6.2. In what follows, we will show how to use the above theorem to prove a Craig interpolation theorem. More precisely, we will prove

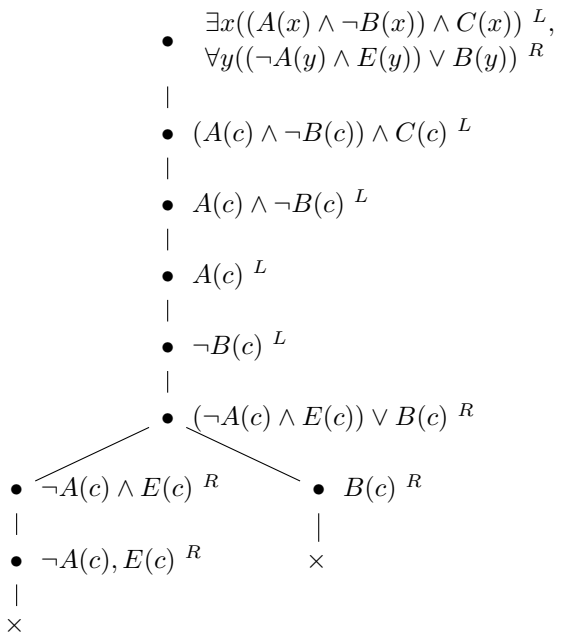


Figure 2 Example of a closed tableau.

the following theorem. It is easy to see that this is an equivalent rephrasing of the Craig interpolation theorem for FO, restricted to equality-free sentences without function symbols.

▶ **Theorem 6.3.** *For all function-free, equality-free* FO *sentences* ϕ, ψ*, if* $\{\phi, \psi\}$ *is unsatisfiable then there is a function-free, equality-free* FO *sentence* θ *such that* $\models \phi \to \theta$ *and* $\models \psi \to \neg\theta$ *and* $\mathrm{sig}(\theta) \subseteq \mathrm{sig}(\phi) \cap \mathrm{sig}(\psi)$.

Proof. Assume that $\{\phi, \psi\}$ is unsatisfiable. Then by Theorem 6.2, there is a closed tableau for $\{\phi, \psi\}$.

First, we annotate each formula occurring on the tableau with L or R depending on whether it was "derived" from ϕ or from ψ. Formally, we label ϕ and ψ as L and R respectively, and every formula which is derived on the basis of an expansion rule gets the same label as the formula to which the expansion rule was applied. In this way, every formula occurring in the tableau is labeled with L or R. See Figure 2 for an example.

Next, for every node n of the tableau, let $\chi_L(n)$ be the conjunction of all L-labeled sentences occurring at node n or at an ancestor of n, and let $\chi_R(n)$ be the conjunction of all R-labeled sentences occurring at node n or at an ancestor of n.

Claim: For every node n there is an equality-free FO sentence θ (which we will call the *interpolant* for n) such that $\models \chi_L(n) \to \theta$ and $\models \chi_R(n) \to \neg\theta$ and $\mathrm{sig}(\theta) \subseteq \mathrm{sig}(\chi_L(n)) \cap \mathrm{sig}(\chi_R(n))$.

This claim implies the theorem, as the interpolant of the root node is an interpolant for $\{\phi, \psi\}$ in the sense of the statement of the theorem. We prove the claim by induction on the size of the subtree below n. The base case is where n is a leaf, in which case one of the closure rules applies:

$$\begin{array}{ll} n \bullet \bot & \qquad n \bullet \alpha, \neg\alpha \\ \mid & \qquad \mid \\ \times & \qquad \times \end{array}$$

We can construct an interpolant for n as follows:

- If $\bot$ occurs on the branch as an L-labeled sentence, then $\bot$ is an interpolant.
- If $\bot$ occurs on the branch as an R-labeled sentence, then $\top$ is an interpolant.
- If α occurs on the branch as an L-labeled sentence and $\neg\alpha$ occurs on the branch as an R-labeled sentence, then α is an interpolant.
- If α occurs on the branch as an R-labeled sentence and $\neg\alpha$ occurs on the branch as an L-labeled sentence, then $\neg\alpha$ is an interpolant.
- If α occurs on the branch as an L-labeled sentence and $\neg\alpha$ occurs on the branch as an L-labeled sentence, then $\bot$ is an interpolant.
- If α occurs on the branch as an R-labeled sentence and $\neg\alpha$ occurs on the branch as an R-labeled sentence, then $\top$ is an interpolant.

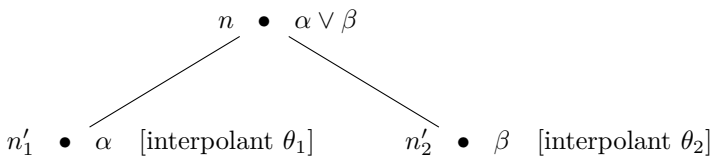
For non-leaf nodes n, we assume that an interpolant has already been computed for each of the children nodes of n, and we reason by cases depending on which of the four expansion rules was applied at node n.

$$\begin{array}{ll} n & \bullet\ \alpha_1 \wedge \alpha_2 \\ & \mid \\ n' & \bullet\ \alpha_i \quad [\text{interpolant } \theta] \end{array}$$

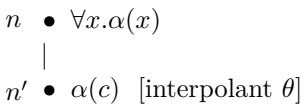
Observe that, in this case, $\chi_L(n)$ and $\chi_L(n')$ are equivalent and have the same signature, and the same holds for $\chi_R(n)$ and $\chi_R(n')$. It follows that θ is also an interpolant for n.

$$\begin{array}{ll} n & \bullet\ \exists x.\alpha(x) \\ & \mid \\ n' & \bullet\ \alpha(c) \text{ for a fresh } c \quad [\text{interpolant } \theta] \end{array}$$

Again, in this case, it can be shown that θ is also an interpolant for n. Note that, since c is fresh, it occurs only once, either in an L-labeled sentence or in an R-labeled sentence, and hence does not occur in the interpolant θ.



We now have two cases, depending on whether $\alpha \vee \beta$ occurs on the branch as an L-labeled sentence or an R-labeled sentence. In the former case, $\theta_1 \vee \theta_2$ is an interpolant for n. Otherwise, $\theta_1 \wedge \theta_2$ is an interpolant for n.



We have four cases, depending on whether $\forall x.\alpha(x)$ occurs on the branch as an L-labeled sentence or an R-labeled sentence, and also on whether c occurs on the branch in an L-labeled sentence or an R-labeled sentence.

- If $\forall x.\alpha(x)$ occurs on the branch as an L-labeled sentence and c occurs on the branch in an L-labeled sentence, then θ is an interpolant for n.
- If $\forall x.\alpha(x)$ occurs on the branch as an R-labeled sentence and c occurs on the branch in an R-labeled sentence, then θ is an interpolant for n.
- If $\forall x.\alpha(x)$ occurs on the branch as an L-labeled sentence and c does *not* occur on the branch in an L-labeled sentence, then $\forall x.\theta[c/x]$ is an interpolant for n.
- If $\forall x.\alpha(x)$ occurs on the branch as an R-labeled sentence and c does *not* occur on the branch in an R-labeled sentence, then $\exists x.\theta[c/x]$ is an interpolant for n.

◀

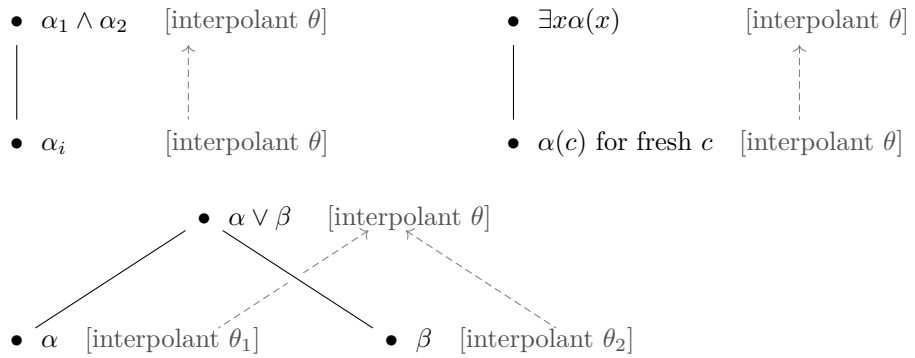
The interpolant constructions in the above proof are summarized in Figure 3.

6.2 Computational Complexity and Size Bounds

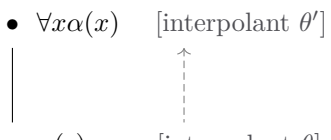
As described above, proof-theoretic proofs of Craig interpolation construct an interpolant for $\phi \to \psi$ from a proof of the validity of $\phi \to \psi$ (in a suitable proof system such as semantic tableaux). It follows from the undecidability of FO that there is no recursive function for constructing a proof of a valid FO implication, so such proof-theoretic approaches are *constructive* only in a weak sense: the interpolant can be effectively constructed once we have laid our hands on a suitable proof of $\phi \to \psi$. This is the best one can hope for:

▶ **Theorem 6.4** ([32]). *Let $|\phi|$ denote the length of an* FO *formula ϕ. There is no recursive function $f : \mathbb{N} \to \mathbb{N}$ such that every valid* FO *implication $\models \phi \to \psi$ has a Craig interpolant ϑ with $|\vartheta| \leq f(|\phi| + |\psi|)$.*

Expansion rules:

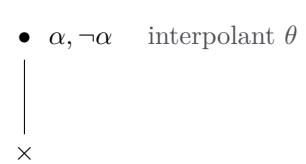


$$\text{where } \theta = \begin{cases} \theta_1 \vee \theta_2 & \text{if } \alpha \vee \beta \text{ is } L\text{-labeled} \\ \theta_1 \wedge \theta_2 & \text{if } \alpha \vee \beta \text{ is } R\text{-labeled} \end{cases}$$



$$\text{where } \theta' = \begin{cases} \exists x \theta[c/x] & \text{if } \forall x \alpha(x) \text{ is } R\text{-labeled and } c \text{ only occurs in } L\text{-labeled formulas} \\ \forall x \theta[c/x] & \text{if } \forall x \alpha(x) \text{ is } L\text{-labeled and } c \text{ only occurs in } R\text{-labeled formulas} \\ \theta & \text{otherwise} \end{cases}$$

Closure rules:



$$\text{where } \theta = \begin{cases} \bot & \text{if } \alpha \text{ and } \neg\alpha \text{ are L-labeled} \\ \top & \text{if } \alpha \text{ and } \neg\alpha \text{ are R-labeled} \\ \alpha & \text{if } \alpha \text{ is } L\text{-labeled and } \neg\alpha \text{ is R-labeled} \\ \neg\alpha & \text{if } \alpha \text{ is } R\text{-labeled and } \neg\alpha \text{ is L-labeled} \end{cases}$$



$$\text{where } \theta = \begin{cases} \bot & \text{if } \bot \text{ is L-labeled} \\ \top & \text{if } \bot \text{ is R-labeled} \end{cases}$$

Figure 3 Tableau rules with corresponding interpolant propagation rules

In particular, this implies that there is no recursive function taking as input two FO formulas ϕ, ψ and returning a Craig interpolant whenever $\models \phi \to \psi$.

As mentioned in Section 4, for decidable fragments of FO, the situation improves regarding the size and the effective computability of interpolants. More precisely, for present purposes, let a *decidable fragment* F of FO be a fragment that is decidable for entailment (i.e., it is decidable whether $\models \phi \to \psi$ for $\phi, \psi \in F$) and such that it is decidable whether a given FO formula belongs to F.

▶ **Proposition 6.5.** *Let F be any decidable fragment of* FO*.*

- *There is a recursive function that takes as input $\phi, \psi \in F$ and produces a Craig interpolant in* FO *whenever $\models \phi \to \psi$.*
- *If F has CIP, then there is a recursive function that takes as input $\phi, \psi \in F$ and produces a Craig interpolant $\vartheta \in F$ whenever $\models \phi \to \psi$.*

Proof. We first test the validity of the entailment $\models \phi \to \psi$. If it is not valid, we terminate, otherwise we search for an interpolant by brute force enumeration, enumerating all triples of strings (ϑ, p_1, p_2) where ϑ is a formula (in FO respectively in F) with $\mathrm{sig}(\vartheta) \subseteq \mathrm{sig}(\phi) \cap \mathrm{sig}(\psi)$ and $\mathrm{fvar}(\vartheta) \subseteq \mathrm{fvar}(\phi) \cap \mathrm{fvar}(\psi)$ and where p_1, p_2 are valid proofs (in some sound and complete proof system for FO) for $\phi \to \vartheta$ and $\vartheta \to \psi$. This is guaranteed to terminate, since we know that a Craig interpolant exists. ◀

The above proposition does not provide any concrete bounds on the size or complexity of computing interpolants. For various fragments, however, concrete bounds have been obtained. See for instance Chapter 4 for the case of modal logic and Chapter 13 for the quantifier-free fragment of FO. Similarly for GNFO, tight bounds are known for the size of Craig interpolants as well as an optimal algorithm for computing them [11], using the method of type elimination sequences (see Chapter 4).

We conclude with an open question. Recall that FO formulas do not in general have uniform interpolants.

▶ **Question 1.** *Consider the following problem:* given an FO-formula ϕ and a signature $\tau \subseteq \mathrm{sig}(\phi)$, does ϕ have a uniform interpolant w.r.t. τ? *This problem is easily seen to be undecidable and to be in Σ^0_3 (the third level of the arithmetical hierarchy). Is it Σ^0_3-complete?*

Acknowledgments

We thank Michael Benedikt, Sam van Gool, Jouko Väänänen, and Christoph Wernhard for helpful feedback on earlier drafts of this chapter.

References

1 Foto N. Afrati. Determinacy and query rewriting for conjunctive queries and views. *Theoretical Computer Science*, 412(11):1005–1021, 2011. doi:10.1016/j.tcs.2010.12.031.

2 Hajnal Andréka, István Németi, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27, 06 1998. doi:10.1023/A:1004275029985.

3 Carlos Areces, Patrick Blackburn, and Maarten Marx. Hybrid logics: Characterization, interpolation and complexity. *Journal of Symbolic Logic*, 66(3):977–1010, 2001.

4 Carlos Areces, Patrick Blackburn, and Maarten Marx. Repairing the interpolation theorem in quantified modal logic. *Annals of Pure and Applied Logic*, 124(1):287–299, 2003. doi:10.1016/S0168-0072(03)00059-9.

5 Vince Bárány, Michael Benedikt, and Balder ten Cate. Rewriting guarded negation queries. In *Proceedings of MFCS 2013*, pages 98–110, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

6 Vince Bárány, Balder ten Cate, and Luc Segoufin. Guarded negation. *Journal of the ACM*, 62(3):22.1–22:26, 2015.

7 Bartosz Bednarczyk and Reijo Jaakkola. Towards a model theory of ordered logics: Expressivity and interpolation. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2022.15.

8 Bartosz Bednarczyk, Daumantas Kojelis, and Ian Pratt-Hartmann. On the limits of decision: the adjacent fragment of first-order logic. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

9 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. Definability and interpolation within decidable fixpoint logics. *Logical Methods in Computer Science*, Volume 15, Issue 3, September 2019. doi:10.23638/LMCS-15(3:29)2019.

10 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. A step up in expressiveness of decidable fixpoint logics. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*. ACM, 2016.

11 Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. Effective interpolation and preservation in guarded logics. *ACM Trans. Comput. Logic*, 17(2), 2015. doi:10.1145/2814570.

12 Michael Benedikt, Julien Leblay, Balder ten Cate, and Efthymia Tsamoura. *Generating plans from proofs: the interpolation-based approach to query reformulation*. Synthesis Lectures on Data Management. Morgan & Claypool, 2016.

13 Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. Interpolation with decidable fixpoint logics. In *LICS*, pages 378–389, 2015. doi:10.1109/LICS.2015.43.

14 Michael Benedikt, Balder ten Cate, and Efthymia Tsamoura. Generating plans from proofs. *ACM Trans. Database Syst.*, 40(4):22:1–22:45, 2016. doi:10.1145/2847523.

15 Johan van Benthem. The many faces of interpolation. *Synthese*, 164(3):451–460, 2008.

16 Evert W. Beth. On Padoa's method in the theory of definition. *Indagationes Mathematicae*, 15:330–339, 1953.

17 Evert W. Beth. Semantic entailment and formal derivability. *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde*, 18(13):309–342, 1955.

Reprinted in Hintikka, Jaakko, ed. (1969). The Philosophy of Mathematics. Oxford University Press.

18 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

19 Balder ten Cate. Interpolation for extended modal languages. *The Journal of Symbolic Logic*, 70(1):223–234, 2005.

20 Balder ten Cate and Jesse Comer. Craig interpolation for decidable first-order fragments. In Naoki Kobayashi and James Worrell, editors, *Foundations of Software Science and Computation Structures*, pages 137–159, Cham, 2024. Springer Nature Switzerland.

21 Balder ten Cate, Willem Conradie, Maarten Marx, and Yde Venema. Definitorially complete description logics. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 79–89. AAAI Press, 2006.

22 Balder ten Cate and Luc Segoufin. Unary negation. *Logical Methods in Computer Science*, Volume 9, Issue 3, September 2013. doi:10.2168/LMCS-9(3:25)2013.

23 Stephen D. Comer. Classes without the amalgamation property. *Pacific Journal of Mathematics*, 28:309–318, 1969.

24 William Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *Journal of Symbolic Logic*, 22(3):250–268, 1957. doi:10.2307/2963593.

25 William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957. doi:10.2307/2963594.

26 William Craig. Elimination problems in logic: A brief history. *Synthese*, 164(3):321–332, 2008. doi:10.1007/s11229-008-9352-4.

27 William Craig. The road to two theorems of logic. *Synthese*, 164(3):333–339, 2008. doi:10.1007/s11229-008-9353-3.

28 Solomon Feferman. Lectures on proof theory. In M. H. Löb, editor, *Proceedings of the Summer School in Logic Leeds, 1967*, pages 1–107, Berlin, Heidelberg, 1968. Springer Berlin Heidelberg.

29 Solomon Feferman. Persistent and invariant formulas for outer extensions. *Compositio Mathematica*, 20:29–52, 1968.

30 Solomon Feferman. Harmonious logic: Craig's interpolation theorem and its descendants. *Synth.*, 164(3):341–357, 2008. doi:10.1007/s11229-008-9354-2.

31 Melvin Fitting. *First-order Logic and Automated Theorem Proving*. Graduate texts in computer science. Springer, 2nd edition, 1996.

32 Harvey Friedman. The complexity of explicit definitions. *Advances in Mathematics*, 20(1):18–29, 1976. doi:10.1016/0001-8708(76)90167-5.

33 Amélie Gheerbrant and Balder ten Cate. Craig interpolation for linear temporal languages. In Erich Grädel and Reinhard Kahle, editors, *Computer Science Logic, 23rd international Workshop, CSL 2009, 18th Annual Conference of the EACSL, Coimbra, Portugal, September 7-11, 2009. Proceedings*, volume 5771 of *Lecture Notes in Computer Science*, pages 287–301. Springer, 2009. doi:10.1007/978-3-642-04027-6_22.

34 Erich Graedel, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *Proceedings of LICS 1997*, page 306, 1997.

35 Yuri Gurevich. Toward logic tailored for computational complexity. In Egon Börger, Walter Oberschelp, Michael M. Richter, Brigitta Schinzel, and Wolfgang Thomas, editors, *Computation and Proof Theory*, pages 175–216, Berlin, Heidelberg, 1984. Springer Berlin Heidelberg.

36 Leon Henkin. An extension of the Craig-Lyndon interpolation theorem. *Journal of Symbolic Logic*, 28(3):201–216, 1963. doi:10.2307/2271066.

37 W. Hodges. *Model Theory*. Encyclopedia of mathematics and its applications. Cambridge University Press, 1992.

38 Eva Hoogland and Maarten Marx. Interpolation and definability in guarded fragments. *Studia Logica*, 70(3):373–409, 2002.

39 Ullrich Hustadt, R Schmidt, and L Georgieva. A survey of decidable first-order fragments and description logics. *Journal on Relational Methods in Computer Science*, 1:251–276, 01 2004.

40 Jean Christoph Jung and Frank Wolter. Living without Beth and Craig: Definitions and interpolants in the guarded and two-variable fragments. In *Proceedings of LICS 2021*, pages 1–14. IEEE Computer Society, jul 2021. doi:10.1109/LICS52264.2021.9470585.

41 Hans Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

42 Roger Lyndon. An interpolation theorem in the predicate calculus. *Pacific Journal of Mathematics*, 9(1):129 – 142, 1959. doi:pjm/1103039458.

43 Roger Lyndon. Properties preserved under homomorphism. *Pacific Journal of Mathematics*, 9:143–154, 1959.

44 Leopold Löwenheim. Über Möglichkeiten im Relativkalkül. *Mathematische Annalen*, 76:447–470, 1915.

45 Michael Mortimer. On languages with two variables. *Math. Log. Q.*, 21:135–140, 1975.

46 Martin Otto. An interpolation theorem. *The Bulletin of Symbolic Logic*, 6(4):447–462, 2000.

47 Alessandro Padoa. Essai d'une théorie algébrique des nombres entiers, précédé d'une introduction logique à une théorie déductive quelconque. In *Bibliothèque du Congrès international de philosophie, Paris, 1900*, volume 3, pages 309–365. Armand Colin, Paris, 1901. English translation in part in J. van Heijenoort (ed.), *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*, Harvard University Press, 1967, pp. 118–123.

48 Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *Logical Methods in Computer Science*, 12, 2016.

49 Andreas Potthoff. *Logische Klassifizierung regulärer Baumsprachen*. PhD thesis, University of Kiel, Germany, 1994.

50 Ian Pratt-Hartmann, Wiesław Szwast, and Lidia Tendera. The fluted fragment revisited. *The Journal of Symbolic Logic*, 84(3):1020–1048, 2019. doi:10.1017/jsl.2019.33.

51 Ian Pratt-Hartmann, Wiesław Szwast, and Lidia Tendera. Quine's fluted fragment is non-elementary. In Laurent Regnier and Jean-Marc Talbot, editors, *25th EACSL Annual Conference on Computer Science Logic*, 25th EACSL Annual Conference on Computer Science Logic (CSL 2016). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, June 2016. doi:10.4230/LIPIcs.CSL.2016.39.

52 William Purdy. Decidability of fluted logic with identity. *Notre Dame Journal of Formal Logic*, 37(1):84 – 104, 1996. doi:10.1305/ndjfl/1040067318.

53 William Purdy. Fluted formulas and the limits of decidability. *The Journal of Symbolic Logic*, 61(2):608–620, 1996. doi:10.2307/2275678.

54 William Purdy. Quine's 'limits of decision'. *The Journal of Symbolic Logic*, 64(4):1439–1466, 1999. doi:10.2307/2586789.

55 William Purdy. Complexity and nicety of fluted logic. *Studia Logica*, 71:177–198, 2002.

56 Willard V. Quine. On the limits of decision. *14th International Congress for Philosophy*, 3:57–62, 1969.

57 Abraham Robinson. A result on consistency and its application to the theory of definition. *Journal of Symbolic Logic*, 25(2):174–174, 1960. doi:10.2307/2964240.

58 Nicole Schweikardt. A short tutorial on order-invariant first-order logic. In Andrei A. Bulatov and Arseny M. Shur, editors, *Computer Science – Theory and Applications*, pages 112–126, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

59 Joseph R. Shoenfield. *Recursion Theory*. Lecture Notes in Logic. Cambridge University Press, 2017.

60 Thoralf Skolem. *Logisch-Kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze nebst einem Theorem über dichte Mengen*. I. Matematisk-naturvidenskabelig Klasse 4, 1-36. Videnskapsselskapet Skrifter, 1920.

61 Jacques Stern. A new look at the interpolation problem. *The Journal of Symbolic Logic*, 40(1):1–13, 1975.

62 Jouko Väänänen. The Craig interpolation theorem in abstract model theory. *Synthese*, 164(3):401–420, 2008.

63 Moshe Y. Vardi. Why is modal logic so robustly decidable? In N. Immerman and Ph. G. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS*, pages 149–183, 1996. doi:10.1090/dimacs/031/05.

Chapter 3

Interpolation in Non-Classical Logics

Wesley Fussner
Institute of Computer Science of the Czech Academy of Sciences, Czechia

Abstract

This chapter surveys some of the main results on interpolation in several of the most prominent families of non-classical logics. Special attention is given to the distinction between the two most commonly studied variants of interpolation—namely, Craig interpolation and deductive interpolation. Our discussion focuses primarily on how these properties present in families of logical systems taken as a whole, particularly those comprising all axiomatic extensions of any of several notable non-classical logics. We consider a range of important examples: superintuitionistic and modal logics, fuzzy logics, paraconsistent logics, relevant logics, and substructural logics.

Contents

1 Introduction 70
2 Consequence, Interpolation, and Deduction Theorems 72
2.1 Consequence Relations 72
2.2 Variants of Interpolation 73
2.3 Deduction Theorems and Relations Among Interpolation Variants 75
3 Fundamental Examples 77
3.1 Superintuitionistic Logics 77
3.2 Modal Logics 79
3.2.1 Normal Modal Logics Generally 80
3.2.2 Extensions of `S4` 80
3.2.3 Non-Normal Modal Logics 81
3.3 Paraconsistent Logics 82
3.3.1 The Logic of Paradox 82
3.3.2 Johansson's Minimal Logic 82
3.3.3 Nelson's Constructive Logic With Strong Negation 83



© Wesley Fussner;
licensed under Creative Commons License CC BY 4.0
In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 3, pages 69–100. Ubiquity Press, 2026.
`https://www.doi.org/10.5334/bdg.c`

3.4 Fuzzy Logics 84
3.4.1 Łukasiewicz's Logic and Its Extensions 84
3.4.2 Hájek's Basic Fuzzy Logic and Its Extensions 86
3.4.3 Monoidal T-norm Based Logics 88
3.5 Relevant Logics 88
3.6 Substructural Logics 91
3.6.1 Interpolation and the Basic Structural Rules 92
3.6.2 Logics with Exchange 93
3.6.3 Logics Without Exchange 94
Acknowledgments **95**
References **95**

1 Introduction

In this chapter, we survey the landscape of interpolation in non-classical logics. We will see an in-depth case study from the non-classical world—namely the modal logic K—in Chapter 4. Here we will see many other logical systems in the vicinity of K, as well as a plethora of other, more distant logics.

One theme that becomes apparent throughout our discussion is that non-classical logics present many fine distinctions that do not appear for classical propositional logic CPL, or even for many well-behaved non-classical systems such as K. The most salient of these is that the relationship between interpolation for the consequence relation of a logic and interpolation for the primitive implication connective of that logic (if any) often breaks down, leading to distinct interpolation properties that transparently coincide for CPL. Our focus will be on the two most prominent variants of interpolation, here called the Craig interpolation property and the deductive interpolation property, but we will also consider a number of other interpolation properties that have proven important in particular logical environments. We will also discuss links between several interpolation properties, most often given by different kinds of deduction theorems.

This chapter covers a lot of territory, ranging over an incredibly diverse collection of logical systems that have arisen from many different motivations. To make sense of this diversity, we will frame our discussion in the language of (Tarskian) consequence relations, and we will begin by recalling some of the basics of these as well as several related notions. Due the scope of the topic, many important aspects of interpolation in non-classical logics will be omitted for lack of space. Our discussion will not include several prominent variants of interpolation, such as uniform interpolation (see Chapter 9) and Lyndon interpolation (see Chapter 5), and will for the most part avoid discussion of the connection between interpolation and Beth definability, leaving coverage of these topics to other chapters of this volume.

We will also omit discussion of the role the underlying language plays in shaping interpolation. The importance of this topic is evident already in the case of variants of CPL itself: If

$\mathtt{CPL}$ is formulated with constants for truth and falsity, then $\mathtt{CPL}$ enjoys the Craig interpolation theorem in the following form:

> Whenever $\alpha, \beta \in \mathsf{Fm}_{\mathtt{CPL}}$ and $\vdash_{\mathtt{CPL}} \alpha \to \beta$, then there exists $\delta \in \mathsf{Fm}_{\mathtt{CPL}}$ such that $\vdash_{\mathtt{CPL}} \alpha \to \delta$, $\vdash_{\mathtt{CPL}} \delta \to \beta$, and $\mathsf{Var}(\delta) \subseteq \mathsf{Var}(\alpha) \cap \mathsf{Var}(\beta)$.

However, the truth constants themselves may supply interpolants for some theorems, like $(p \wedge \neg p) \to (q \vee \neg q)$. When truth constants are missing from the language, Craig interpolation therefore must instead by formulated to exclude these kinds of examples:

> Whenever $\alpha, \beta \in \mathsf{Fm}_{\mathtt{CPL}}$, $\nvdash_{\mathtt{CPL}} \neg\alpha$, $\nvdash_{\mathtt{CPL}} \beta$, and $\vdash_{\mathtt{CPL}} \alpha \to \beta$, then there exists $\delta \in \mathsf{Fm}_{\mathtt{CPL}}$ such that $\vdash_{\mathtt{CPL}} \alpha \to \delta$, $\vdash_{\mathtt{CPL}} \delta \to \beta$, and $\mathsf{Var}(\delta) \subseteq \mathsf{Var}(\alpha) \cap \mathsf{Var}(\beta)$.

The variants of $\mathtt{CPL}$ with and without truth constants are best understood as *different logics* when it comes to interpolation, despite being notational variations of the same system. The lesson here is that precise formulations of the systems in question—and of interpolation itself—can have quite significant ramifications. Indeed, a logic that is lacking interpolation (in some specified form) can often be made more expressive with a mild expansion of the underlying language in a way that restores interpolation. This phenomenon is poorly understood in the general setting of non-classical logics (compare, e.g., [45, Theorem 14] and [64]), and the possible variations in language are virtually endless. We will thus mostly exclude this important topic from our discussion here.

Most of this chapter consists of a tour of interpolation in particular families of logical systems, usually organized by recalling a few of the most naturally defined logics in those families and then discussing what is known about interpolation in the axiomatic extensions of these. Our tour of these systems touches only on the highlights, focusing on a few of the most salient results and making no effort to give a comprehensive treatment. Many of the theorems we summarize are major results whose proofs are well beyond the scope of this chapter. In most cases, we cannot do justice to even a sketch of these proofs in this lone chapter, so proofs will be entirely omitted. However, we typically indicate the overall proof methodology pursued in a given study. There are, for the most part, three main lines of argumentation: Proof-theoretic investigations anchored in Maehara's method (see Chapter 5 for a full discussion), algebraic methods focused on the link between interpolation and various kinds of amalgamation (discussed extensively in Chapter 8), and model-theoretic studies that exploit quantifier-elimination theorems. We give no more than allusions to the techniques at play here, leaving the discussion of methods to other chapters. However, we provide an extensive bibliography in order to point the reader to resources where the main technical work is carried out.

Our overall organizational motif—that is, considering prominent logics in turn and, for each of these, indicating what is known about interpolation in axiomatic extensions—was selected for four reasons. Firstly, and perhaps most obviously, this scheme provides some principled means of systematizing the huge amount of information available in the literature,

and in particular giving the reader some idea of what classifications are known. Secondly, the chosen organizational scheme gives a coarse measure of *how robust* interpolation is within certain logical environments: As we will see, certain logics (e.g., intuitionistic logic, Section 3.1) have very few axiomatic extensions with interpolation, whereas other closely related logics (e.g., $\mathtt{SemFL}^{+}_{\mathtt{cm}}$, Section 3.6.3) have many. The third reason we adopt our chosen organizational scheme is also comparative: For some logics, we have an exhaustive classification of axiomatic extensions with various kinds of interpolation (again, intuitionistic logic, Section 3.1), while in other, seemingly analogous cases our techniques fail us (e.g., Johansson's minimal logic, Section 3.3.2). Side-by-side comparisons of how interpolation operates in the axiomatic extensions of given logics gives some indication of what our methods are capable of, as well as points to many open questions. We explicitly identify several such open questions along the way, hoping to inspire readers to further advance the lines of research articulated here. Fourthly and finally, the logics enjoying interpolation are especially *natural* among their cousins in a given family, as determined by a range of different philosophical motivations; see Chapter 16. The organizational scheme adopted here is thus, we hope, a useful device for practitioners arriving at interpolation via these motivations, though we don't discuss them further here.

2 Consequence, Interpolation, and Deduction Theorems

Our discussion of interpolation in non-classical logics will be carried out within the context of (Tarskian) consequence relations, which provide a framework allowing us to compare a range of differing logical systems. After discussing the basics of consequence relations, we will introduce a few of the most relevant interpolation notions for this chapter, and subsequently discuss deduction theorems and how they relate the variants of interpolation we will consider. For background on consequence relations and the other preliminary material discussed in this section, please consult the monographs [16, 23].

2.1 Consequence Relations

As is most common in the literature, we will always consider propositional logics that are defined over a denumerable set Var of propositional variables and a language $\mathcal{L}$ consisting of finitely many logical connectives, each of finite (possibly 0) arity. We denote by $\mathsf{Fm}_{\mathcal{L}}$ the collection of all formulas built from Var using the connectives from $\mathcal{L}$. Whenever $\alpha \in \mathsf{Fm}_{\mathcal{L}}$, we denote the set of variables appearing in α by $\mathsf{Var}(\alpha)$. If Γ is a set of formulas, we also write $\mathsf{Var}(\Gamma) = \bigcup\{\mathsf{Var}(\alpha) \mid \alpha \in \Gamma\}$.

A *consequence relation* over $\mathcal{L}$ is a relation $\vdash$ from $\mathcal{P}(\mathsf{Fm}_{\mathcal{L}})$ to $\mathsf{Fm}_{\mathcal{L}}$ satisfying the following two conditions, where $\Gamma \cup \Pi \cup \{\alpha\} \subseteq \mathsf{Fm}_{\mathcal{L}}$:

- Reflexivity: If $\alpha \in \Gamma$, then $\Gamma \vdash \alpha$.
- Transitivity: If $\Gamma \vdash \alpha$ and $\Pi \vdash \beta$ for every $\beta \in \Gamma$, then $\Pi \vdash \alpha$.

It is immediate that every consequence relation also satisfies monotonicity: If $\Gamma \vdash \alpha$ and $\Gamma \subseteq \Pi$, then $\Pi \vdash \alpha$.

The consequence relations that we consider in this chapter satisfy two further properties:

- Structurality: If $\Gamma \vdash \alpha$, then $\sigma[\Gamma] \vdash \sigma(\alpha)$ for every substitution σ.
- Finitarity: If $\Gamma \vdash \alpha$, then there exists a finite subset $\Gamma' \subseteq \Gamma$ such that $\Gamma' \vdash \alpha$.

If $\Gamma, \Pi \subseteq \mathsf{Fm}_{\mathcal{L}}$ and $\vdash$ is a consequence relation over $\mathcal{L}$, then we write $\Gamma \vdash \Pi$ if $\Gamma \vdash \alpha$ for every $\alpha \in \Pi$.

Aside from our few references to first-order logic in this chapter, we will take *logics* (over the language $\mathcal{L}$) to be pairs $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$, where $\mathcal{L}$ is some finitary language and $\vdash_{\mathtt{L}}$ is a finitary, structural consequence relation over $\mathcal{L}$. If $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ is a logic, a formula $\alpha \in \mathsf{Fm}_{\mathcal{L}}$ is called a *theorem* of $\mathtt{L}$ provided that $\emptyset \vdash_{\mathtt{L}} \alpha$, which we usually abbreviate by $\vdash_{\mathtt{L}} \alpha$. By an *extension* of a logic $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ we mean any logic of the form $(\mathcal{L}, \vdash'_{\mathtt{L}})$, where $\vdash'_{\mathtt{L}}$ contains $\vdash_{\mathtt{L}}$. We reserve the term *expansion* for enlargements of the language and will not use 'extension' for this. If $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ is a logic, a *fragment* of $\mathtt{L}$ is a logic $(\mathcal{L}', \vdash_{\mathtt{L}})$ where $\mathcal{L}' \subseteq \mathcal{L}$.

There is a long tradition of studying a given logic $\mathtt{L}$ by looking at its extensions, particularly those that arise by adding axioms (i.e., as opposed to arbitrary rules). If $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ is a logic, then a logic $\mathtt{L}' = (\mathcal{L}, \vdash_{\mathtt{L}'})$ is called an *axiomatic extension* of $\mathtt{L}$ if there is a subset $\Pi \subseteq \mathsf{Fm}_{\mathcal{L}}$ that is closed under substitutions such that, for any $\Gamma \cup \{\alpha\} \subseteq \mathsf{Fm}_{\mathcal{L}}$,

$$\Gamma \vdash_{\mathtt{L}'} \alpha \iff \Gamma, \Pi \vdash_{\mathtt{L}} \alpha.$$

The only axiomatic extensions of classical propositional logic CPL are CPL itself and the trivial logic consisting of all formulas in the language. However, for non-classical logics, there are often numerous proper axiomatic extensions. These always form a lattice when ordered by inclusion of the associated consequence relations. The structure of the lattice of axiomatic extensions of a given logic casts considerable light on the logic itself. In much of this chapter, we will examine, for many prominent non-classical logics $\mathtt{L}$, which of the axiomatic extensions of $\mathtt{L}$ have various interpolation properties. This sort of inquiry has been carried out in a wide range of logical contexts, and illuminates how interpolation operates in such contexts. Of particular benefit of the wide view taken in this chapter, this mode of investigation also allows for the comparison of interpolation phenomena across varying contexts and thereby gives us one perspective on how interpolation operates on a 'big picture' level.

2.2 Variants of Interpolation

There are a huge number of interpolation notions that have been considered in the non-classical logic literature, many of which are specialized to particular contexts. In this chapter, we will primarily consider two such notions that make sense for most well-studied logical systems. The first of them, and probably the most familiar, is the *Craig interpolation property* (or CIP). It can be formulated for any logic $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ that has an implication-like connective $\rightarrow$ in its language, and may be stated as follows:

Whenever $\alpha, \beta \in \mathsf{Fm}_{\mathcal{L}}$ and $\vdash_{\mathtt{L}} \alpha \to \beta$, then there exists $\delta \in \mathsf{Fm}_{\mathcal{L}}$, called a *Craig interpolant*, such that $\vdash_{\mathtt{L}} \alpha \to \delta$, $\vdash_{\mathtt{L}} \delta \to \beta$, and $\mathsf{Var}(\delta) \subseteq \mathsf{Var}(\alpha) \cap \mathsf{Var}(\beta)$. (CIP)

Of course, here we are contemplating that there is a particular implication-like connective $\to$ that is primitive in the language, and that there is only one such implication. This need not be the case: Some logics have multiple primitive implication connectives, and some have definable implications that we may want to consider. Indeed, one might speak of the CIP for some two-variable term $I(p, q)$ that is definable in the language of $\mathcal{L}$ but does not appear as a primitive connective in $\mathcal{L}$. We do not dwell on this point, but note that many logics have a form of the CIP for some defined implication. For instance, there are continuum-many axiomatic extensions of linear logic that have a version of the CIP with respect to the non-standard implication $I(p, q) = !p \to !q$; see [28, Proposition 3.16]. For most of the logics we will consider, it will be clear from context which implication we are referring to when we discuss the CIP.

The preceding comments underscore some of the motivations for having an implication-free formulation of interpolation. The most prominent such formulation is given by the *deductive interpolation property* (or DIP), which for a logic $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ provides that:

Whenever $\alpha, \beta \in \mathsf{Fm}_{\mathcal{L}}$ and $\alpha \vdash_{\mathtt{L}} \beta$, then there exists $\delta \in \mathsf{Fm}_{\mathcal{L}}$, called a *deductive interpolant*, such that $\alpha \vdash_{\mathtt{L}} \delta$, $\delta \vdash_{\mathtt{L}} \beta$, and $\mathsf{Var}(\delta) \subseteq \mathsf{Var}(\alpha) \cap \mathsf{Var}(\beta)$. (DIP)

Deductive interpolation makes sense for any logic, even those that have no implication-like connective at all, and is not otherwise bound up with the language of the particular logical system in question. It is thus a good notion for comparing interpolation phenomena across rather different logical contexts. Note that the DIP has sometimes been referred to by other names in the literature: Maksimova sometimes calls it the interpolation principle of deducibility (see [50]), or the IPD for short, and Madarász calls it the turnstile Craig interpolation property (see [47]). Sometimes causing confusion, the DIP is occasionally simply called the Craig interpolation property.

The relationship between the CIP (for some given implication) and the DIP is quite complicated. A given logic may have either one, both, or neither of these properties. We will return to the relationship between these two notions shortly in Section 2.3.

Several stronger variants of the CIP and the DIP are commonly studied in the literature. These mostly fall into two categories. First, sometimes additional constraints are placed on the variables appearing in an interpolant, as in Lyndon interpolation (see Chapter 5). Second, sometimes additional constraints are placed on the interpolant itself, such as demanding that it is in some sense 'minimal' or 'maximal', as in uniform interpolation (see Chapter 9). Other constraints can be placed on the language appearing in the interpolant, such as demanding that certain connectives appear in common; see, e.g., [8]. For the most part, we will leave these topics to subsequent chapters, focusing on the CIP and DIP in their most basic formulation.

However, we will touch on a few other versions of interpolation that are important for certain logics, two of which we mention now.

First, the following stronger kind of interpolation is sometimes considered in the literature. Here we will call it the *Maehara interpolation property* (or MIP), but elsewhere it is sometimes called the strong deductive interpolation property or the SDIP. We say that a logic $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ has the Maehara interpolation property provided that:

> Whenever $\Gamma \cup \Pi \cup \{\alpha\} \subseteq \mathsf{Fm}_{\mathcal{L}}$ and $\Gamma, \Pi \vdash_{\mathtt{L}} \alpha$, then there exists $\delta \in \mathsf{Fm}_{\mathcal{L}}$ such that $\Gamma \vdash_{\mathtt{L}} \delta$ and $\delta, \Pi \vdash_{\mathtt{L}} \alpha$, and also $\mathsf{Var}(\delta) \subseteq \mathsf{Var}(\Gamma) \cap \mathsf{Var}(\Pi \cup \{\alpha\})$. (MIP)

Obviously, the MIP implies the DIP for any logic.

Second, in certain logics for which the DIP fails, it is sometimes useful to consider a slightly weaker variant of the MIP. Assume that $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ is a logic with a falsity constant $\bot$ appearing in its language. We say that $\mathtt{L}$ has the *weak interpolation property* (or WIP) provided that:

> If $\alpha, \beta \in \mathsf{Fm}_{\mathcal{L}}$ and $\alpha, \beta \vdash_{\mathtt{L}} \bot$, then there is a formula δ such that $\alpha \vdash_{\mathtt{L}} \delta$, $\delta, \beta \vdash_{\mathtt{L}} \bot$, and $\mathsf{Var}(\delta) \subseteq \mathsf{Var}(\alpha) \cap \mathsf{Var}(\beta)$. (WIP)

2.3 Deduction Theorems and Relations Among Interpolation Variants

Deduction theorems link deducibility to theoremhood, and thus unsurprisingly play a significant role in connecting the CIP and the DIP. Like interpolation, deduction theorems come in a range of different forms that are often specialized to particular logical contexts. The most familiar of these are *classical deduction theorems*, which mirror the deduction theorem found in CPL. We say that a logic $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ has a classical deduction theorem provided that, for any $\Gamma \cup \{\alpha, \beta\} \subseteq \mathsf{Fm}_{\mathcal{L}}$,

$$\Gamma, \alpha \vdash_{\mathtt{L}} \beta \iff \Gamma \vdash_{\mathtt{L}} \alpha \to \beta.$$

One may readily show that if $\mathtt{L}$ has a classical deduction theorem, then $\mathtt{L}$ has the CIP if and only if $\mathtt{L}$ has the DIP. This is true, for example, for CPL as well as propositional intuitionistic logic (see Section 3.1 below) and for many other logical systems.

Moving to a slightly higher level of abstraction, we say that $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ has an *explicit deduction theorem* if there is a formula $I(p, q)$ in $\mathcal{L}$ in at most two variables p, q such that for any $\Gamma \cup \{\alpha, \beta\} \subseteq \mathsf{Fm}_{\mathcal{L}}$,

$$\Gamma, \alpha \vdash_{\mathtt{L}} \beta \iff \Gamma \vdash_{\mathtt{L}} I(\alpha, \beta).$$

The classical deduction theorem takes $I(p, q)$ to be $p \to q$. On the other hand, some logics have an explicit deduction theorem but not a classical one. For instance, the global consequence relation of the modal logic S4 (see Section 3.2) satisfies the following explicit deduction theorem:

$$\Gamma, \alpha \vdash_{\mathtt{L}} \beta \iff \Gamma \vdash_{\mathtt{L}} \Box\alpha \to \beta.$$

Actually, this holds for any axiomatic extension of S4 and this shows that having an explicit deduction theorem is insufficient to guarantee that the DIP implies the CIP for some given logic (as we will see in Section 3.2, there are extensions of S4 with the DIP but not the CIP).

Moving up another layer of abstraction, we say that a logic $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ has a *local deduction theorem* if there is a set Λ of formulas in at most two variables such that for any $\Gamma \cup \{\alpha, \beta\} \subseteq \mathsf{Fm}_{\mathcal{L}}$,

$$\Gamma, \alpha \vdash_{\mathtt{L}} \beta \iff \Gamma \vdash_{\mathtt{L}} I(\alpha, \beta) \text{ for some } I \in \Lambda.^{1}$$

The global consequence relation of the normal modal logic K4 (see Chapter 4 and Section 3.2 below) has a local deduction theorem but not an explicit one. Many of the logics we consider have a local deduction theorem of a particularly nice form: Each of the formulas $I(p, q) \in \Lambda$ has the form $t(p) \to q$ for some formula t in one variable. We will call these *implicative local deduction theorems*, though this is not a standard term in the literature.

The relevance of implicative local deduction theorems to our discussion is summarized in the following two results. They are both folklore but mostly only proven in special cases (see, e.g., [43]), so we include proofs for completeness.

▶ **Proposition 1.** *Let* $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ *be a logic with an implication connective* $\to$ *in its language, and assume that:*

1. L *satisfies modus ponens, i.e.,* $\alpha, \alpha \to \beta \vdash_{\mathtt{L}} \beta$ *for any* $\alpha, \beta \in \mathsf{Fm}_{\mathcal{L}}$.
2. L *has an implicative local deduction theorem given by a set of formulas* Λ.

Then if L *has the CIP, then* L *has the DIP.*

Proof. Suppose that $\alpha \vdash_{\mathtt{L}} \beta$. Then there exists $t(p) \to q \in \Lambda$ such that $t(\alpha) \to \beta$ is a theorem of L, so by the CIP there exists $\delta \in \mathsf{Fm}_{\mathcal{L}}$ such that $\vdash_{\mathtt{L}} t(\alpha) \to \delta$, $\vdash_{\mathtt{L}} \delta \to \beta$, and $\mathsf{Var}(\delta) \subseteq \mathsf{Var}(\alpha) \cap \mathsf{Var}(\beta)$. By the deduction theorem, $\vdash_{\mathtt{L}} t(\alpha) \to \delta$ implies $\alpha \vdash_{\mathtt{L}} \delta$. On the other hand, from modus ponens, $\vdash_{\mathtt{L}} \delta \to \beta$ implies $\delta \vdash_{\mathtt{L}} \beta$. ◀

Most well-studied logics that have local deduction theorems actually have implicative local deduction theorems in the above sense, and we will sometimes just say that these logics have local deduction theorems in our discussion. With the exception of logic of paradox, discussed in Section 3.3, all of the logics we consider in this chapter validate modus ponens.

In some exceptional contexts, the DIP implies the CIP as well. This is the case, for example, in the presence of a classical deduction theorem, but also in certain cases when an explicit deduction theorem is absent (such as the case of the relevant logic R-mingle, as formulated with truth constants; see [57]). The precise conditions under which this occurs remain a mystery.

---

[1] Actually, in abstract algebraic logic, often entire sets of formulas are taken to stand in place of individual formulas acting as an implication-like connective. The practicality of this is uncertain; see the discussion on uniterm vs. multiterm deduction detachment theorems at [23, p. 166].

▶ **Open Question 2.** *Under what circumstances does the DIP imply the CIP?*

The next proposition concerns logics that additionally have a (not necessarily primitive) conjunction-like connective. Given a logic $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$, we say that a formula $C(p, q)$ in at most two variables is a *conjunction* for $\mathtt{L}$ if for every set of formulas $\Gamma \cup \{\alpha, \beta, \gamma\}$,

$$\Gamma, \alpha, \beta \vdash_{\mathtt{L}} \gamma \iff \Gamma, C(\alpha, \beta) \vdash_{\mathtt{L}} \gamma.$$

Of course, in many familiar logics, C may be taken to be the primitive conjunction connective in the language, but the above formulation makes sense even for logics without such a primitive conjunction connective. When $\mathtt{L}$ has a conjunction, we say that $\mathtt{L}$ is a *conjunctive* logic.

▶ **Proposition 3.** *Let $\mathtt{L} = (\mathcal{L}, \vdash_{\mathtt{L}})$ be a logic with an implication connective $\to$ in its language, and assume that:*

1. *$\mathtt{L}$ satisfies modus ponens, i.e., $\alpha, \alpha \to \beta \vdash_{\mathtt{L}} \beta$ for any $\alpha, \beta \in \mathsf{Fm}_{\mathcal{L}}$.*
2. *$\mathtt{L}$ has an implicative local deduction theorem given by a set of formulas Λ.*
3. *$\mathtt{L}$ is conjunctive.*

Then $\mathtt{L}$ has the DIP if and only if $\mathtt{L}$ has the MIP.

Proof. It is clear that the MIP implies the DIP. For the converse, suppose that $\Gamma \cup \Pi \cup \{\alpha\} \subseteq \mathsf{Fm}_{\mathcal{L}}$ and that $\Gamma, \Pi \vdash_{\mathtt{L}} \alpha$. By finitarity, there exist finite subsets $\Gamma' \subseteq \Gamma$ and $\Pi' \subseteq \Pi$ such that $\Gamma', \Pi' \vdash_{\mathtt{L}} \alpha$. Since $\mathtt{L}$ is conjunctive, by iterating C there are formulas $C(\Gamma'), C(\Pi')$ so that $C(\Gamma'), C(\Pi') \vdash_{\mathtt{L}} \alpha$. By the implicative local deduction theorem, there exists a formula $t(p)$ in at most one free variable such that $C(\Gamma') \vdash_{\mathtt{L}} t(C(\Pi')) \to \alpha$, and by the DIP there exists a formula δ such that

$$\mathsf{Var}(\delta) \subseteq \mathsf{Var}(C(\Gamma')) \cap \mathsf{Var}(t(C(\Pi')) \to \alpha) = \mathsf{Var}(\Gamma') \cup \mathsf{Var}(\Pi' \cup \{\alpha\}) \subseteq \mathsf{Var}(\Gamma) \cap \mathsf{Var}(\Pi \cup \{\alpha\})$$

and $C(\Gamma') \vdash_{\mathtt{L}} \delta$ and $\delta \vdash t(C(\Pi')) \to \alpha$ hold. Applying the deduction theorem and conjunctivity again gives $\Gamma' \vdash_{\mathtt{L}} \delta$ and $\delta, \Pi' \vdash_{\mathtt{L}} \alpha$, so $\Gamma \vdash_{\mathtt{L}} \delta$ and $\delta, \Pi \vdash_{\mathtt{L}} \alpha$ hold by monotonicity. ◀

3 Fundamental Examples

The remainder of this chapter is devoted to a survey of what is known about interpolation in several prominent families of non-classical logics. The account offered here is neither comprehensive nor representative, but rather intended as a pointer to a few of the landmarks in the vast literature on interpolation in non-classical logics. The proofs of the many of the results conveyed here are extremely involved, and are omitted. For some of the high-level ideas used in these proofs, please consult Chapter 4, Chapter 5, Chapter 8, and Chapter 9.

3.1 Superintuitionistic Logics

We begin our discussion of fundamental examples with propositional intuitionistic logic IPC and its axiomatic extensions, which are often called *superintuitionistic logics* or *intermediate*

logics. The logic IPC is usually formulated in the basic language $\mathcal{L}_{\mathsf{bas}} = \{\wedge, \vee, \to, \bot, \top\}$, where $\neg\alpha$ and $\alpha \leftrightarrow \beta$ are introduced as abbreviations for $\alpha \to \bot$ and $(\alpha \to \beta) \wedge (\beta \to \alpha)$, respectively.

The consequence relation of IPC can be specified in a number ways. Syntactically, it may be presented by a Hilbert-style calculus (see [12, Section 2.6]) or an analytic sequent calculus such as Gentzen's LJ (see [35] and Chapter 5). Semantically, $\vdash_{\mathtt{IPC}}$ can be captured in terms of the intuitionistic Kripke frames or Heyting algebras, which respectively provide relational and equivalent algebraic semantics for IPC. The reader may consult [12] for a detailed account.

Crucially, each axiomatic extension L of IPC enjoys a classical deduction theorem

$$\Gamma, \alpha \vdash_{\mathtt{L}} \beta \iff \Gamma \vdash_{\mathtt{L}} \alpha \to \beta,$$

so the CIP for $\to$ and the DIP coincide for all of these logics, and also coincide with the Maehara interpolation property MIP.

The first result on interpolation for intuitionistic logic is due to Schütte, who established (the predicate version of) the Craig interpolation property for intuitionistic predicate logic in [78]. Schütte's methods were proof theoretic, and were subsequently sharpened by Nagashima [65], who also used proof-theoretic tools. Gabbay later provided a semantic proof of the Craig interpolation property for intuitionistic predicate logic using Kripke models; see [31].

Returning to the propositional setting, Maksimova studied interpolation for IPC and its extensions in the 1970s, leading to an exhaustive description of axiomatic extensions of IPC with the CIP and thus, equivalently, the DIP and the MIP. This is summarized in the following theorem. For context, note that by Jankov's theorem, there are continuum-many axiomatic extensions of IPC.

▶ **Theorem 4** ([48]). *There are exactly eight superintuitionistic logics with the CIP or, equivalently, with the DIP or MIP. These eight logics are:*

1. IPC *itself.*
2. *Classical Propositional Logic* CPL*, axiomatized relative to* IPC *by* $\alpha \vee \neg\alpha$.
3. *Gödel-Dummett Logic* LC*, axiomatized relative to* IPC *by* $(\alpha \to \beta) \vee (\beta \to \alpha)$.
4. *Jankov's Logic* KC*, axiomatized relative to* IPC *by* $\neg\alpha \vee \neg\neg\alpha$.
5. *The logic* $\mathtt{LP}_2$*, axiomatized relative to* IPC *by* $\alpha \vee (\alpha \to (\beta \vee \neg\beta))$.
6. *The logic axiomatized relative to* $\mathtt{LP}_2$ *by* $(\alpha \to \beta) \vee (\beta \to \alpha) \vee (\alpha \leftrightarrow \neg\beta)$.
7. *The logic axiomatized relative to* $\mathtt{LP}_2$ *by* $\neg\alpha \vee \neg\neg\alpha$.
8. *The trivial logic consisting of all formulas.*

Maksimova's proof of Theorem 4 depends on the equivalence of CIP for any given axiomatic extension L of IPC with the amalgamation property (or, equivalently for varieties of Heyting algebras, the superamalgamation property) for the associated class of Heyting algebras modeling L; see Chapter 8. A full account of these results can also be found in the monograph of Gabbay and Maksimova [33, Chapter 6].

Theorem 4 also has relevance to positive intuitionistic propositional logic $\mathtt{IPC}^+$, i.e., the fragment of IPC without $\bot$.

▶ **Proposition 5** ([48]). *There are exactly four axiomatic extensions of* $\mathtt{IPC}^+$ *with the CIP (or, equivalently, with the DIP or MIP). These four logics are:*

1. $\mathtt{IPC}^+$ *itself.*
2. *The positive fragment of* CPL.
3. *The positive fragment of the Gödel-Dummett Logic* LC.
4. *The trivial logic consisting of all formulas.*

Further studies of interpolation in IPC have sharpened it in a number of ways. Of these, work on uniform interpolation in IPC and its extensions, began by Pitts in [72], has proven especially prominent. This is further discussed in Chapter 9. Interpolation and Beth definability for several fragments of intuitionistic logic have recently been studied in [19].

3.2 Modal Logics

In their most common formulation, modal logics expand CPL by additional unary connectives $\Box$ and $\Diamond$, respectively intended to capture various kinds of 'necessity' and 'possibility'. Modal logics are hence typically formulated in the language $\mathcal{L}_{\mathsf{mod}} = \{\wedge, \vee, \rightarrow, \bot, \top, \Box, \Diamond\}$ that expands the basic language $\mathcal{L}_{\mathsf{bas}}$ of CPL and IPC by symbols for these unary connectives. Considered as an expansion of CPL, modal logics typically validate the equivalence of $\Box\alpha$ and $\neg\Diamond\neg\alpha$, so in fact only one of the connectives $\Box$ or $\Diamond$ need be taken as primitive. However, for modal logics that expand non-classical base logics—for example, intuitionistic modal logics—the connectives $\Box$ and $\Diamond$ need not be interdefinable; see [79]. Of course, one may also include several modal operators of the $\Box$ type and their dual $\Diamond$ connectives, obtaining multimodal logics. Much can be said about interpolation in the multimodal setting; see [32, Section 4.5]. Here, however, we confine our attention to the unimodal case.

In Chapter 4, we have already seen the basic modal logic K, which can be specified either proof-theoretically or as the logic of all (modal) Kripke frames. Actually, K is really *two* logics, depending on whether one takes its local consequence relation $\vdash^{\ell}_{\mathtt{K}}$ or its global consequence relation $\vdash^{g}_{\mathtt{K}}$; see [12] for relevant definitions. Because deductive interpolation for the local consequence relation is tantamount to Craig interpolation, we will usually just consider the global consequence relation and refer to the DIP and the CIP, denoting $\vdash^{g}_{\mathtt{L}}$ simply as $\vdash_{\mathtt{L}}$ for axiomatic extensions of $\mathtt{K} = (\mathcal{L}_{\mathsf{mod}}, \vdash^{g}_{\mathtt{K}})$. Axiomatic extensions of K are usually called *normal* modal logics.

Every normal modal logic L satisfies a (implicative) local deduction theorem of the form

$$\Gamma, \alpha \vdash_{\mathtt{L}} \beta \iff (\alpha \wedge \Box\alpha \wedge \Box^2\alpha \wedge \ldots \wedge \Box^n\alpha) \rightarrow \beta \text{ for some } n \in \mathbb{N},$$

where $\Box^k\alpha$ abbreviates the expression $\Box\Box\ldots\Box\alpha$, where $\Box$ occurs k times. Thus, for any normal modal logic L, the CIP implies the DIP and the MIP for $\vdash_{\mathtt{L}}$. For any normal modal logic L, the DIP also implies the weak interpolation property WIP; see [51].

3.2.1 Normal Modal Logics Generally

Many fundamental normal modal logics arise as the logics of easily described classes of modal Kripke frames. For instance:

- `T` is the extension of `K` by the axiom $\Box\alpha \to \alpha$, and is the logic of the class of reflexive frames.
- `D` is the extension of `K` by the axiom $\Box\alpha \to \Diamond\alpha$, and is the logic of serial frames.
- `K4` is the extension of `K` by the axiom $\Box\alpha \to \Box\Box\alpha$, and is the logic of the class of transitive frames.
- `D4` is the extension of `D` by the axiom $\Box\alpha \to \Box\Box\alpha$, and is the logic of transitive and serial frames.
- `S4` is the extension of `T` by $\Box\alpha \to \Box\Box\alpha$, and is the logic of the class of frames whose accessibility relation is a preorder.
- `S5` is the extension of `S4` by the axiom $\Diamond\alpha \to \Box\Diamond\alpha$, and is the logic of frames whose accessibility relation is an equivalence relation.

Each of `K`, `T`, `D`, `K4`, `D4`, `S4`, and `S5` has the CIP, and hence the DIP and the MIP. In fact, each of these has the stronger Lyndon interpolation property for implication; see Chapter 5. Various proofs exist for these elementary results, and the reader may consult the monograph of Gabbay and Maksimova [33, Chapter 5] for detailed information.

One more fundamental normal modal logic demands special attention. Provability logic `G` is the extension of `K4` by the axiom $\Box(\Box\alpha \to \alpha) \to \Box\alpha$, and Smorynski showed in [80] that `G` has the CIP (hence also the DIP and the WIP). The logic `G` has proven quite important to understanding the scope of interpolation in normal modal logics, and Maksimova has established the following result.

▶ **Theorem 6** ([50, Theorem 10]). *There are continuum-many axiomatic extensions of* `G` *with the CIP, and hence the DIP and WIP as well.*

The next theorem, regarding weak interpolation, fills out this picture. In the following, `S4.1` refers to extension of `S4` by the McKinsey axiom $\Box\Diamond\alpha \to \Diamond\Box\alpha$.

▶ **Theorem 7** ([51, Proposition 1]). *Any normal modal logic containing* `S4.1` *or* `G` *has the WIP.*

3.2.2 Extensions of S4

While Theorems 6 and 7 may give the reader the impression that interpolation is rather common in normal modal logics, it turns out that this is not the case in extensions of `S4`: Maksimova has shown in [49] that there are just finitely many axiomatic extensions of `S4` with the DIP or the CIP.

▶ **Theorem 8** ([49, 33]). *Including the trivial logic consisting of all formulas, there are at most 37 axiomatic extensions of* S4 *with* CIP *and at most 49 axiomatic extensions of* S4 *with the* DIP*. There are at least 31 axiomatic extensions of* S4 *with the* CIP *and at least 43 axiomatic extensions of* S4 *with the* DIP*. In particular, there are at least 12 extensions of* S4 *that have the* DIP *but not the* CIP*.*

The proof relies on algebraic techniques linking interpolation to amalgamation (see Chapter 8) as well as the Gödel translation, linking interpolation for extensions of S4 to corresponding extensions of IPC. The question of interpolation remains open for six axiomatic extensions of S4, and the main difficulty seems to be that these logics are non-canonical. This remains one of the longest-standing problems in interpolation.

▶ **Open Question 9.** *How many axiomatic extensions of* S4 *have the* DIP*? What about the* CIP*?*

A full discussion of interpolation in extensions of S4, together with a precise list of the extensions of S4 known to have the CIP and the DIP, may be found in [33, Chapter 8].

We note that Grzegorczyk's logic Grz is among extensions of S4 with the CIP. The logic Grz is axiomatized relative to S4 by the $\Box(\Box(\alpha \to \Box\alpha) \to \alpha) \to \alpha$, and was first shown to have the CIP by Boolos in [10]. Interestingly, for extensions of Grz, the CIP and the DIP coincide; counting the trivial logic, there are seven of these (see [33, Theorem 8.46]).

3.2.3 Non-Normal Modal Logics

Non-normal modal logics are those that do not validate the characteristic axiom $\Box(\alpha \to \beta) \to (\Box\alpha \to \Box\beta)$ of the logic K or otherwise do not validate the rule of necessitation. Dropping these conditions presents a significant challenge, since, for example, the characteristic axiom of K is validated in all modal Kripke frames, and hence non-normal modal logics cannot be studied using the relational semantical methods typical of normal modal logics.

A paradigmatic example of a non-normal modal logic is given by the basic monotone modal logic, which replaces the aforementioned characteristic axiom of K by the rule 'from $\alpha \to \beta$, derive $\Box\alpha \to \Box\beta$'. Santocanale and Venema have shown in [76] that the basic monotone modal logic has a uniform variant of the CIP.

Various kinds of interpolation have also been established for several other non-normal modal logics, such as the congruential modal logic E that replaces the characteristic axiom of K by the rule 'from $\alpha \leftrightarrow \beta$, infer $\Box\alpha \leftrightarrow \Box\beta$', which has a uniform Lyndon variant of the CIP. See [81] for this as well as for a recent proof-theoretic study of interpolation in several basic systems of non-normal modal logic.

3.3 Paraconsistent Logics

The *principle of explosion* stipulates that, for some given logic L with a negation connective $\neg$, we have

$$\alpha, \neg\alpha \vdash \beta.$$

'Paraconsistent logic' is an umbrella term most commonly referring to any of several logics that refute the principle of explosion. Such logics are motivated by a number of concerns, most notably the need for reasoning non-trivially in the presence of contradictions and attendant applications. Here we consider just a few of the most prominent families of paraconsistent systems.

3.3.1 The Logic of Paradox

Probably the most famous paraconsistent logic is Priest's logic of paradox LP; see [73]. The logic LP is often defined over the language $\mathcal{L}_{\mathtt{LP}} = \{\wedge, \vee, \neg\}$ and is specified algebraically by certain matrix models, as in, e.g., [63]. Sometimes an implication connective $\rightarrow$ is also included in the language, but this connective does not satisfy modus ponens or modus tollens; see [73]. Considered over the language $\mathcal{L}_{\mathtt{LP}}$ and defined according to the usual matrix definitions, LP enjoys the following form of the deductive interpolation property:

> If $\nvdash_{\mathtt{LP}} \beta$ and $\alpha \vdash_{\mathtt{LP}} \beta$, then there is a formula δ such that $\alpha \vdash_{\mathtt{LP}} \delta$, $\delta \vdash_{\mathtt{LP}} \beta$, and $\mathsf{Var}(\delta) \subseteq \mathsf{Var}(\alpha) \cap \mathsf{Var}(\beta)$.

See [63] for a discussion placing this interpolation result in the context of so-called split interpolation, and see [9] for a more wide-ranging discussion of split interpolation in three-valued logics.

3.3.2 Johansson's Minimal Logic

Some other paraconsistent logics are rather closer in flavor to IPC. Johansson's minimal logic Jhn (see [42]) is one of these. It can be formulated in the basic language $\mathcal{L}_{\mathsf{bas}} = \{\wedge, \vee, \rightarrow, \bot, \top\}$ according to a Hilbert-style calculus that relaxes some of the conditions of IPC; see [70, Chapter 2]. Indeed, IPC itself can be axiomatized relative to Jhn by adding the single axiom $\bot \rightarrow \alpha$. In semantic terms, Jhn can be characterized as the logic corresponding to *j-algebras*, which are algebraic structures of the form $(A, \wedge, \vee, \rightarrow, \bot, \top)$ such that $(A, \wedge, \vee, \rightarrow, \top)$ is a Brouwerian algebra (aka an implicative lattice), i.e. a $\bot$-free subreduct of a Heyting algebra, and $\bot$ is an arbitrary constant.

Although it is a close cousin of IPC, understanding interpolation in Jhn has proven very challenging. Among other things, it remains open whether there are finitely many axiomatic extensions of Jhn with CIP, which is equivalent to DIP in this setting. There are, however, some results confined to tractable extensions of Jhn, and we summarize a couple notable landmarks.

▶ **Proposition 10** ([55, Corollary 5]). *Let* JX *be the extension of* Jhn *by the axiom* $(\bot \to \alpha) \vee (\alpha \to \bot)$. *There are finitely many axiomatic extensions of* JX *with the CIP or, equivalently, the DIP.*

▶ **Proposition 11** ([54, Theorem 5.8]). *Let* Gl *be the extension of* Jhn *by the law of excluded middle* $\alpha \vee \neg\alpha$. *There are exactly twenty axiomatic extensions of* Gl *with the CIP or, equivalently, the DIP.*

The logic Gl has played a significant role in the study of interpolation in extensions of Jhn. In [52], Maksimova shows that the study of the WIP for extensions of Jhn can be reduced to the case of Gl. A complete description of axiomatic extensions of Jhn with this weak interpolation property is given in [53], which shows that the set of such logics is the union of eight pairwise disjoint intervals in the lattice of axiomatic extensions.

Many questions about interpolation in extensions of Jhn remain open.

▶ **Open Question 12.** *How many axiomatic extensions of* Jhn *have the CIP (equivalently, the DIP)? Are there infinitely many such extensions?*

▶ **Open Question 13.** *Is it possible to give a complete classification of extensions of* Jhn *with the CIP (equivalently, the DIP)?*

Despite the apparent difficulty of the previous questions, we note that in some sense interpolation in extensions of Jhn is constrained to certain regions of the lattice of extensions. Odintsov has shown that the lattice of extensions of Jhn may be partitioned into certain intervals according to a classification based on the intuitionistic and negative companions of these logics; see [68]. Only finitely many of these intervals contain logics with the CIP. For the most up-to-date summary of what is currently known, the reader may consult [56].

3.3.3 Nelson's Constructive Logic With Strong Negation

Nelson's logic with strong negation [67] amounts to the expansion of intuitionistic propositional logic IPC by an additional negation $\sim$ that satisfies De Morgan's Laws. There are several different variants of this logic, most often formulated in various fragments of the language $\mathcal{L}_{\mathsf{N}} = \{\wedge, \vee, \to, \sim, \bot, \top\}$. We will focus mostly on Nelson's paraconsistent logic $\mathtt{N4}^{\bot}$, which can be specified as a logic in the language $\mathcal{L}_{\mathsf{N}}$ by a Hilbert-style calculus or using algebraic models; see [70, Chapter 8]. The fragment of $\mathtt{N4}^{\bot}$ without $\sim$ coincides with IPC. On the other hand, the fragment without $\bot, \top$, denoted N4, is also sometimes considered since it dispenses with the intuitionistic negation $\alpha \mapsto \alpha \to \bot$. Nelson's logic also has a much-studied variant that satisfies the principle of explosion, and this logic, denoted N3, can be obtained by adding to N4 the single axiom $\sim\alpha \to (\alpha \to \beta)$; see [68, Section 8.1]. The version of N3 with both $\sim$ and the intuitionistic negation is denoted $\mathtt{N3}^{\bot}$.

Using algebraic tools, Goranko has obtained a complete classification of axiomatic extensions of $\mathtt{N3}^{\bot}$ with the Craig interpolation property.

▶ **Theorem 14** ([40, Theorem 71]). *There are exactly fifteen axiomatic extensions of* $\mathtt{N3}^{\perp}$ *with the CIP. In particular, seven of the non-trivial axiomatic extensions of* $\mathtt{N3}^{\perp}$ *with the CIP arise from expanding the seven non-trivial extensions of* IPC *with the CIP (see Theorem 4) with a strong negation* $\sim$ *satisfying* $\neg\alpha \to (\alpha \to \beta)$*. The remaining seven non-trivial axiomatic extensions of* $\mathtt{N3}^{\perp}$ *with the CIP are the aforementioned logics respectively extended with the axiom* $\neg(\alpha \leftrightarrow \sim \alpha)$*.*

Metcalfe has also given an analytic sequent calculus for $\mathtt{N3}^{\perp}$ in [58], which can be used to give a different proof that $\mathtt{N3}^{\perp}$ has the CIP.

Odintsov has widened Goranko's classification to give a complete classification of axiomatic extensions of $\mathtt{N4}^{\perp}$ with the CIP.

▶ **Theorem 15** ([69, Theorem 5.9]). *There are exactly 29 axiomatic extensions of* $\mathtt{N4}^{\perp}$ *with the CIP. Fifteen of these are given by Theorem 14. Seven of the remaining logics are given by expanding the seven non-trivial extensions of* IPC *with the CIP with a strong negation* $\sim$*. The remaining seven logics are the respective axiomatic extensions of the latter logics by the axiom* $\neg\neg(\alpha \vee \sim \alpha)$*.*

Note that the axiomatic extensions of $\mathtt{N4}^{\perp}$ have a classical deduction theorem (see [66]), so the preceding theorems also given complete classifications for axiomatic extensions with the DIP.

3.4 Fuzzy Logics

The moniker *fuzzy logic* typically applies to any of several systems intended to capture reasoning where truth values are vague, fuzzy, or uncertain.[2] Historically, the development of fuzzy logic has been heavily influenced by various efforts to endow the real unit interval $[0, 1]$ with logical structure (often algebraic in nature) making it into a set of truth values, where the extrema 0 and 1 represent 'crisp' or classical truth values. The restriction of the logical structure overlaying $[0, 1]$ to the crisp values in $\{0, 1\}$ should then recover CPL according to this program.

Fuzzy logic has undergone several successive phases of generalization over the past century or so, leading to frameworks that progressively encompass the aforementioned ideas in broader and broader environments. We will organize our discussion of interpolation in fuzzy logics more or less in order of increasing generality, touching on the most prominent frameworks.

3.4.1 Łukasiewicz's Logic and Its Extensions

Probably the most thoroughly studied fuzzy logics—and also among the oldest—are the various systems of Łukasiewicz, of which the infinite-valued Łukasiewicz logic Ł is the most

[2] These are sometimes called 'many-valued' logics. However, since most of the non-classical logics considered here arguably have many truth values in various semantic frameworks, we do not use that term here.

prominent; see [46], and for a historical account see [13]. The logic Ł may be formulated in a number of equivalent languages. For the sake of relating it to other logics that we discuss, we note that it can be formulated in the language $\mathcal{L}_{\mathsf{sc}} = \{\wedge, \vee, \cdot, \to, 0, 1\}$, where $\cdot$ gives a kind of 'strong conjunction' and the negation $\neg\alpha$, which abbreviates $\alpha \to 0$, satisfies De Morgan's laws as well as the law of double negation $\alpha \leftrightarrow \neg\neg\alpha$. In many sources, instead of $\cdot$ the De Morgan dual connective $\oplus$ is used, i.e., $\alpha \oplus \beta$ abbreviates $\neg(\neg\alpha \cdot \neg\beta)$. The connective $\oplus$ thus represents a kind of 'strong disjunction'.

The logic Ł has a rather elaborate proof theory (see [60]), so Ł and its extensions are usually studied using semantic methods. These most usually involve the equational class of *MV-algebras* (see [14]), which give the equivalent algebraic semantics of Ł. In particular, Ł is sound and complete with respect to a single algebraic model defined on $[0, 1]$, where $\wedge$ and $\vee$ interpret the usual binary maximum and minimum, $\cdot$ is defined by $x \cdot y = \max\{x + y - 1, 0\}$, where $+$ is the usual sum of real numbers, and $\to$ is uniquely specified by stipulating that $x \to y$ is the greatest element z of $[0, 1]$ such that $x \cdot z \leq y$. This single algebraic model generates the entire equational class of MV-algebras.

Every axiomatic extension $\mathtt{L}$ of Ł has a (implicative) local deduction theorem of the form

$$\Gamma, \alpha \vdash_{\mathtt{L}} \beta \iff \Gamma \vdash_{\mathtt{L}} \alpha^n \to \beta \text{ for some positive integer } n,$$

where α^n is an abbreviation for the formula $\alpha \cdot \alpha \cdot \ldots \cdot \alpha$ in which α occurs n times. Thus, for any axiomatic extension $\mathtt{L}$ of Ł, the CIP implies the DIP. However, as shown by Komori, there are very few axiomatic extensions of Ł with the CIP.

▶ **Theorem 16** ([44]). *The only axiomatic extensions of* Ł *with the CIP are* CPL *and the trivial logic consisting of all formulas.*

On the other hand, there are many more axiomatic extensions of Ł with the DIP. The following result is due to Di Nola and Lettieri, who proved it in an equivalent algebraic formulation by giving a complete description of equational classes of MV-algebras with the amalgamation property; see Chapter 8.

▶ **Theorem 17** (cf. [20]). *The axiomatic extensions of* Ł *that have the DIP are exactly those characteristic with respect to a single totally ordered MV-algebra. Thus, in particular, there countably infinitely many of these.*

Metcalfe, Montagna, and Tsinakis have obtained an analogous, albeit somewhat more complicated, classification for axiomatic extensions of the positive fragment of Ł that have the DIP; see [59, Theorem 63]. Like Di Nola and Lettieri, their result is phrased in algebraic terms: They actually give a complete classification of equational classes of Wajsberg hoops that have the amalgamation property, and the result for the DIP follows because these algebraic structures provide the equivalent algebraic semantics of positive Łukasiewicz logic. We do not state the classification in full, but note the following consequence.

▶ **Proposition 18** (cf. [59, Theorem 63]). *Let* Ł^+ *be the positive fragment of* Ł*. There are countably infinitely many axiomatic extensions of* Ł^+ *with the DIP.*

3.4.2 Hájek's Basic Fuzzy Logic and Its Extensions

As discussed above, Łukasiewicz logic is characterized by a single algebraic model constructed on top of the real unit interval $[0,1]$, where the strong conjunction is interpreted by the binary operation $x \cdot y = \max\{x + y - 1, 0\}$. This operation is associative, commutative, has 1 as a unit element, and is monotone in each of its coordinates. The binary operations on $[0,1]$ satisfying this latter list of conditions are often called *triangular norms* or *t-norms* for short. Triangular norms have played an extremely important role in the development of approximate reasoning, and the Łukasiewicz t-norm defined by $x \cdot y = \max\{x + y - 1, 0\}$ has an additional important property: It is continuous as a map $[0,1] \times [0,1] \to [0,1]$, where $[0,1]$ has its usual topology.

Continuous t-norms have a number of nice properties. Among other things, each continuous t-norm has a corresponding residual operation $\to\colon [0,1] \times [0,1] \to [0,1]$ defined by stipulating that $x \to y = \max\{z \in [0,1] \mid x \cdot z \leq y\}$, which is equivalent to stipulating the *residuation law*,

$$x \cdot y \leq z \iff x \leq y \to z.$$

Further, the operations of binary infimum $\wedge$ and binary supremum $\vee$ turn out to be definable in the this language too. Hájek undertook an extensive study of the logics arising in this way from continuous t-norms in [41]. There he provided a Hilbert-style axiomatization for his basic fuzzy logic BL as a logic in the language $\mathcal{L}_{\mathsf{sc}} = \{\wedge, \vee, \cdot, \to, 0, 1\}$. Later on, Cignoli, Godo, Esteva, and Torrens showed in [15] that Hájek's axiomatization is complete: BL is exactly the logic of continuous t-norms.

Interpolation for BL and neighboring logics has been the subject of lots of work for several decades now. Although BL is the logic of all continuous t-norms, arbitrary axiomatic extensions of BL may not be characterized by continuous t-norms on [0,1]. Among these, Baaz and Veith have shown in [7] that the only logic with the CIP is Gödel-Dummett logic (see Theorem 4), which may be viewed as the extension of BL by the axiom $x \leftrightarrow x \cdot x$. In particular, BL itself does not have the CIP. This result has subsequently been strengthened by Montagna [64], who showed that the only axiomatic extensions of BL with the CIP are (definitionally equivalent to) superintuitionistic logics.

▶ **Theorem 19** ([64], cf. [7]). *The axiomatic extensions of Hájek's basic fuzzy logic* BL *with the CIP are exactly:*

1. *Gödel-Dummett logic* LC*, axiomatized relative to* BL *by* $x \leftrightarrow x \cdot x$.
2. *The axiomatic extension of* LC *characterized by the three-element Heyting algebra.*
3. *Classical propositional logic* CPL.
4. *The trivial logic consisting of all formulas.*

Axiomatic extensions of BL have the same local deduction theorem as Ł, i.e., every axiomatic extension L of BL satisfies

$$\Gamma, \alpha \vdash_{\mathtt{L}} \beta \iff \Gamma \vdash_{\mathtt{L}} \alpha^n \to \beta \text{ for some positive integer } n,$$

so the CIP implies the DIP in this setting, and the DIP is equivalent to the MIP. However, the DIP need not imply the CIP for an axiomatic extension of BL. Indeed, the situation is much more complicated for deductive interpolation. The first progress on this was made by Montagna in 2006.

▶ **Theorem 20** ([64]). *Hájek's basic fuzzy logic* BL *has the DIP, but there are uncountably many axiomatic extensions of* BL *that do not have the DIP.*

Montagna did not obtain a complete description of axiomatic extensions of BL with the DIP, or even determine whether there are countably many of these (there are at infinitely many of them since Ł is an axiomatic extension of BL). Subsequent work by Cortonesi, Marchioni, and Montagna [17] (applying techniques from first-order model theory) and Aguzzoli and Bianchi [1, 2] (applying more algebraic techniques) significantly deepened our understanding of the DIP in axiomatic extensions of BL, and a complete description of these was finally obtained by Fussner and Santschi in [30]. This description turns out to be rather elaborate, but we summarize a few of its salient features in the following theorem.

▶ **Theorem 21** ([30]). *There are only countably many axiomatic extensions of* BL *with the DIP. Indeed, the subposet of these in the lattice of all axiomatic extensions of* BL *is the disjoint union of countably infinitely many finite intervals. The same is true of the positive fragment of* BL.

Indeed, the axiomatic extensions of BL with the DIP are explicitly described in [30] in terms of their characteristic generating models. Although this description is quite explicit, an explicit axiomatization of these countably infinitely many logics is not known.

▶ **Open Question 22.** *Is it possible to give explicit axiomatizations for the extensions of* BL *with the DIP? Are all of them finitely axiomatizable?*

The description of extensions of BL with the DIP from [30] draws on techniques from regular languages, and suggests that there is some decidability result lurking in the background. In connection with the previous question, one might wonder about the following point.

▶ **Open Question 23.** *Is the problem of determining whether a given finitely axiomatized extension of* BL *has the DIP effectively decidable?*

The previous question amounts, of course, to whether it is possible to determine algorithmically whether a given finitely axiomatized extension of BL is one of the logics listed in [30].

3.4.3 Monoidal T-norm Based Logics

One of the most notable features of continuous t-norms, considered as opposed to arbitrary t-norms, is that each continuous t-norm $\cdot$ has a residual defined by $x \to y = \max\{z \in [0,1] \mid x \cdot z \leq y\}$. This fact is not a consequence of full continuity, but instead only involves the lower semicontinuity of the t-norm $\cdot$. Consequently, logics associated to lower semicontinuous t-norms have recently attracted considerable attention. The first proper study of this topic was [22], which introduced the monoidal t-norm based logic MTL as a logic defined over the basic language $\{\wedge, \vee, \cdot, \to, 0, 1\}$. This logic is characteristic with respect to arbitrary algebras of lower semicontinuous t-norms, and BL itself is the axiomatic extension of MTL by the divisibility condition $x \cdot (x \to y) \leftrightarrow x \wedge y$.

To date, there is not much systematic information about interpolation in axiomatic extensions of MTL that go beyond the case of BL extensions. Analogous proofs as those given by Baaz and Veith [7] and Montagna [64] show that the only axiomatic extensions of MTL with the CIP are the axiomatic extensions of BL with the CIP, i.e., those summarized in Theorem 19. By showing the failure of amalgamation in certain associated varieties, it can be shown with small algebraic models that MTL does not have the DIP either; see [29, 39], but we have not yet identified any minimal axiomatic extensions of MTL with the DIP. Many other open questions remain in this vicinity, e.g., the cardinality of the set of axiomatic extensions of MTL with the DIP is not known.

▶ **Open Question 24.** *Is there a minimum axiomatic extension of* MTL *with the DIP? Equivalently, is there a largest variety of MTL-algebras with the amalgamation property?*

▶ **Open Question 25.** *Are there uncountably many axiomatic extensions of* MTL *with the the DIP?*

3.5 Relevant Logics

Relevant logics arise from the intuition that, in any logically valid implication $\alpha \to \beta$, the antecedent α and the consequent β ought to be germane to one another. This is often made precise by requiring the *variable sharing property*, which demands that for a logic L,

$$\vdash_{\mathtt{L}} \alpha \to \beta \Rightarrow \mathsf{Var}(\alpha) \cap \mathsf{Var}(\beta) \neq \emptyset.$$

The main line of research on relevant logics descends from the school of Anderson and Belnap, represented, for example, by the monographs [3, 4]. Probably the most prominent logical system in this family is the logic R, which is often specified either in the language $\{\wedge, \vee, \to, \neg\}$ or $\{\wedge, \vee, \cdot, \to, \neg\}$. The operation $\cdot$ represents a form of conjunction often called *cotenability*, and may be defined from the others by stipulating that $\alpha \cdot \beta$ abbreviates $\neg(\neg\alpha \to \beta)$. The most extensively studied axiomatic extension of R is probably the logic R-mingle RM, axiomatized relative to R itself by adding

$$\alpha \to (\alpha \to \alpha),$$

an axiomatic form of the mingle rule (see Section 3.6).

Various technical concerns have also motivated the addition of logical constants—often called *Ackermann constants* in the relevantist literature—$\mathtt{t}$ and $\mathtt{f}$ to the language. When this is done, $\mathtt{t}$ is usually stipulated to be a neutral element for cotenability and $\mathtt{f}$ is its negation. The logics in this expanded signature are often denoted by $\mathtt{R}^{\mathtt{t}}$ and $\mathtt{RM}^{\mathtt{t}}$, respectively. All of these logics can be variously specified by Hilbert-style systems or algebraic semantics in terms of so-called *De Morgan monoids* [3], or by relational semantics that employ a ternary accessibility relation [75]. In the case of R-mingle and its variants, the accessibility relation can be taken to be binary; see [21, 24].

The presence of the Ackermann constants makes a big difference for these logics: $\mathtt{R}$ does not have a local deduction theorem, but $\mathtt{R}^{\mathtt{t}}$ has an explicit deduction theorem of the form

$$\Gamma, \alpha \vdash_{\mathtt{R}^{\mathtt{t}}} \beta \iff \Gamma \vdash (\alpha \wedge \mathtt{t}) \to \beta.$$

The same deduction theorem holds for axiomatic extensions of $\mathtt{R}^{\mathtt{t}}$, including $\mathtt{RM}^{\mathtt{t}}$. On the other hand, $\mathtt{RM}$ has an explicit deduction theorem with the non-standard form

$$\Gamma, \alpha \vdash_{\mathtt{RM}} \beta \iff (\neg(\alpha \to \neg\beta) \vee (\alpha \to \beta)) \wedge (\neg\alpha \vee \beta),$$

a result due to Tokarz; see [18, p. 126].

Urquhart showed in [82] that the CIP fails for a huge number of relevant logics, notably including $\mathtt{R}$ and $\mathtt{R}^{\mathtt{t}}$, as well as the related systems $\mathtt{E}$ of entailment and $\mathtt{T}$ of ticket entailment. Urquhart's proof used techniques from projective geometry to construct some ternary-relational models witnessing the failure of amalgamation (see Chapter 8) from which an implication serving as a counterexample for the CIP could be extracted. As an artifact of these geometric techniques, the counterexample witnessing failure of the CIP was very large: The largest of the three algebras giving the amalgamation failure had cardinality exceeding 10^{247}, and the implication showing the failure of CIP had 14 variables. Later on, by exhibiting the failure of Beth definability, Urquhart gave a much simpler proof of the failure of CIP in $\mathtt{R}^{\mathtt{t}}$ and some other logics in the vicinity, showing that the valid three-variable implication

$$\vdash_{\mathtt{R}^{\mathtt{t}}} [\mathtt{t} \wedge (q \to \mathtt{f}) \wedge ((q \wedge p) \to \mathtt{t})] \to \{[\mathtt{t} \wedge (\mathtt{t} \to r) \wedge (\mathtt{f} \to (p \vee r))] \to (q \to r)\},$$

has no Craig interpolant; see [83, 11].

Prior to Urquhart's work, Anderson and Belnap showed at [3, pp. 416-417] that $\mathtt{RM}$ lacks CIP in the following formulation:

> Whenever $\nvdash_{\mathtt{RM}} \neg\alpha$, $\nvdash_{\mathtt{RM}} \beta$, and $\vdash_{\mathtt{RM}} \alpha \to \beta$, then there exists a formula δ such that $\vdash_{\mathtt{RM}} \alpha \to \delta$, $\vdash_{\mathtt{RM}} \delta \to \beta$, and $\mathsf{Var}(\delta) \subseteq \mathsf{Var}(\alpha) \cap \mathsf{Var}(\beta)$.

Avron has introduced another implication-like connective $\supset$ in $\mathtt{RM}$, defined in terms of the primitive connectives by

$$\alpha \supset \beta = (\alpha \to \beta) \vee \beta$$

Avron has shown in [5] that this connective gives a classical deduction theorem for $\mathtt{RM}$ and that it satisfies the following interpolation property:

If $\vdash_{\mathtt{RM}} \alpha \supset \beta$ and $\not\vdash_{\mathtt{RM}} \beta$, then there is a formula δ such that $\vdash_{\mathtt{RM}} \alpha \to \delta$, $\vdash_{\mathtt{RM}} \delta \to \beta$, and $\mathsf{Var}(\delta) \subseteq \mathsf{Var}(\alpha) \cap \mathsf{Var}(\beta)$.

Returning to the case with logical constants $\mathtt{t}$ and $\mathtt{f}$, Meyer reportedly showed in [62] that $\mathtt{RM}^{\mathtt{t}}$ has the CIP. Much later, Marchioni and Metcalfe gave an exhaustive classification of the axiomatic extensions of $\mathtt{RM}^{\mathtt{t}}$ with the CIP, using a combination of techniques from algebra and model theory, in particular tools from quantifier elimination.

▶ **Theorem 26** ([57]). *Including the trivial logic consisting of all formulas, there are exactly nine axiomatic extensions of* $\mathtt{RM}^{\mathtt{t}}$ *with the CIP. Further, the CIP and the DIP coincide for axiomatic extensions of* $\mathtt{RM}^{\mathtt{t}}$*, so these are also exactly the axiomatic extension with the DIP.*

The fact that the CIP and the DIP coincide in this setting is especially notable since, unlike the many of the previously considered cases that exhibit this behavior, $\mathtt{RM}^{\mathtt{t}}$ does not have a classical deduction theorem for the usual implication.

The logic $\mathtt{RM}^{\mathtt{t}}$ is characterized by linearly ordered De Morgan monoids called *Sugihara monoids*, but Sugihara monoids do not exhaust the linearly ordered algebraic models of $\mathtt{R}^{\mathtt{t}}$. The class of all semilinear De Morgan monoids—that is, those De Morgan monoids that are subalgebras of direct products of linearly ordered ones—characterizes an axiomatic extension $\mathtt{SemR}^{\mathtt{t}}$ of which the class of all semilinear De Morgan monoids is the equivalent algebraic semantics. The following result contrasts against the apparent scarcity of relevant logics with interpolation.

▶ **Proposition 27** ([29, Proposition 5.3]). *The logic* $\mathtt{SemR}^{\mathtt{t}}$ *does not have the DIP and hence does not have the CIP either, but there are infinitely many axiomatic extensions of* $\mathtt{SemR}^{\mathtt{t}}$ *that have the DIP.*

We do not yet have a complete classification of axiomatic extensions of $\mathtt{R}^{\mathtt{t}}$ with the DIP or the CIP, nor even such a classification for $\mathtt{SemR}^{\mathtt{t}}$. Obtaining such a complete classification likely demands a more refined structural understanding of De Morgan monoids.

▶ **Open Question 28.** *Is it possible to classify the extensions of* $\mathtt{SemR}^{\mathtt{t}}$ *with the DIP? What about the CIP? What about for* $\mathtt{R}^{\mathtt{t}}$*,* $\mathtt{R}$ *and* $\mathtt{RM}$*?*

Urquhart's many results are often wrongly summarized as saying that relevant logics lack interpolation. As we have seen above, the situation is more complicated. To the extent that the variable sharing property can be taken as a hallmark for relevance, the following question is raised.

▶ **Open Question 29.** *What is the precise relationship between interpolation and the variable sharing property?*

$$\frac{\Gamma_1, \Pi_1, \Pi_2, \Gamma_2 \Rightarrow \Delta}{\Gamma_1, \Pi_2, \Pi_1, \Gamma_2 \Rightarrow \Delta}\ (\mathrm{e}) \qquad \frac{\Gamma_1, \Pi, \Pi, \Gamma_2 \Rightarrow \Delta}{\Gamma_1, \Pi, \Gamma_2 \Rightarrow \Delta}\ (\mathrm{c})$$

$$\frac{\Gamma_1, \Gamma_2 \Rightarrow \Delta}{\Gamma_1, \Pi, \Gamma_2 \Rightarrow \Delta}\ (\mathrm{i}) \qquad \frac{\Pi \Rightarrow}{\Gamma_1, \Pi, \Gamma_2 \Rightarrow \Delta}\ (\mathrm{o})$$

$$\frac{\Gamma_1, \Pi_1, \Gamma_2 \Rightarrow \Delta \quad \Gamma_1, \Pi_2, \Gamma_2 \Rightarrow \Delta}{\Gamma_1, \Pi_1, \Pi_2, \Gamma_2 \Rightarrow \Delta}\ (\mathrm{m})$$

Figure 1 The basic structural rules of proof theory, as formulated in [61]. Here each Γ_i, Π, Δ, and so on denote finite sequences of formulas, with at most one formulas appearing to the right of $\Rightarrow$.

3.6 Substructural Logics

Many of the logics discussed in the previous sections can be regarded as *substructural logics*. There is no mathematically precise demarcation of this family of logics, but they are unified by the idea of dropping or restricting the application of the structural rules of exchange (e), left weakening (i), right weakening (o), contraction (c), and mingle (m) that are validated in the Gentzen calculus for IPC; see Figure 1 and also Chapter 5. Most often, this involves including a 'strong conjunction' connective $\cdot$ in the basic language, which simulates comma as it appears in Gentzen calculi. Further modification of the basic signature of IPC is necessitated in some cases. For example, comma is non-commutative when the exchange rule (e) is dropped, and this presents the possibility of having separate modus ponens rules from the left and the right. Thus, we may have two separate implication connectives $\backslash$ and $/$ linked to these two modus ponens rules: $\alpha, \alpha\backslash\beta \Rightarrow \beta$ and $\beta/\alpha, \alpha \Rightarrow \beta$. One may also include a truth constant e and a falsity constant f. For our purposes here, we take the basic language of substructural logics to be $\mathcal{L}_{\mathtt{FL}} = \{\wedge, \vee, \cdot, \backslash, /, \mathsf{e}, \mathsf{f}\}$. We will also consider the positive (i.e., f-free) language $\mathcal{L}^+_{\mathtt{FL}}$.

Many well-studied substructural logics can be understood as axiomatic extensions of the full Lambek calculus FL, a logic arising from an analytic Gentzen-style sequent calculus formulated in the language $\mathcal{L}_{\mathtt{FL}}$ that omits all of the basic structural rules appearing in Figure 1. Many of the logics we have considered—for instance, CPL, IPC, Ł, BL, MTL, $\mathtt{R}^{\mathsf{t}}$, $\mathtt{RM}^{\mathsf{t}}$, and all axiomatic extensions of these—can be realized in an equivalent formulation as axiomatic extensions of FL. In the latter axiomatic extensions—and indeed in any axiomatic extensions of FL validating the exchange rule—one may prove that $\alpha\backslash\beta$ and β/α are interderivable. Thus, in the presence of exchange, one may get away with only a single implication in the language, taking us back to the more familiar formulation of several logics with exchange that we have already seen.

The logic FL and its axiomatic extensions also admit an equivalent algebraic semantics in terms of generalizations of Heyting algebras called *FL-algebras*. Similar remarks apply to the positive fragment of FL, which is algebraized by the equational class of *residuated lattices*. See the monographs [61, 34] for details, along with Chapter 8.

Since FL has two distinct implications, it isn't immediately clear what we mean by the CIP in this context. Thankfully, one may show that having CIP with respect to $\backslash$ is equivalent to having the CIP with respect to $/$, for any axiomatic extension of FL. Hence we may speak of the CIP for some given axiomatic extension of FL tout court.

Crucially, FL does not even have a local deduction theorem. It admits a parameterized local deduction theorem (see e.g. [34, Section 2.4]), but this kind of deduction theorem is insufficient for the usual proof that the CIP implies the DIP that holds in many logics. It is shown at [43, Proposition 10] that the CIP and the MIP are independent for axiomatic extensions of FL, but it is not currently known whether the CIP implies the DIP for arbitrary axiomatic extensions of FL.

▶ **Open Question 30.** *Let* L *be an axiomatic extension of* FL*. If* L *has the CIP, does* L *have the DIP?*

The lack of a useful deduction theorem for FL presents many challenges to the study of interpolation for FL and its extensions. In the presence of the exchange rule (e), the deduction theorem for FL simplifies considerably and axiomatic extensions with exchange are much better behaved from the perspective of interpolation. This is discussed further in Subsection 3.6.2 below.

3.6.1 Interpolation and the Basic Structural Rules

Aside from FL itself, the most fundamental axiomatic extensions of FL arise from adding basic structural rules, as in Figure 1, to the Gentzen calculus defining FL.[3] For the basic structural rules appearing in Figure 1, these are denoted by adding subscripts naming the added structural rules so that, for example, $\mathtt{FL_e}$ denotes FL with the exchange rule, $\mathtt{FL_{ec}}$ denotes FL with both the exchange and contraction rules, and so forth. The two weakening rules (i) and (o) are often considered together, so that $\mathtt{FL_w}$ denotes the addition of both of them. The logic $\mathtt{FL_{ecw}}$ is a definitional variant of IPC.

Because the aforementioned logics have well-behaved analytic sequent calculi, the CIP can be established for many of them using proof-theoretic techniques grounded in Maehara's method; see Chapter 5. The following results from the application of this proof strategy.

[3] Note that these extensions really are axiomatic: The exchange rule is captured by the axiom $(\alpha\cdot\beta)\to(\beta\cdot\alpha)$, the contraction rule by $\alpha\to(\alpha\cdot\alpha)$, the left weakening rule by $\alpha\to\mathsf{e}$, the right weakening rule by $\mathsf{f}\to\alpha$, and the mingle rule by $(\alpha\cdot\alpha)\to\alpha$.

▶ **Theorem 31** (See [71], [34, Section 5.1.3]). *Each of* $\mathtt{FL}$, $\mathtt{FL_e}$, $\mathtt{FL_w}$, $\mathtt{FL_{ew}}$, *and* $\mathtt{FL_{ec}}$ *has the CIP.*

Due to the lack of a local deduction theorem, we do not know whether $\mathtt{FL}$ and $\mathtt{FL_w}$ have the DIP. Santschi and Jipsen have recently shown in [77] that the equational class of FL-algebras does not have the amalgamation property and hence that $\mathtt{FL}$ does not have the Robinson property (see Chapter 8), but in the context of $\mathtt{FL}$ this appears to be insufficient to show that the DIP fails.

▶ **Open Question 32.** *Does* $\mathtt{FL}$ *have the the DIP?*

For logics with exchange, however, the situation is simpler.

3.6.2 Logics with Exchange

Although general axiomatic extensions of $\mathtt{FL}$ have only a parameterized local deduction theorem, every axiomatic extension $\mathtt{L}$ of $\mathtt{FL_e}$ has an implicative local deduction theorem of the following form:

$$\Gamma, \alpha \vdash_{\mathtt{L}} \beta \iff \Gamma \vdash_{\mathtt{L}} (\alpha \wedge \mathsf{e})^n \to \beta \text{ for some positive integer } n.$$

Consequently, for each axiomatic extension $\mathtt{L}$ of $\mathtt{FL_e}$, the CIP implies the DIP and so, by Theorem 31, each of $\mathtt{FL_e}$, $\mathtt{FL_{ew}}$, and $\mathtt{FL_{ec}}$ has the DIP. Moreover, because extensions of $\mathtt{FL_e}$ are conjunctive, the DIP is equivalent to the MIP in this setting. The local deduction theorem also eases the application of algebraic techniques in the study of the DIP for extensions of $\mathtt{FL_e}$—in particular allowing proof strategies based on amalgamation as in Chapter 8—and there has commensurately been quite a lot of success in studying the DIP in various axiomatic extensions of $\mathtt{FL_e}$. We have already seen this in the case of Ł, $\mathtt{BL}$, $\mathtt{R}^{\mathsf{t}}$, and $\mathtt{RM}^{\mathsf{t}}$, all of which can be realized as extensions of $\mathtt{FL_e}$.

Girard's linear logic [38], in both its classical and intuitionistic variants, can also be understood as arising from the calculus for $\mathtt{FL_e}$ by including additional constants $\bot$ and $\top$ as well as 'exponentials' ! and ?, which represent certain kinds of $\mathtt{S4}$-like modal box connectives. Roorda has shown that the CIP for these systems, as well as several fragments thereof, can also be established using the proof-theoretic methods previously mentioned; see [74]. Intuitionistic and classical linear logic also have implicative local deduction theorems due to Avron [6], so Roorda's result entails the DIP for both classical and intuitionistic linear logic as well. Indeed, using algebraic tools, Fussner and Santschi exhibited uncountably many axiomatic extensions of each of $\mathtt{FL}$, classical linear logic, and intuitionistic linear logic with the DIP.

▶ **Theorem 33** ([28]). *There are continuum-many axiomatic extensions of* $\mathtt{FL_e}$ *with the DIP. The same holds of the expansion of* $\mathtt{FL_e}$ *by constants designating bounds* $\bot$ *and* $\top$*, as well as by the exponentials* ! *and* ?*. In particular, each of intuitionistic linear logic and classical linear logic has continuum-many axiomatic extensions with the DIP.*

Notably, the axiomatic extensions constructed in [28] do not have the CIP. The number of axiomatic extensions of $\mathtt{FL_e}$ with the CIP is apparently unknown. However, since there are continuum-many axiomatic extensions of $\mathtt{FL_e}$ without the DIP, there are also continuum-many axiomatic extensions of $\mathtt{FL_e}$ without the CIP.

▶ **Open Question 34.** *How many axiomatic extensions of* $\mathtt{FL_e}$ *have the CIP? Are there infinitely many? Uncountably many?*

To close our discussion of logics with exchange, we note that Abelian logic—typically understood as the logic of Abelian lattice-ordered groups but also admitting a nice proof theory, as in [60]—can be also be realized as an axiomatic extension of $\mathtt{FL_e}$. Because Abelian lattice ordered groups have the amalgamation property, it can also be shown that Abelian logic has the DIP. See [28] for a discussion of amalgamation in Abelian lattice-ordered groups in the context of residuated lattices.

3.6.3 Logics Without Exchange

As the previous sections attest, the DIP is reasonably common among extensions of $\mathtt{FL_e}$. Until recently, it was thought that this was not the case among logics without exchange. Indeed, quite recently it was thought that the exchange rule (e) may be derivable in every axiomatic extension of FL with the DIP, cf. [37, Problem 5]. This was shown to not be the case by Gil-Férez, Jipsen, and Metcalfe in [36], which exhibited an equational class of idempotent residuated lattices with the amalgamation property that has non-commutative members. This equational class gives the equivalent algebraic semantics of a certain extension of $\mathtt{FL_{cm}}$ in which (e) is not derivable and which has the DIP. The aforementioned equational class is *semilinear* in the sense that its members consist of FL-algebras that are subalgebras of direct products of totally ordered FL-algebras, i.e., the equational class is generated by totally ordered algebras.

As in the case of De Morgan monoids (see Section 3.5), the equational class of FL-algebras generated by the linearly ordered FL-algebras characterizes an axiomatic extension SemFL of FL. It was shown in [25] that axiomatic extensions of $\mathtt{SemFL_{cm}}$ have a local deduction theorem of a non-standard form. Using this fact, [26] gave several more natural examples of axiomatic extensions of FL that have the DIP and in which (e) is not derivable. In fact, these results, as well as those of [36], actually apply not only to $\mathtt{SemFL_{cm}}$, but also its f-free fragment $\mathtt{SemFL^{+}_{cm}}$.

Recently, by building on [25, 26], Fussner, Metcalfe, and Santschi proved the following result.

▶ **Theorem 35** ([27, Theorem A])**.** *There are continuum-many axiomatic extensions of* $\mathtt{SemFL^{+}_{cm}}$ *with the DIP in which exchange is not derivable.*

Note that exchange is derivable in $\mathtt{FL_{cw}}$, which constitutes a definitionally equivalent formulation of $\mathtt{FL_{ecw}}$ (i.e., of IPC). On the other hand, the mingle rule (m) may be viewed as a slightly less powerful variant of weakening, and exchange is not derivable in $\mathtt{FL_{cm}}$ or even in

$\mathtt{SemFL}_{\mathtt{cm}}$. The previous theorem is thus rather striking in comparison to the case of IPC, since $\mathtt{FL}_{\mathtt{cm}}$ is one natural candidate for a logic resulting from IPC by 'dropping' exchange. On the other hand, adding exchange to $\mathtt{SemFL}_{\mathtt{cm}}$ results in the number of axiomatic extensions with the DIP dropping from uncountably many to only finitely many.

▶ **Theorem 36** ([27, Theorem B]). *There are exactly 60 axiomatic extensions of* $\mathtt{SemFL}^{+}_{\mathtt{ecm}}$ *with the DIP.*

Thus, in at least some contexts, the exchange rule is actually an impediment to deductive interpolation. This stands in contrast to previous expectations that there may be no logics without exchange that have the DIP. Note also that $\mathtt{SemFL}^{+}_{\mathtt{ecm}}$ properly contains the positive fragment of Gödel-Dummett logic, so the previous theorem can also be contrasted against the results of Maksimova discussed in Section 3.1.

The previous pair of theorems concern the positive logics $\mathtt{SemFL}^{+}_{\mathtt{cm}}$ and $\mathtt{SemFL}^{+}_{\mathtt{ecm}}$. If we return the falsity constant f to the signature, we of course still retain continuum-many axiomatic extensions of $\mathtt{SemFL}_{\mathtt{cm}}$ with the DIP and in which exchange cannot be derived. On the other hand, $\mathtt{SemFL}_{\mathtt{ecm}}$ has many more extensions with DIP than $\mathtt{SemFL}_{\mathtt{cm}}$. The exact number of these is not known, but it is shown in [27, Proposition 5.14] that $\mathtt{SemFL}_{\mathtt{ecm}}$ has finitely many axiomatic extensions with the DIP and that there are at least 60^4 of these.

Many questions about interpolation in extensions of FL remain open. It is still unknown whether there are uncountably many axiomatic extensions of $\mathtt{FL}_{\mathtt{ew}}$ with the DIP, and relatively little is known about extensions of FL with the CIP, aside from the few previously mentioned cases that are settled by application of Maehara's method.

▶ **Open Question 37.** *Are there uncountably many axiomatic extensions of* $\mathtt{FL}_{\mathtt{ew}}$ *with the DIP?*

▶ **Open Question 38.** *Are there infinitely many axiomatic extensions of* $\mathtt{FL}_{\mathtt{ew}}$ *with the CIP?*

Acknowledgments

The work contained in this chapter was supported by the Czech Science Foundation project 25-18306M, INTERACT. The author would like to thank Johan van Benthem, Nick Bezhanishvili, and Han Gao for their many useful comments on a draft of this chapter.

References

1 S. Aguzzoli and M. Bianchi. Amalgamation property for varieties of BL-algebras generated by one chain with finitely many components. In U. Fahrenberg, M. Gehrke, L. Santocanale, and M. Winter, editors, *Proceedings of the 19th International Conference on Relational and Algebraic Methods in Computer Science (RAMiCS 2021)*, volume 13027 of *LNCS*, pages 1–18. Springer, 2021. doi:10.1007/978-3-030-88701-8_1.

2 S. Aguzzoli and M. Bianchi. Amalgamation property for some varieties of BL-algebras generated by one finite set of BL-chains with finitely many components. In R. Glück, L. Santocanale, and M. Winter, editors, *Proceedings of the 20th International Conference on Relational and Algebraic Methods in Computer Science (RAMiCS 2023)*, volume 13896 of *LNCS*, pages 1–16. Springer, 2023. doi:10.1007/978-3-031-28083-2_1.

3 A.R. Anderson and N.D. Belnap. *Entailment: The Logic of Relevance and Necessity, Vol. I*. Princeton University Press, 1975.

4 A.R. Anderson, N.D. Belnap, and J.M. Dunn. *Entailment: The Logic of Relevance and Necessity, Vol. II*. Princeton University Press, 1992.

5 A. Avron. On an implication connective of RM. *Notre Dame J. Form. Log.*, 27(2):201–209, 1986. doi:10.1305/ndjfl/1093636612.

6 A. Avron. The semantics and proof theory of linear logic. *Theoretical Computer Science*, 57(2-3):161–184, 1988. doi:10.1016/0304-3975(88)90037-0.

7 M. Baaz and H. Veith. Interpolation in fuzzy logic. *Arch. Math. Logic*, 38:461–489, 1999. doi:10.1007/s001530050151.

8 J. van Benthem. Modal foundations for predicate logic. *Logic Journal of the IGPL*, 5(2):259–286, March 1997. doi:10.1093/jigpal/5.2.259.

9 Q. Blomet. Split interpolation: Refining Craig's theorem via three-valued logics. *Studia Logica*, 2025. doi:10.1007/s11225-025-10213-w.

10 G. Boolos. On systems of modal logic with provability interpretations. *Theoria*, 46(1):7–18, 1980. doi:10.1111/j.1755-2567.1980.tb00686.x.

11 R. Brady. *Relevant Logics and their Rivals, Volume II: A Continuation of the Work of Richard Sylvan, Robert Meyer, Val Plumwood, and Ross Brady*. Ashgate, Aldershot, UK, 2003. With contributions by M. Bunder, A. Fuhrmann, A. Loparić, E. Mares, C. Mortensen, and A. Urquhart.

12 A. Chagrov and M. Zakharyaschev. *Modal Logic*. Oxford University Press, 1996. doi:10.1093/oso/9780198537793.001.0001.

13 R. Cignoli. The algebras of Łukasiewicz many-valued logic: A historical overview. In S. Aguzzoli, A. Ciabattoni, B. Gerla, C. Manara, and V. Marra, editors, *Algebraic and Proof-theoretic Aspects of Non-classical Logics*, volume 4460 of *LNAI*, pages 69–83. Springer, 2007. doi:10.1007/978-3-540-75939-3_5.

14 R. Cignoli, I. D'Ottaviano, and D. Mundici. *Algebraic foundations of many-valued reasoning*, volume 7 of *Trends Log. Stud. Log. Libr.* Dordrecht: Kluwer Academic Publishers, 2000. doi:10.1007/978-94-015-9480-6.

15 R. Cignoli, F. Esteva, L. Godo, and A. Torrens. Basic fuzzy logic is the logic of continuous t-norms and their residua. *Soft Computing*, 4:106–112, 2000. doi:10.1007/s005000000044.

16 A. Citkin and A. Muravitsky. *Consequence Relations*. Oxford Logic Guides. Oxford University Press, 2022. doi:10.1093/oso/9780192866417.001.0001.

17 T. Cortonesi, E. Marchioni, and F. Montagna. Quantifier elimination and other model-theoretic properties of BL-algebras. *Notre Dame J. Form. Log.*, 52:339–379, 2011. doi:10.1215/00294527-1499336.

18 J. Czelakowski. *Protoalgebraic Logics*. Kluwer Academic Publishers, 2001. doi:10.1007/978-94-017-2807-2_3.

19 P.M. Dekker. Interpolation and Beth definability in implicative fragments of IPC. Bachelor thesis. University of Amsterdam, Institute for Logic, Language and Computation, 2020.

20 A. Di Nola and A. Lettieri. One chain generated varieties of MV-algebras. *J. Algebra*, 225(2):667–697, 2000. doi:10.1006/jabr.1999.8136.

21 J.M. Dunn. A Kripke-style semantics for R-mingle using a binary accessibility relation. *Studia Logica*, 35(2):163–172, 1976. doi:10.1007/bf02120878.

22 F. Esteva and L. Godo. Monoidal t-norm based logic: Toward a logic for left-continuous t-norms. *Fuzzy Sets and Systems*, 125(3):271–288, 2001. doi:10.1016/S0165-0114(01)00098-7.

23 J. Font. *Abstract Algebraic Logic: An Introductory Textbook*. College Publications, 2016.

24 W. Fussner and N. Galatos. Categories of models of R-mingle. *Ann. Pure Appl. Logic*, 170:1188–1242, 2019. doi:10.1016/j.apal.2019.05.003.

25 W. Fussner and N. Galatos. Semiconic idempotent logic I: Structure and local deduction theorems. *Ann. Pure Appl. Logic*, 175, 103442, 2024. doi:10.1016/j.apal.2024.103443.

26 W. Fussner and N. Galatos. Semiconic idempotent logic II: Beth definability and deductive interpolation. *Ann. Pure Appl. Logic*, 176, 103528, 2025. doi:10.1016/j.apal.2024.103528.

27 W. Fussner, G. Metcalfe, and S. Santschi. Interpolation and the exchange rule. Manuscript. Available at https://arxiv.org/abs/2310.14953., 2023.

28 W. Fussner and S. Santschi. Interpolation in linear logic and related systems. *ACM Trans. Comput. Log.*, 25(4):19, 2024. doi:10.1145/3680284.

29 W. Fussner and S. Santschi. Amalgamation in semilinear residuated lattices. *Studia Logica*, 2025. doi:10.1007/s11225-025-10206-9.

30 W. Fussner and S. Santschi. Interpolation in Hájek's basic logic. *Ann. Pure Appl. Logic*, 176, 103615, 2025. doi:10.1016/j.apal.2025.103615.

31 D.M. Gabbay. Semantic proof of Craig's interpolation theorem for intuitionistic logic and its extensions. In R.O. Gandy and C.M.E. Yates, editors, *Logic Colloquium '69*, pages 391–410. North-Holland, 1971.

32 D.M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Studies in Logic and the Foundations of Mathematics. Elsevier, Amsterdam, 2003.

33 D.M. Gabbay and L.L. Maksimova. *Interpolation and Definability: Modal and Intuitionistic Logics*. Oxford, 2005.

34 N. Galatos, P. Jipsen, T. Kowalski, and H. Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Elsevier, 2007. doi:10.1016/s0049-237x(07)x8002-9.

35 G. Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39(3):176–219, 1934. in German.

36 J. Gil-Férez, P. Jipsen, and G. Metcalfe. Structure theorems for idempotent residuated lattices. *Algebra Universalis*, 81(2):25, 2020. doi:10.1007/s00012-020-00659-5.

37 J. Gil-Férez, A. Ledda, and C. Tsinakis. The failure of the amalgamation property for semilinear varieties of residuated lattices. *Math. Slovaca*, 65(4):817–828, 2015. doi:10.1515/ms-2015-0057.

38 J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–101, 1987. doi:10.1016/0304-3975(87)90045-4.

39 V. Giustarini and S. Ugolini. Algebraic structure theory and interpolation failures in semilinear logics. Manuscript. Available at https://arxiv.org/abs/2408.17400., 2024.

40 V. Goranko. The Craig interpolation theorem for propositional logics with strong negation. *Studia Logica*, 44:291–317, 1985. doi:10.1007/bf00394448.

41 P. Hájek. *Metamathematics of Fuzzy Logic*. Trends in Logic. Springer, New York, 1998. doi:10.1007/978-94-011-5300-3.

42 I. Johansson. Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus. *Compositio Mathematika*, 4:119–136, 1937.

43 H. Kihara and H. Ono. Interpolation properties, Beth definability properties and amalgamation properties for substructural logics. *Journal of Logic and Computation*, 20(4):823–875, 2010. doi:10.1093/logcom/exn084.

44 Y. Komori. Super-Łukasiewicz propositional logics. *Nagoya Math. J.*, 84:119–133, 1981. doi:10.1017/s0027763000019577.

45 M. Kracht. Modal consequence relations. In P. Blackburn, J. Van Benthem, and F. Wolter, editors, *Handbook of Modal Logic, vol. 3*, Studies in Logic and Practical Reasoning, pages 491–545. Elsevier, 2007. doi:10.1016/s1570-2464(07)80011-5.

46 J. Łukasiewicz. O logice trówarkościwej. *Ruch Filozoficzny*, 6:170–171, 1920. in Polish.

47 J.X. Madarász. Interpolation and amalgamtion: Pushing the limits. part I. *Studia Logica*, 61:311–345, 1998. doi:10.1023/a:1005064504044.

48 L.L. Maksimova. Craig's theorem in superintuitionistic logics and amalgamable varieties of pseudo-boolean algebras. *Algebra and Logic*, 16(5):427–455, 1977. doi:10.1007/bf01670006.

49 L.L. Maksimova. Interpolation theorems in modal logics and amalgamable varieties of topological Boolean algebras. *Algebra and Logic*, 18(5):556–586, 1979. doi:10.1007/bf01673502.

50 L.L. Maksimova. Amalgamation and interpolation in normal modal logics. *Studia Logica*, 50:457–471, 1991. doi:10.1007/bf00370682.

51 L.L. Maksimova. On a form of interpolation in modal logics. Logic Colloquium '05. *Bull. Symb. Log.*, 12(2):340, 2006. doi:10.2178/bsl/1146620064.

52 L.L. Maksimova. Weak interpolation in extensions of minimal logic. In S. Feferman and W. Sieg, editors, *Proofs, Categories and Computations: Essays in honor of Grigori Mints*, pages 159–170. College Publications, 2010.

53 L.L. Maksimova. Decidability of weak interpolation property over the minimal logic. *Algebra and Logic*, 50(2):106–132, 2011. doi:10.1007/s10469-011-9127-8.

54 L.L. Maksimova. Interpolation and definability over the logic Gl. *Studia Logica*, 99:249–267, 2011. doi:10.1007/s11225-011-9351-1.

55 L.L. Maksimova. Interpolation and the projective Beth property in well-composed logics. *Algebra and Logic*, 51(2):163–184, 2012. doi:10.1007/s10469-012-9180-y.

56 L.L. Maksimova and V.F. Yun. Extensions of the minimal logic and the interpolation problem. *Siberian Mathematical Journal*, 59(4):681–693, 2018. doi:10.1134/s0037446618040109.

57 E. Marchioni and G. Metcalfe. Craig interpolation for semilinear substructural logics. *Mathematical Logic Quaterly*, 58(6):468–481, 2012. doi:10.1002/malq.201200004.

58 G. Metcalfe. A sequent calculus for constructive logic with strong negation as a substructural logic. *Bulletin of the Section of Logic*, 38(1–2):5–12, 2009.

59 G. Metcalfe, F. Montagna, and C. Tsinakis. Amalgamation and interpolation in ordered algebras. *J. Algebra*, 402:21–82, 2014. doi:10.1016/j.jalgebra.2013.11.019.

60 G. Metcalfe, N. Olivetti, and D. Gabbay. *Proof Theory for Fuzzy Logics*. Springer, 2008.

61 G. Metcalfe, F. Paoli, and C. Tsinakis. *Residuated Structures in Algebra and Logic*, volume 277 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2023. doi:10.1090/surv/277.

62 R.K. Meyer. Relevantly interpolating in RM. ANU Logic Group Research Paper No. 9, 1980.

63 P. Milne. A non-classical refinement of the interpolation property for classical propositional logic. *Logique et Analyse*, 235:273–281, 2016.

64 F. Montagna. Interpolation and Beth's property in propositional many-valued logics: a semantic investigation. *Ann. Pure Appl. Logic*, 141(1–2):148–179, 2006. doi:10.1016/j.apal.2005.11.001.

65 T. Nagashima. An extension of the Craig-Schütte interpolation theorem. *Ann. Japan Assoc. Phil. Sci.*, 3:12–18, 1966.

66 T. Nascimento, U. Rivieccio, J. Marcos, and M. Spinks. Nelson's logic S. *Logic Journal of the IGPL*, 28(6):1182–1206, 2020. doi:10.1093/jigpal/jzaa015.

67 D. Nelson. Contructible falsity. *J. Symb. Log.*, 14(1), 1949.

68 S.P. Odintsov. Logic of classic refutability and class of extensions of minimal logic. *Logic and Logical Philosophy*, 9(4):91–107, 2004. doi:10.12775/llp.2001.006.

69 S.P. Odintsov. Transfer theorems for extensions of the paraconsistent Nelson logic. *Algebraic and Logic*, 45(4), 2006. doi:10.1007/s10469-006-0021-8.

70 S.P. Odintsov. *Constructive Negations and Paraconsistency*. Springer, 2008. doi:10.1007/978-1-4020-6867-6.

71 H. Ono and Y. Komori. Logics without the contraction rule. *J. Symb. Log.*, 50:169–201, 1985. doi:10.2307/2273798.

72 A.M. Pitts. On an interpretation of second-order quantification in first-order intuitionistic propositional logic. *J. Symb. Log.*, 57(1):33–52, 1992. doi:10.2307/2275175.

73 G. Priest. The logic of paradox. *J. Philos. Logic*, 8(1):219–241, 1979. doi:10.1007/bf00258428.

74 D. Roorda. Interpolation in fragments of classical linear logic. *J. Symb. Log.*, 59(2):419–444, 1994. doi:10.2307/2275398.

75 R. Routley and M. Meyer. The semantics of entailment. In H. Leblanc, editor, *Truth, Syntax, Modality*, pages 199–243. North-Holland, 1973.

76 L. Santocanale and Y. Venema. Uniform interpolation for monotone modal logic. In L. Beklemishev, V. Goranko, and V. Shehtman, editors, *Advances in Modal Logic 2010*, pages 350–370. CSLI Publications, 2010.

77 S. Santschi and P. Jipsen. Residuated lattices do not have the amalgamation property. Manuscript. Available at https://arxiv.org/abs/2504.16111., 2025.

78 K. Schütte. Der Interpolationssatz der intuitionistischen Prädikatenlogik. *Math. Ann.*, 148:192–200, 1962.

79 A.K. Simspon. *The Proof Theory and Semantics of Intuitionistic Modal Logic*. PhD thesis, University of Edinburgh, 1994.

80 C. Smorynski. Beth's theorem and self-referential sentences. In A. Macintyre, J. Paris, and L. Pacholski, editors, *Logic Colloquium '77*, pages 253–261. North-Holland, 1978.

81 A.A. Tabatabai, R. Iemhoff, and R. Jalali. Uniform Lyndon interpolation for basic non-normal modal logics. In A. Silva, R. Wassermann, and R. de Queiroz, editors, *Logic, Language, Information, and Computation (WoLLIC 2021)*, volume 13028 of *LNCS*, pages 287–301. Springer, 2021. doi:10.1007/978-3-030-88853-4_18.

82 A. Urquhart. Failure of interpolation in relevant logics. *J. Philos. Logic*, 22:449–479, 1993. doi:10.1007/bf01349560.

83 A. Urquhart. Beth's definability theorem in relevant logics. In E. Orlowska, editor, *Logic at Work, Essays dedicated to the Memory of Helena Rasiowa*. Physica-Verlag, 1999.

Chapter 4

Six Proofs of Interpolation for the Modal Logic K

Nick Bezhanishvili
University of Amsterdam, The Netherlands

Balder ten Cate
University of Amsterdam, The Netherlands

Rosalie Iemhoff
Utrecht University, The Netherlands

Abstract

In this chapter, we present six different proofs of Craig interpolation for the modal logic K, each using a different set of techniques (model-theoretic, proof-theoretic, syntactic, automata-theoretic, using quasi-models, and algebraic). We compare the pros and cons of each proof technique.

Contents

1 Introduction **102**
2 Basic Definitions **103**
3 First Proof (Model-Theoretic) **108**
4 Second Proof (Proof-Theoretic) **112**
5 Third Proof (Syntactic) **120**
6 Fourth Proof (Automata-Theoretic) **124**
7 Fifth Proof (via Quasi-Models and Type Elimination Sequences) **130**
8 Sixth Proof (via Modal Algebras and Duality Theory) **136**
9 Lower Bounds **143**
Acknowledgments **144**
References **144**



© Nick Bezhanishvili, Balder ten Cate, and Rosalie Iemhoff;
licensed under Creative Commons License CC BY 4.0

In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 4, pages 101–147. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.d

1 Introduction

A logic has the Craig interpolation property if, whenever an implication $\varphi \to \psi$ is valid, there is a formula ϑ (which we will call an "interpolant") such that $\varphi \to \vartheta$ and $\vartheta \to \psi$ are valid, and such that all the non-logical symbols occurring in ϑ occur both in φ and in ψ. Craig [11] proved that first-order logic has this property. This result was later refined by Lyndon [31], who proved that every valid implication has an interpolant, such that every non-logical symbol occurring positively in the interpolant occurs positively in both the antecedent and the consequent, and likewise for negative occurrences. Several further strengthenings of these interpolation theorems were later obtained for first-order logic. See Chapter 2 for more details. Many modal logics also have Craig interpolation, and indeed, Lyndon interpolation [17]. Some (such as the basic modal logic K) even enjoy a strong form of interpolation that fails for first-order logic, called *uniform interpolation*, whereby the interpolant can be constructed in a uniform way, such that it does not depend on the consequent, but only on the antecedent and shared language (i.e., the set of non-logical symbols shared by the antecedent and consequent).

Chapter 3 provides a broader picture regarding interpolation properties for non-classical logics and the landscape of modal logics with Craig interpolation. In this chapter, we take a close look at one specific interpolation theorem, namely the Craig interpolation theorem for the modal logic K:

▶ **Theorem 1.1** (Craig Interpolation for the Modal Logic K). *Every valid modal implication* $\models \varphi \to \psi$ *has a* Craig interpolant*, that is, a modal formula* ϑ *such that the following hold:*

1. $\models \varphi \to \vartheta$,
2. $\models \vartheta \to \psi$,
3. $\mathrm{sig}(\vartheta) \subseteq \mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi)$,

Note that this version of Craig interpolation for modal logic is sometimes called *arrow-interpolation* (also known as *local interpolation*), as opposed to *turnstile-interpolation* (also known as *global interpolation* or *deductive interpolation*), cf. Chapter 3.

There are many ways to prove this theorem. In this chapter we review six proofs. Each draws on a different set of techniques (model theoretic, proof theoretic, syntactic, automata-theoretic, using quasi-models, and algebraic), and each proof has different advantages and disadvantages. For example, some techniques generalize better to other logics, some techniques allow proving stronger forms of interpolation such as *uniform interpolation*, and some techniques yield an algorithm for producing interpolants of (worst-case) *optimal size*. Thus, different techniques are useful for different purposes. In addition, we hope that the different proofs of interpolation provide the reader with a glimpse at the different angles from which modal logics can be studied more generally.

A Note on Formula Size. One of the questions we will be interested in is whether the different proofs of Craig interpolation yield algorithms for computing interpolants, and, if so,

whether they produce small interpolants. Here, it is important to point out that the *size* of a modal formula can be measured in different ways. The standard definition of formula length is given by the number of symbols needed to write the formula as a string. We denote this by $|\varphi|$. The other notion of size is as the cardinality of the set of its subformulas. We denote this by $|\varphi|_{\mathrm{DAG}}$. It captures the size of the formula when represented succinctly as a directed acyclic graph (DAG). In general, $|\varphi|_{\mathrm{DAG}}$ can be exponentially smaller than $|\varphi|$ [14, 4]. This is illustrated by the following example.

▶ **Example 1.2.** Consider the modal formula χ_n defined inductively as follows:

$$\begin{array}{lcl} \chi_0 & = & (\Box p \vee \Box \neg p) \\ \chi_{i+1} & = & (\Box \chi_i \vee \Box \neg \chi_i) \end{array}$$

In other words, $\chi_n = \blacksquare^n p$, where $\blacksquare \varphi$ is short for the "non-contingency statement" $\Box\varphi \vee \Box\neg\varphi$. Observe that $|\chi_n|_{\mathrm{DAG}} = 4n + 6$ whereas $|\chi| = 14 \cdot 2^n - 6$ (if we include the parentheses). In fact, it has been shown that every modal formula ψ equivalent to χ_n has size $|\psi| \geq 2^n$ [14]. A similar super-polynomial gap between $|\varphi|$ and $|\varphi|_{\mathrm{DAG}}$ is believed to hold already for propositional formulas, but proving it would amount to separating the complexity class NC_1 from $\mathsf{P/poly}$, an open problem in complexity theory [40]. See Chapter 1 for more details. ⌟

As we will see, some of the approaches yield an algorithm for computing a Craig interpolant of singly exponential size, even under the standard formula size metric ($|\varphi|$), while others yield an algorithm for computing a Craig interpolant of singly exponential size under the more permissive DAG-based size metric ($|\varphi|_{\mathrm{DAG}}$). We will discuss matching lower bounds on Craig interpolant size in Section 9.

Outline. In Section 2 we review basic definitions. In the subsequent six sections, we review different proofs of Theorem 1.1. In Section 9 we discuss some lower bound results for computing Craig interpolants.

2 Basic Definitions

Although we assume familiarity with modal logics, we briefly recall some basic definitions in order to fix our notation. See [5] for more details.

Modal Formulas: Syntax and Kripke Semantics

By a *propositional signature* we will mean a set of proposition letters (also known as *propositional variables*). Given a propositional signature τ, the *modal formulas* w.r.t. τ are generated by the following grammar:

$$\chi ::= p \mid \top \mid \bot \mid \neg\chi \mid (\chi \vee \chi) \mid (\chi \wedge \chi) \mid \Diamond\chi \mid \Box\chi$$

Table 1 Kripke semantics for modal logic

$M, w \models p$	iff	$w \in V(p)$
$M, w \models \top$	always	
$M, w \models \bot$	never	
$M, w \models \neg\chi$	iff	$M, w \not\models \chi$
$M, w \models \chi_1 \wedge \chi_2$	iff	$M, w \models \chi_1$ and $M, w \models \chi_2$
$M, w \models \chi_1 \vee \chi_2$	iff	$M, w \models \chi_1$ or $M, w \models \chi_2$
$M, w \models \Diamond\chi$	iff	$M, w' \models \chi$ for some $w' \in W$ with $(w, w') \in R$
$M, w \models \Box\chi$	iff	$M, w' \models \chi$ for all $w' \in W$ with $(w, w') \in R$

where $p \in \tau$. Note that it will be important that we allow $\top$ and $\bot$ as atomic formulas. We will view the implication $\varphi \to \psi$ as a shorthand notation for $\neg\varphi \vee \psi$. We will denote by $\mathrm{sig}(\chi)$ the propositional signature of χ, that is, the set of proposition letters occurring in χ.

A modal formula is in *negation normal form (NNF)* if every negation sign is immediately in front of a proposition letter. In other words, the modal formulas in NNF are generated by the grammar $\chi ::= p \mid \neg p \mid \top \mid \bot \mid \chi_1 \wedge \chi_2 \mid \chi_1 \vee \chi_2 \mid \Diamond\chi \mid \Box\chi$. We denote by $\mathrm{NNF}(\chi)$ the result of bringing a formula χ into NNF (by using the De Morgan laws and the duality laws $\neg\Diamond\chi \equiv \Box\neg\chi$ and $\neg\Box\chi \equiv \Diamond\neg\chi$).

A *Kripke frame* is a pair $F = (W, R)$ where W is a set of *worlds* and $R \subseteq W \times W$. A *Kripke model* for a propositional signature τ is a triple $M = (W, R, V)$ where (W, R) is a Kripke frame (the *underlying frame* of M), and $V : \tau \to \wp(W)$ assigns to every proposition letter a set of worlds. Satisfaction of a modal formula at a world in a Kripke model, denoted $M, w \models \chi$, is defined as usual, cf. Table 1. For a set of formulas Φ, we will write $M, w \models \Phi$ if $M, w \models \varphi$ for all $\varphi \in \Phi$.

We will also refer to a pair (M, w), where $M = (W, R, V)$ is a Kripke model and $w \in W$ a world, as a *pointed Kripke model*. Thus, satisfaction can be viewed as a relation between pointed Kripke models and modal formulas.

Validity, Frame Classes and Logics

We write $\models \chi$ if the modal formula χ is satisfied at every world in every Kripke model. This can be further relativized to a class of Kripke frames (or, *frame class* for short). Let $\mathcal{F}$ be any frame class. Then we write $\mathcal{F} \models \varphi$ if φ is satisfied at every world in every Kripke model whose underlying frame belongs to $\mathcal{F}$. Every frame class induces a *modal logic*. Formally, fix a countably infinite propositional signature τ. Then by a *normal modal logic* we will mean a set L of modal formulas that includes all substitution instances of propositional tautologies as well as all formulas of the form $\Box(\chi_1 \to \chi_2) \to (\Box\chi_1 \to \Box\chi_2)$ and is closed under modus ponens (if $\chi_1 \to \chi_2 \in L$ and $\chi_1 \in L$, then $\chi_2 \in L$), necessitation (if $\chi \in L$, then $\Box\chi \in L$),

Table 2 Standard translation $\mathrm{ST}_x(\cdot)$ from modal logic to first-order logic

$$
\begin{array}{lcl}
\mathrm{ST}_x(p) & = & P(x) \\
\mathrm{ST}_x(\top) & = & \top \\
\mathrm{ST}_x(\bot) & = & \bot \\
\mathrm{ST}_x(\neg\chi) & = & \neg\mathrm{ST}_x(\chi) \\
\mathrm{ST}_x(\chi_1 \wedge \chi_2) & = & \mathrm{ST}_x(\chi_1) \wedge \mathrm{ST}_x(\chi_2) \\
\mathrm{ST}_x(\chi_1 \vee \chi_2) & = & \mathrm{ST}_x(\chi_1) \vee \mathrm{ST}_x(\chi_2) \\
\mathrm{ST}_x(\Diamond\chi) & = & \exists y(R(x,y) \wedge \mathrm{ST}_y(\chi)) \\
\mathrm{ST}_x(\Box\chi) & = & \forall y(R(x,y) \rightarrow \mathrm{ST}_y(\chi))
\end{array}
$$

and substitution (if $\chi \in L$, then every substitution instance of χ belongs to L). For every frame class $\mathcal{F}$, the set of modal formulas $\{\varphi \mid \mathcal{F} \models \varphi\}$ is, in this sense, a modal logic. We will refer to it as *the modal logic of* $\mathcal{F}$. The modal logic of the class of all frames is known as the modal logic K, while, for example, the modal logic of the class of transitive frames is known as the modal logic $\mathsf{K4}$.

Modal Logic as a Fragment of First-Order Logic

A Kripke model with propositional signature τ can be viewed as a first-order structure over a corresponding relational first-order signature σ. More precisely, σ consists of a single binary relation symbol R plus, for each $p \in \tau$, a corresponding unary relation symbol P. It should be clear how every Kripke model with propositional signature τ can be viewed as a first-order structure over the corresponding first-order signature σ. This correspondence between Kripke models and first-order structures is, in fact, so basic and transparent that we will take it for granted and we will allow ourselves to freely treat Kripke models as first-order structures. This will allow us, for instance, to interpret first-order formulas (over the correspondence signature) in Kripke models. Following this perspective, modal logic can be viewed as a fragment of first-order logic, in the following sense: for every modal formula φ there is a first-order formula $\psi(x)$ such that, for all pointed Kripke models (M, w), $M, w \models \varphi$ iff $M, w \models \psi$. Here, by the latter, we mean that M satisfies $\psi(x)$ under the assignment that sends x to w. This first-order equivalent $\psi(x)$ of a modal formula φ can be obtained from φ by means of a simple, inductive, translation, which is known as the *standard translation*. It is given in Table 2.

▶ **Theorem 2.1.** *For all pointed Kripke models* (M, w) *and modal formulas* φ*,* $M, w \models \varphi$ *iff* $M, w \models \mathrm{ST}_x(\varphi)$*.*

There is no such translation from first-order logic back to modal logic. For example, the first-order formula $\varphi(x) = \exists y(R(x,y) \wedge R(y,y) \wedge P(y))$ is not equivalent to any modal formula. This can be proved formally using *bisimulations*.

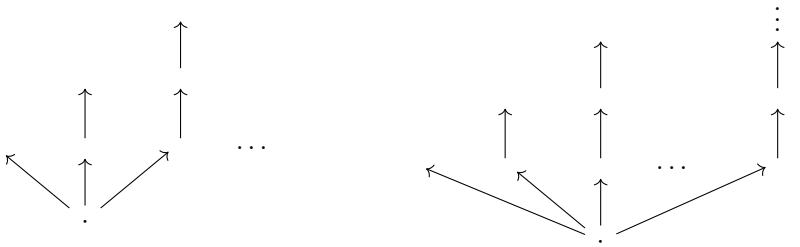


Figure 1 Pointed Kripke models (over the empty propositional signature) that are modally indistinguishable but not bisimilar. The first has branches of all finite lengths. The second has, in addition, an infinite branch.

Given two Kripke models $M = (W, R, V), M' = (W', R', V')$ for the same propositional signature τ, a *τ-bisimulation* between M and M' is a binary relation $Z \subseteq W \times W'$ such that the following conditions hold for all $(w, w') \in Z$:

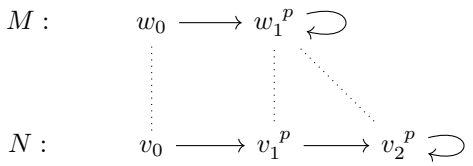
Atomic Harmony For all $p \in \tau$, $w \in V(p)$ iff $w' \in V'(p)$,

Forth For each $(w, v) \in R$ there is a $(w', v') \in R'$ such that $(v, v') \in Z$,

Back For each $(w', v') \in R'$ there is a $(w, v) \in R$ such that $(v, v') \in Z$.

A τ-bisimulation Z between $M = (W, R, V)$ and $M' = (W', R', V')$ is said to be *full* if for each $w \in W$ there is a $w' \in W'$ with $(w, w') \in Z$, and, conversely, for each $w' \in W'$ there is a $w \in W$ with $(w, w') \in Z$. We write $(M, w) \underline{\leftrightarrow}_\tau (M', w')$, and we say that (M, w) is *τ-bisimilar* to (M', w'), if there is a τ-bisimulation Z between M and M' with $(w, w') \in Z$.

▶ **Example 2.2.** Let $\tau = \{p\}$ and consider the following pointed Kripke models:



Then (M, w_0) and (N, v_0) are τ-bisimilar. As bisimulation we can pick the binary relation $Z = \{(w_0, v_0), (w_1, v_1), (w_1, v_2)\}$ as indicated by the dotted lines. ⌟

It is a well-known fact that modal formulas are bisimulation-invariant.

▶ **Theorem 2.3.** *Let (M, w) and (M', w') be pointed Kripke models with $(M, w) \underline{\leftrightarrow}_\tau (M', w')$ for some propositional signature τ. Then for all modal formulas χ with $\mathrm{sig}(\chi) \subseteq \tau$, we have that $M, w \models \chi$ iff $M', w' \models \chi$.*

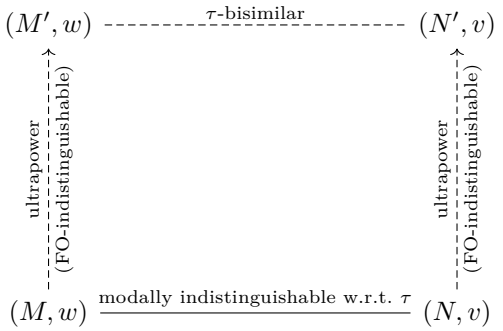


Figure 2 Diagrammatic depiction of Theorem 2.5.

In other words, if two pointed Kripke models (M, w) and (M', w') are τ-bisimilar, then they are *modally indistinguishable w.r.t.* τ, by which we mean that they satisfy the same modal formulas in the propositional signature τ.

▶ **Example 2.4.** We can use Theorem 2.3, in combination with the Kripke models from Example 2.2, to show that the FO-formula $\varphi(x) = \exists y(R(x, y) \wedge R(y, y) \wedge P(y))$ is not equivalent to (the standard translation of) any modal formula in the propositional signature $\tau = \{p\}$. Indeed, if there were such a modal formula, it would have to be true in (M, w_0) but false in (N, v_0). But this is impossible, as the two are τ-bisimilar. ⌟

The converse of Theorem 2.3 does not hold: modal indistinguishability does not imply the existence of a bisimulation.[1] Figure 1 depicts a well-known example of pointed Kripke models that are modally indistinguishable w.r.t. τ but not τ-bisimilar (where we can even pick $\tau = \emptyset$). The following "detour theorem" provides a weaker form of converse. We will make use of it in one of our proofs.

▶ **Theorem 2.5.** *For all pointed Kripke models (M, w) and (N, v) over the same propositional signature τ, the following are equivalent:*

1. *(M, w) and (N, v) are modally indistinguishable w.r.t. τ,*
2. *There exist pointed Kripke models (M', w) and (N', v), where (M', w) is an ultrapower of (M, w) and (N', v) is an ultrapower of (N, v), such that $(M', w) \leftrightarrows_\tau (N', v)$.*

Ultrapowers are a well-known construction in model theory (cf. [21]). The precise definition of ultrapowers is not relevant here. The only property that will be important for us is the following: if B is an ultrapower of a structure A, then A and B are indistinguishable for

[1] By the Hennessy-Milner Theorem, the converse does hold for "image-finite" models, i.e., Kripke models in which each world has at most finitely many successors.

first-order logic. More precisely, if a pointed Kripke structure (M', w) is an ultrapower of a pointed Kripke structure (M, w), then (M', w) and (M, w) are indistinguishable for first-order formulas $\varphi(x)$. In particular, this also means that (M', w) and (M, w) are modally indistinguishable. Theorem 2.5 shows that, although modal indistinguishability does not imply bisimulation, it implies the existence of bisimilar ultrapowers (cf. Figure 2). We refer to [5] for more details and for a proof of Theorem 2.5.

3 First Proof (Model-Theoretic)

Model-theoretic proofs of Craig interpolation typically proceed by contraposition: from the assumption that there is no Craig interpolant for $\varphi \to \psi$, we derive that $\varphi \to \psi$ is not valid by constructing a model for $\varphi \wedge \neg\psi$. Gabbay's original proof of Craig interpolation for modal logic [17] falls in this category. The construction of the model for $\varphi \wedge \neg\psi$ typically involves a method for amalgamating a suitable pair of pointed Kripke models $M, w \models \varphi$ and $N, v \models \neg\psi$ into a single pointed Kripke model satisfying $\varphi \wedge \neg\psi$. The *bisimulation product* construction (also known as *zigzag product*), originally introduced in [35] (see also [36]), is such an amalgamation method. Relationships between interpolation properties and amalgamation properties have also been studied from the point of view of algebraic semantics, cf. the long line of work by Maksimova [32, 33, 34]. See also [22, 45] and Section 8 of this chapter.

Before we introduce bisimulation products, we discuss a model-theoretic criterion (partly inspired by [2]) characterizing the existence of an interpolant for a given implication. Given modal formulas φ, ψ and a propositional signature τ, we will say that φ *entails* ψ *along* τ*-bisimulations* if the following holds: for every pair of pointed Kripke structures (M, w) and (N, v), if $M, w \models \varphi$ and $(M, w) \leftrightarrows_\tau (N, v)$, then $N, v \models \psi$. We can think of this as a strong form of entailment: it says when $M, w \models \varphi$, this implies that ψ is true not only in (M, w) but in every pointed Kripke model that is τ-bisimilar to (M, w).

▶ **Proposition 3.1** (Interpolant Existence Criterion). *For all modal formulas φ and ψ, the following are equivalent:*

1. *$\varphi \to \psi$ is a valid implication that has a Craig interpolant.*
2. *φ entails ψ along $(\mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi))$-bisimulations*

Proof. The direction from 1 to 2 is easy: let ϑ be the interpolant. Then $M, w \models \varphi$ implies $M, w \models \vartheta$, which implies $N, v \models \vartheta$ (by Theorem 2.3), which implies $N, v \models \psi$. For the direction from 2 to 1, we reason by contraposition. First, suppose that $\varphi \to \psi$ is not valid, i.e., there is a pointed Kripke stucture $M, w \models \varphi \wedge \neg\psi$. Then, clearly, φ does not entail ψ along $(\mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi))$-bisimulations (because the identity relation is a bisimulation from (M, w) to itself). Next, suppose that $\varphi \to \psi$ is valid but does not have a Craig interpolant.

Claim 1: There are pointed Kripke models $M, w \models \varphi$ and $N, v \models \neg\psi$ such that (M, w) and (N, v) are modally indistinguishable w.r.t. $\mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi)$.

Proof of claim: a standard argument involving a double application of the *compactness theorem*. The compactness theorem for modal logic states that if a set Σ of modal formulas is unsatisfiable, then some finite subset of Σ is already unsatisfiable. Let Σ be the set of all modal formulas in signature $\mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi)$ that are entailed by φ. Clearly, every finite subset of $\Sigma \cup \{\neg\psi\}$ is satisfiable (otherwise the conjunction of the respective formulas from Σ would be a Craig interpolant). Therefore, by compactness, there is a pointed Kripke model $N, v \models \Sigma \cup \{\neg\psi\}$. Next, let Γ be the set of all modal formulas in signature $\mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi)$ true in N, v. Clearly, every finite subset of $\Gamma \cup \{\varphi\}$ is satisfiable (otherwise the negation of the conjunction of the respective formulas from Γ would have been a formula belonging to Σ, contradicting the fact that $N, v \models \Sigma$). It follows again by compactness that there is a pointed Kripke model $M, w \models \Gamma \cup \{\varphi\}$. By construction, we have that $M, w \models \varphi$ and $N, v \models \neg\psi$ and the two pointed Kripke models satisfy the same modal formulas in the signature $\mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi)$.

Next, Theorem 2.5 allows us to "upgrade" (M, w) and (N, v) to FO-equivalent pointed Kripke models (M', w) and (N', v) such that $M', w \models \varphi$ and $N', v \models \neg\psi$ that are bisimilar w.r.t. $\mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi)$. Therefore, φ does not entail ψ along $(\mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi))$-bisimulations. ◀

Proposition 3.1 holds not only for the modal logic K, but for *every* normal modal logic defined by first-order frame conditions (by the same proof, since ultrapowers preserve all FO sentences, and the modal logic of a first-order frame class is always compact). Indeed, without going into details, it is worth mentioning that Proposition 3.1 holds true for every *canonical* modal logic [35].[2] In light of Proposition 3.1, in order to prove the Craig interpolation property, it suffices to prove that, whenever a modal implication $\varphi \to \psi$ is valid, then φ entails ψ along $(\mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi))$-bisimulations. This is what we will use bisimulation products for.

▶ **Definition 3.2** (Bisimulation Product). *Let $M = (W, R, V)$ and $M' = (W', R', V')$ be Kripke models over the same propositional signature.*

- *The* direct product *$M \times N$ is the Kripke model whose domain is the cartesian product $W \times W'$; whose accessibility relation consists of all pairs of pairs $((w_1, u_1), (w_2, u_2))$ such that $(w_1, w_2) \in R$ and $(u_1, u_2) \in R'$; and where a pair (w, u) belongs to the valuation of a proposition letter p iff $w \in V(p)$ and $u \in V'(p)$.*
- *A* subdirect product *of M and N is any induced substructure of $M \times N$.[3] The subdirect product of M and N with domain $Z \subseteq W \times W'$ is denoted by $M \times_Z N$.*

---

[2] Remarkably, this is the case regardless of whether the logic has the Craig interpolation property. Indeed, for certain modal logics lacking the Craig interpolation property, Proposition 3.1 can be used to decide whether a Craig interpolant exists for a given valid implication, cf. Chapter 10.

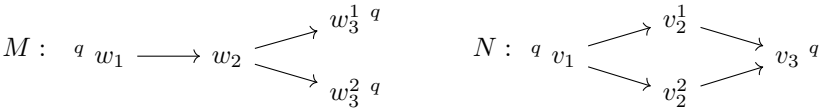
[3] By an *induced substructure* of a Kripke model $M = (W, R, V)$ we mean a Kripke model $M' = (W', R', V')$ where $W' \subseteq W$ and where R' and V' are obtained by restricting R and V to W'.

- *A* bisimulation product *of M and N is a subdirect product $M \times_Z N$ such that Z is a bisimulation between M and N. If Z is a full bisimulation, we also call $M \times_Z N$ a* full bisimulation product.

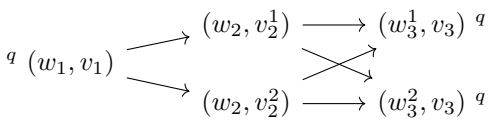
The same definitions apply also to Kripke frames (without the valuations).[4]

Recall that a *full bisimulation* is a bisimulation whose domain is the full domain of A and whose range is the full domain of B. The fullness requirement will in fact not play a major role in what follows. However, we should point out that the most common definition of a bisimulation product in prior literature [35, 36] is in fact what we call full bisimulation products here.

▶ **Example 3.3.** Consider the following two Kripke models in the signature $\{q\}$.



The binary relation $Z = \{(w_1, v_1), (w_2, v_2^1), (w_2, v_2^2), (w_3^1, v_3), (w_3^2, v_3)\}$ is a bisimulation (indeed, a full bisimulation), and the bisimulation product $M \times_Z N$ is the following Kripke model:



⌟

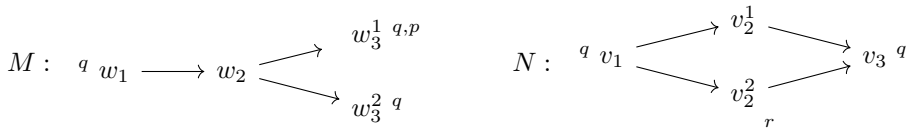
As promised, bisimulation products provide an amalgamation method. The precise amalgamation property we need is the following:

▶ **Theorem 3.4** ([35]). *Whenever $(M_1, w_1) \underline{\leftrightarrow}_{\sigma\cap\tau} (M_2, w_2)$, there is a pointed Kripke model (N, v) such that $(N, v) \underline{\leftrightarrow}_{\sigma} (M_1, w_1)$ and $(N, v) \underline{\leftrightarrow}_{\tau} (M_2, w_2)$.*

Proof. Let $M_1 = (W_1, R_1, V_1)$ and $M_2 = (W_2, R_2, V_2)$, and let Z be a $\sigma \cap \tau$-bisimulation between M_1 and M_2 such that $(w_1, w_2) \in Z$. Let $M_1 \times_Z M_2$ be the bisimulation product in signature $\sigma \cap \tau$. Finally, we expand $M_1 \times_Z M_2$ to a Kripke model $N = (W, R, V)$ in the full signature $\sigma \cup \tau$ by extending the valuation function V of $M_1 \times_Z M_2$ such that $V(p) = \{(u, v) \in Z \mid u \in V_1(p)\}$ for $p \in \sigma \setminus \tau$, and setting $V(p) = \{(u, v) \in Z \mid v \in V_2(p)\}$ for $p \in \tau \setminus \sigma$. It can be verified that the relations $Z_1 = \{((u, v), u) \mid (u, v) \in Z\}$ and $Z_2 = \{((u, v), v) \mid (u, v) \in Z\}$ are bisimulations from the structure thus obtained to M_1 and M_2 respectively, with respect to the signature σ and τ, respectively. ◀

[4] For Kripke frames, (full) bisimulation products can be equivalently defined as subdirect products whose natural projections are (surjective) bounded morphisms.

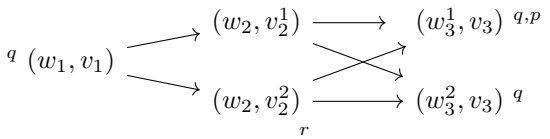
▶ **Example 3.5.** This example builds on Example 3.3. Consider the following two Kripke models in the signature $\sigma = \{p, q\}$ and $\tau = \{q, r\}$, respectively:



Observe that the reducts of M and N to the common signature $\sigma \cap \tau$ are precisely the Kripke models in Example 3.3. As we saw in Example 3.3, the binary relation

$$Z = \{(w_1, v_1), (w_2, v_2^1), (w_2, v_2^2), (w_3^1, v_3), (w_3^2, v_3)\}$$

is a bisimulation with respect to the common signature $\sigma \cap \tau$. Following the proof of Theorem 3.4, to amalgamate the two structures, we take the bisimulation product with respect to the common signature, and we expand it to the full signature $\sigma \cup \tau$ by interpreting the proposition letters in $\sigma \setminus \tau$ according the left projection and interpreting the proposition letters in $\tau \setminus \sigma$ according to the right projection. In this way, we obtain:



It can be verified that this amalgamated structure is bisimilar to M w.r.t. σ and bisimilar to N w.r.t. τ. ⌟

Proof of Craig interpolation for modal logic. Let φ, ψ be modal formulas such that $\varphi \to \psi$ does *not* have a Craig interpolant. Then, by Proposition 3.1, there are pointed Kripke models $M, w \models \varphi$ and $N, v \models \neg\psi$ that are bisimilar w.r.t. $\mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi)$. By Theorem 3.4, there is a pointed Kripke model that is bisimilar in signature $\mathrm{sig}(\varphi)$ to (M, w) and bisimilar in signature $\mathrm{sig}(\psi)$ to (N, v). It immediately follows that this pointed Kripke structure satisfies $\varphi \wedge \neg\psi$ and therefore $\varphi \to \psi$ is not a valid modal implication. ◀

This approach generalizes to the modal logic of any frame class defined by first-order frame conditions that are preserved by bisimulation products, or, alternatively, by full bisimulation products and generated subframes (since a bisimulation can always be "upgraded" to a full bisimulation by taking generated submodels). This includes all frame classes defined by universal Horn sentences, since these are closed under subdirect products and bisimulations are an instance of subdirect products. See also Section 8. In [6], a precise syntactic characterization is given of the first-order frame conditions that are preserved by bisimulation products (which forms a proper generalization of universal Horn sentences). In order to state this

characterization, let a *strict p-formula* be a first-order formula obtained from atomic formulas (including equality statements, $\top$ and $\bot$) using conjunction, disjunction, existential quantifiers, and bounded universal quantifiers of the form $\forall y(Rxy \to \cdot)$, where x and y are distinct variables.

▶ **Theorem 3.6** ([6]). *A first-order sentence φ is preserved by bisimulation products iff φ is equivalent to a conjunction of sentences of the form $\forall \boldsymbol{x}(\psi \to \chi)$, where ψ is a strict p-formula and χ is an atomic formula.*

Many interesting first-order frame conditions are covered by this. For instance, all universal Horn conditions. A notable exception is the Church–Rosser condition $\forall xyz(Rxy \land Ryz \to \exists u(Ryu \land Rzu))$. Indeed, the modal logic of the frame class defined by this condition lacks Craig interpolation (cf. [6]). The modal logic $\mathsf{K.Alt}_2$ (i.e., the logic of the class of frames in which every world has at most two successors) is an example of a modal logic that is not closed under bisimulation products but still has Craig interpolation (cf. [25]).

In summary:

- ✓ The model theoretic proof establishes Craig interpolation for the modal logics of many first-order definable frame classes.
- ✗ The model theoretic proof does not apply to non-elementary logics such as GL and Grz, nor to non-elementary language extensions such as the modal μ-calculus.
- ✗ This proof technique does not extend in any obvious way to prove uniform interpolation.
- ✗ The model theoretic proof does not yield an effective method for constructing Craig interpolants.

4 Second Proof (Proof-Theoretic)

In this section we turn to a proof-theoretic proof of Craig interpolation for K. The proof is called proof-theoretic because it makes use of a proof system for K, in this case a sequent calculus, which is introduced below. Proof-theoretic proofs usually have nice benefits:[5] most of them, and this certainly holds for the proof in this section, are constructive in that they contain a method to explicitly construct an interpolant from the proof of the implication that is interpolated. Another benefit is that they easily generalize to proof-theoretic proofs of CIP for other logics with a similar proof system. Also, strengthenings of the interpolation property are usually easy to obtain from proof-theoretic proofs, we will see examples of this in Section 4.7 and in Chapter 5.

[5] We do not mean to imply that these benefits only hold for proof-theoretic proofs, other methods of proof may have similar or other benefits.

The proof system of choice in this section is a *sequent calculus* (G3K), a proof system that derives sequents instead of formulas. *Sequents* are expressions of the form $(\Gamma \Rightarrow \Delta)$, where Γ and Δ are finite multisets of modal formulas. Their intended interpretation is $I(\Gamma \Rightarrow \Delta) \equiv_{df} (\bigwedge \Gamma \rightarrow \bigvee \Delta)$. Because of the choice for sequents rather than formulas, rules for many logical operators (connectives, modalities) can be expressed elegantly, and the sequent calculus G3K in this section is an example of this phenomenon. To prove that K has interpolation, the notion is first generalized to sequents in Section 4.3 under the name sequent interpolation. Then it is shown in Section 4.6 that the calculus G3K has sequent interpolation which, as we will see, implies that K has interpolation. This method of proof is called the *Maehara method*, named after its inventor. A proof for classical logic that uses that method can be found in [41].

We prove that G3K has sequent interpolation by showning that it has rule interpolation, which roughly means the following. The proof that a derivable sequent has a sequent interpolant proceeds by induction on the depth of the derivation of the sequent: the induction hypothesis assumes that the premise(s) of the last inference have interpolants and then it is shown that so has the conclusion. Rule interpolation means that for each rule separately the interpolant for the conclusion can be obtained from those of the premises, independent of the rest of the proof, it can be obtained *locally*. The precise formulation can be found in Section 4.3. Usually, in proofs of CIP that use sequent calculi, this locality of the definition of interpolants is not explicitly mentioned, although in most cases the local version is what is actually proven.

4.1 Preliminaries

Sequent Calculi

We denote finite multisets of formulas by $\Gamma, \Pi, \Delta, \Sigma$. $\Gamma \cup \Pi$, also denoted Γ, Π, is the multiset that contains all formulas of Γ and Π (as multisets). A *sequent* is an expression $(\Gamma \Rightarrow \Delta)$, with the intended interpretation $I(\Gamma \Rightarrow \Delta) \equiv_{df} (\bigwedge \Gamma \rightarrow \bigvee \Delta)$, where $\bigwedge \varnothing \equiv_{df} \top$ and $\bigvee \varnothing \equiv_{df} \bot$. A *(sequent) rule* is a pair consisting of a sequence of sequents (the premises) and one sequent (the conclusion). It is an *(sequent) axiom* if the set of premises is empty. A *sequent calculus* G consists of a set of axioms and rules. Thus a sequent calculus is like a usual proof system but about sequents instead of formulas.

As in the introduction of this section, the signature of a multiset Γ consists of all propositional letters that occur in some formula in Γ and is denoted $\mathrm{sgn}(\Gamma)$. For sequents, $\mathrm{sgn}(\Gamma \Rightarrow \Delta) \equiv_{df} \mathrm{sgn}(\Gamma) \cup \mathrm{sgn}(\Delta)$. The *common language* of two sequents S_1 and S_2 is $\mathrm{sgn}(S_1) \cap \mathrm{sgn}(S_2)$. A positive (negative) occurence of a prositional letter in a formula is defined as usual. A prositional letter occurs *positively (negatively)* in a sequent S whenever it occurs positively (negatively) in $I(S)$.

$$\frac{}{\Gamma, p \Rightarrow p, \Delta}\,\mathrm{Ax_{id}} \qquad \frac{}{\Gamma, \bot \Rightarrow \Delta}\,\mathrm{Ax_{\bot}} \qquad \frac{}{\Gamma \Rightarrow \top, \Delta}\,\mathrm{Ax_{\top}}$$

$$\neg_r\ \frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma \Rightarrow \neg\varphi, \Delta} \qquad \neg_l\ \frac{\Gamma \Rightarrow \varphi, \Delta}{\Gamma, \neg\varphi \Rightarrow \Delta}$$

$$\wedge_r\ \frac{\Gamma \Rightarrow \varphi, \Delta \qquad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \wedge \psi, \Delta} \qquad \wedge_l\ \frac{\Gamma, \varphi, \psi \Rightarrow \Delta}{\Gamma, \varphi \wedge \psi \Rightarrow \Delta}$$

$$\vee_r\ \frac{\Gamma \Rightarrow \varphi, \psi, \Delta}{\Gamma \Rightarrow \varphi \vee \psi, \Delta} \qquad \vee_l\ \frac{\Gamma, \varphi \Rightarrow \Delta \qquad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \vee \psi \Rightarrow \Delta}$$

Figure 3 Sequent calculus G3p

$$R_{\Box}\ \frac{\Gamma \Rightarrow \varphi}{\Pi, \Box\Gamma \Rightarrow \Box\varphi, \Delta}$$

Figure 4 Sequent calculus G3K: G3p plus the rule $R_{\Box}$.

The Calculus G3K

Calculus G3K consists of the calculus G3p (Figure 3) plus the modal rule $R_{\Box}$ (Figure 4). G3p is the standard calculus without structural rules for CPC from [42], which is a restriction to the propositional language of the calculus G3 that already occurs in [27, Chapter XV] under that same name. For simplicity we leave $\Diamond$ out of the language and treat it as an abbreviation of $\neg\Box\neg$, so that the only modal rules we need are for the $\Box$-operator. When convenient, axioms are sometimes called zero-premise rules.

A *derivation* in G3K of a sequent S is a finite tree labelled with sequents such that the root is labelled with S and for any node with label S' that is not a leaf there is a(n instance of a) rule in G3K such that its conclusion is S' and its premises are exactly the direct successors of S' in the tree. The derivation is a *proof* once all leafs are (instances of) axioms. S *has a proof* or *is provable* in G3K, denoted $\vdash S$, if there is a proof in G3K that ends with S. The *depth* of a proof is the length of its longest branch. Thus axioms are proofs of depth one.

The *principal*, *active*, and *auxiliary formulas* of a rule are defined as usual for the rules in G3p: the displayed formula in the conclusion is *principal*, and the displayed formulas in the premises are the *active* formulas. The formulas in Γ and Δ are *auxiliary*. In axioms $\mathrm{Ax_{\bot}}$ and $\mathrm{Ax_{\top}}$ formulas $\bot$ and $\top$ are *principal*, respectively, and in axiom $\mathrm{Ax_{id}}$ both p are *principal*. For $R_{\Box}$ the principal formula is $\Box\varphi$ and all formulas in $\Box\Gamma$ in the conclusion and all formulas in Γ as wel as φ in the premise are active. The auxiliary formulas are the formulas in Π and Δ.

The following theorem establishes that proof system G3K is sound and complete with respect to logic K, a proof can be found in e.g. [38].

▶ **Theorem 4.1.** *For any sequent S: S is provable in* G3K *if and only if $I(S)$ holds in* K.

The following well-known and easy-to-prove fact about G3K, which we need below, is its closure under weakening. For a proof, see [42].

▶ **Lemma 4.2.** *(Weakening lemma)*
If $\vdash \Gamma \Rightarrow \Delta$, then $\vdash \Gamma, \varphi \Rightarrow \Delta$ and $\vdash \Gamma \Rightarrow \varphi, \Delta$.

4.2 Sequent Interpolation

As explained at the beginning of this section, there is a generalization of Craig interpolation to sequents and sequent calculi such that whenever a logic has a sequent calculus that satisfies this sequent interpolation, the logic has Craig interpolation. That K has interpolation is shown by proving that G3K has sequent interpolant, as is done in Section 4.6.

4.3 Split Sequents

For the generalization of interpolation to sequents, we need the notion of split: a *split* of a sequent $\Gamma \Rightarrow \Delta$ is an expression $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2$ (called a *split sequent*) such that $\Gamma = \Gamma_1 \cup \Gamma_2$ and $\Delta = \Delta_1 \cup \Delta_2$ and $\Gamma_1; \Gamma_2$ and $\Delta_1; \Delta_2$ are ordered pairs, so that $\Gamma_1; \Gamma_2$ is in general not equal to $\Gamma_2; \Gamma_1$. The formulas in Γ_1, Δ_1 belong to the *left split* and the formulas in Γ_2, Δ_2 belong to the *right split*. Let $\mathcal{P}s$ denote the set of all split sequents that consist of formulas in our modal language. We denote split sequents with S^p or S^p_i.

Using the terminology from [19], a formula χ is a *split-interpolant* for split $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2$ if $\vdash \Gamma_1 \Rightarrow \chi, \Delta_1$ and $\vdash \Gamma_2, \chi \Rightarrow \Delta_2$ and χ is in the common language of $\Gamma_1 \Rightarrow \Delta_1$ and $\Gamma_2 \Rightarrow \Delta_2$, i.e. $\mathrm{sgn}(\chi) \subseteq \mathrm{sgn}(\Gamma_1 \Rightarrow \Delta_1) \cap \mathrm{sgn}(\Gamma_2 \Rightarrow \Delta_2)$. We say *interpolant* instead of *split-interpolant* when the context determines which interpolant we mean, which is most of the time. A sequent calculus has *sequent interpolation* if any split of any derivable sequent has an interpolant.

4.4 Rule Interpolation

The proof that K has CIP follows from a stronger statement, Lemma 4.3, that states that every rule in G3K interpolates, which is defined as follows. Here axioms are considered to be zero-premise rules. Let $\mathcal{F}$ denote the set of all formulas in our modal language. A one-premise (two- and zero-premise) rule R *interpolates* if there exists a function $\delta_R : \mathcal{F} \times \mathcal{P}s \to \mathcal{F}$ ($\delta_R : \mathcal{F} \times \mathcal{F} \times \mathcal{P}s \to \mathcal{F}$ and $\delta_R : \mathcal{P}s \to \mathcal{F}$ in the two-premise and zero-premise case resp.) such that for every instance of the rule with conclusion S and premise(s) S_1 (and S_2), for every partition S^p of S, there exist splits S^p_1 (and S^p_2) of the premises such that for any interpolant(s) χ_1 of S^p_1 (and χ_2 of S^p_2) formula $\delta_R(\chi_1, S^p)$ $\big(\delta_R(\chi_1, \chi_2, S^p)$ and $\delta_R(S^p)$ in the two-premise and zero-premise case resp.$\big)$ is an interpolant of S^p. Such a function δ_R is an *interpolant function* for R. Note that the domain of the delta-function for axioms is $\mathcal{P}s$, so that in this case the interpolant of the conclusion only depends on the split of the conclusion, as it should be since in the case of axioms there are no premises.

$$LL(\mathrm{Ax_{id}})\ \frac{}{\boldsymbol{p}, \Gamma_1; \Gamma_2 \overset{\bot}{\Rightarrow} \boldsymbol{p}, \Delta_1; \Delta_2} \qquad LR(\mathrm{Ax_{id}})\ \frac{}{\boldsymbol{p}, \Gamma_1; \Gamma_2 \overset{p}{\Rightarrow} \Delta_1; \Delta_2, \boldsymbol{p}}$$

$$RL(\mathrm{Ax_{id}})\ \frac{}{\Gamma_1; \Gamma_2, \boldsymbol{p} \overset{\neg p}{\Rightarrow} \boldsymbol{p}, \Delta_1; \Delta_2} \qquad RR(\mathrm{Ax_{id}})\ \frac{}{\Gamma_1; \Gamma_2, \boldsymbol{p} \overset{\top}{\Rightarrow} \Delta_1; \Delta_2, \boldsymbol{p},}$$

$$L(\mathrm{Ax_{\bot}})\ \frac{}{\bot, \Gamma_1; \Gamma_2 \overset{\bot}{\Rightarrow} \Delta_1; \Delta_2} \qquad R(\mathrm{Ax_{\bot}})\ \frac{}{\Gamma_1; \Gamma_2, \bot \overset{\top}{\Rightarrow} \Delta_1; \Delta_2}$$

$$L(\mathrm{Ax_{\top}})\ \frac{}{\Gamma_1; \Gamma_2 \overset{\bot}{\Rightarrow} \top, \Delta_1; \Delta_2} \qquad R(\mathrm{Ax_{\top}})\ \frac{}{\Gamma_1; \Gamma_2 \overset{\top}{\Rightarrow} \Delta_1; \Delta_2, \top}$$

Figure 5 Rule interpolation for the axioms of G3K

4.5 Rule Interpolation for G3K

In Figures 5 and 6 we define for every rule of G3K a function δ_R that we prove to be an interpolation function of R in Lemma 4.3, in the following way. For the axioms we distinguish cases according to the principal formula(s) being in the left or the right side of the split. Note that for axiom $\mathrm{Ax_{id}}$ there are four cases because it has two principal formulas. For a given split S^p of axiom R, consider in Figure 5 that particular split of R and let $\delta_R(S^p)$ be the formula on top of that sequent arrow. For example,

$$\delta_{\mathrm{Ax_{id}}}(\Gamma_1; \Gamma_2, p \Rightarrow p, \Delta_1; \Delta_2) = \neg p.$$

For the rules, Figure 6, we distinguish cases according to the presence of the principal formula in the left or the right side of the split. As one can see, given such a split of the conclusion, the split that is chosen for the premise(s) is in all cases the one in which the active formula(s) is (are) in the same part of the split as the principal formula, and all other formulas remain as they are. The formula(s) on top of the sequent arrow(s) in the premise(s) is (are) the assumed interpolant(s) χ_1 (and χ_2) of the split premises and $\delta_R(\chi_1, S^p)$ $\big(\delta_R(\chi_1, \chi_2, S^p)\big)$ is defined to be the formula on top of the sequent arrow in the conclusion. For example,

$$\delta_{R_{\Box}}\big(\chi, (\Pi_1, \Box\Gamma_1; \Pi_2, \Box\Gamma_2 \Rightarrow \Delta_1; \Delta_2, \Box\varphi)\big) = \Box\chi.$$

Note that the definition of the interpolant function in Figures 5 and 6 is schematic in the sense that for the definition of δ_R we do not have to distinguish different instances of rule R, we just distinguish according to the location of the principal formula(s) in the split of the conclusion.

$$L(\neg_l)\ \frac{\Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \varphi, \Delta_1;\Delta_2}{\neg\varphi, \Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2} \qquad R(\neg_l)\ \frac{\Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2, \varphi}{\Gamma_1;\Gamma_2, \neg\varphi \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2}$$

$$L(\neg_r)\ \frac{\varphi, \Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2}{\Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \neg\varphi, \Delta_1;\Delta_2} \qquad R(\neg_r)\ \frac{\Gamma_1;\Gamma_2, \varphi \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2}{\Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2, \neg\varphi}$$

$$L(\wedge_l)\ \frac{\varphi, \psi, \Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2}{\varphi \wedge \psi, \Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2} \qquad R(\wedge_l)\ \frac{\Gamma_1;\Gamma_2, \varphi, \psi \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2}{\Gamma_1;\Gamma_2, \varphi \wedge \psi \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2}$$

$$L(\vee_r)\ \frac{\Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \varphi, \psi, \Delta_1;\Delta_2}{\Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \varphi \vee \psi, \Delta_1;\Delta_2} \qquad R(\vee_r)\ \frac{\Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow}, \varphi, \psi, \Delta_1;\Delta_2}{\Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \Delta_1;\Delta_2, \varphi \vee \psi}$$

$$L(\vee_l)\ \frac{\varphi, \Gamma_1;\Gamma_2 \overset{\chi_1}{\Rightarrow} \Delta_1;\Delta_2 \quad \psi, \Gamma_1;\Gamma_2 \overset{\chi_2}{\Rightarrow} \Delta_1;\Delta_2}{\varphi \vee \psi, \Gamma_1;\Gamma_2 \overset{\chi_1 \vee \chi_2}{\Rightarrow} \Delta_1;\Delta_2} \qquad R(\vee_l)\ \frac{\Gamma_1;\Gamma_2, \varphi \overset{\chi_1}{\Rightarrow} \Delta_1;\Delta_2 \quad \Gamma_1;\Gamma_2, \psi \overset{\chi_2}{\Rightarrow} \Delta_1;\Delta_2}{\Gamma_1;\Gamma_2, \varphi \vee \psi \overset{\chi_1 \wedge \chi_2}{\Rightarrow} \Delta_1;\Delta_2}$$

$$L(\wedge_r)\ \frac{\Gamma_1;\Gamma_2 \overset{\chi_1}{\Rightarrow} \varphi, \Delta_1;\Delta_2 \quad \Gamma_1;\Gamma_2 \overset{\chi_2}{\Rightarrow} \psi, \Delta_1;\Delta_2}{\Gamma_1;\Gamma_2 \overset{\chi_1 \vee \chi_2}{\Rightarrow} \varphi \wedge \psi, \Delta_1;\Delta_2} \qquad R(\wedge_r)\ \frac{\Gamma_1;\Gamma_2 \overset{\chi_1}{\Rightarrow} \Delta_1;\Delta_2, \varphi \quad \Gamma_1;\Gamma_2 \overset{\chi_2}{\Rightarrow} \Delta_1;\Delta_2, \psi}{\Gamma_1;\Gamma_2 \overset{\chi_1 \wedge \chi_2}{\Rightarrow} \Delta_1;\Delta_2, \varphi \wedge \psi}$$

$$L(R_\Box)\ \frac{\Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} \varphi;}{\Pi_1, \Box\Gamma_1;\Pi_2, \Box\Gamma_2 \overset{\neg\Box\neg\chi}{\Rightarrow} \Box\varphi, \Delta_1;\Delta_2} \qquad R(R_\Box)\ \frac{\Gamma_1;\Gamma_2 \overset{\chi}{\Rightarrow} ;\varphi}{\Pi_1, \Box\Gamma_1;\Pi_2, \Box\Gamma_2 \overset{\Box\chi}{\Rightarrow} \Delta_1;\Delta_2, \Box\varphi}$$

Figure 6 Rule interpolation for the rules of G3K

4.6 Main Theorem

▶ **Lemma 4.3.** *(Rule interpolation)*
Every rule R in G3K *interpolates, with interpolant function δ_R.*

Proof. Recall that we consider axioms to be zero-premise rules, so the statement in the lemma covers axioms as well. In Figures 5 and 6 the function δ_R is defined for each rule R in G3K, Section 4.5 explains how the figures have to be interpreted. We prove the lemma by showing that for any rule R, δ_R is an interpolant function. We treat rules $\mathrm{Ax_{id}}$, $\neg_l$, $\wedge_r$ and $R_\Box$, the other rules can be treated in a similar way.

$\mathrm{Ax_{id}}$: We consider two cases: $RL(\mathrm{Ax_{id}})$ and $RR(\mathrm{Ax_{id}})$, the other two are analogous. In the first case, $S^p = \Gamma_1;\Gamma_2, p \Rightarrow p, \Delta_1;\Delta_2$ and $\delta_{\mathrm{Ax_{id}}}(S^p) = \neg p$. And indeed, $\neg p$ is an inpolant of

S^p, as $\vdash \Gamma_1 \Rightarrow p, \neg p, \Delta_1$ and $\vdash \Gamma_2, p, \neg p \Rightarrow \Delta_2$. In the second case, $S^p = \Gamma_1; \Gamma_2, p \Rightarrow \Delta_1; \Delta_2, p$ and $\delta_{\mathrm{Ax_{id}}}(S^p) = \top$. Since $\vdash \Gamma_1 \Rightarrow \top, \Delta_1$ and $\vdash \Gamma_2, p \Rightarrow p, \Delta_2, \top$ is indeed an interpolant for S^p.

$\neg_l$: Consider instance

$$\neg_l \frac{\Gamma \Rightarrow \varphi, \Delta}{\Gamma, \neg\varphi \Rightarrow \Delta}$$

and a split $S^p = (\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2)$ of the conclusion. We consider the case that $\neg\varphi$ belongs to Γ_1, the case that it belongs to Γ_2 is analogous. Thus $\Gamma_1 = \Pi, \neg\varphi$ and the split of the premise is, by definition, $\Pi; \Gamma_2 \Rightarrow \varphi, \Delta_1; \Delta_2$. Suppose χ is an interpolant of the split premise, then $\delta_{L\neg}(\chi, S^p) = \chi$. By assumption, $\vdash \Pi \Rightarrow \chi, \varphi, \Delta_1$ and $\vdash \Gamma_2, \chi \Rightarrow \Delta_2$. Clearly, the former implies $\vdash \Pi, \neg\varphi \Rightarrow \chi, \Delta_1$, which means $\vdash \Gamma_1 \Rightarrow \chi, \Delta_1$. Thus $\delta_{L\neg}(\chi, S^p)$ is indeed an interpolant of S^p.

$\wedge_r$: Consider instance

$$\wedge_r \frac{\Gamma \Rightarrow \varphi, \Delta \qquad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \wedge \psi, \Delta}$$

and a split $S^p = (\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2)$ of the conclusion. First, we treat the case that the principal formula $\varphi \wedge \psi$ belongs to Δ_1, which means that $\Delta_1 = \varphi \wedge \psi, \Sigma$ for some multiset Σ. Thus the inherited splits of the left and right premise are $\Gamma_1; \Gamma_2 \Rightarrow \varphi, \Sigma; \Delta_2$ and $\Gamma_1; \Gamma_2 \Rightarrow \psi, \Sigma; \Delta_2$. Suppose that χ_1 and χ_2 are interpolants of the respective splits:

$$\vdash \Gamma_1 \Rightarrow \chi_1, \varphi, \Sigma \quad \vdash \Gamma_2, \chi_1 \Rightarrow \Delta_2 \quad \vdash \Gamma_1 \Rightarrow \chi_2, \psi, \Sigma \quad \vdash \Gamma_2, \chi_2 \Rightarrow \Delta_2.$$

Using the rules of the calculus and its closure under weakening (Lemma 4.2) this clearly implies that

$$\vdash \Gamma_1 \Rightarrow \chi_1 \vee \chi_2, \varphi \wedge \psi, \Sigma \quad \vdash \Gamma_2, \chi_1 \vee \chi_2 \Rightarrow \Delta_2,$$

proving that $\delta_{(\wedge_r)}(\chi_1, \chi_2, S^p) = \chi_1 \vee \chi_2$ is an interpolant of S^p.

Second, we treat the case that $\varphi \wedge \psi$ belongs to Δ_2, so that $\Delta_2 = \varphi \wedge \psi, \Sigma$ for some multiset Σ. Thus the inherited splits of the left and right premise are $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \varphi, \Sigma$ and $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \psi, \Sigma$. For any interpolants χ_1 and χ_2 of the respective splits we have

$$\vdash \Gamma_1 \Rightarrow \chi_1, \Delta_1 \quad \vdash \Gamma_2, \chi_1 \Rightarrow \varphi, \Sigma \quad \vdash \Gamma_1 \Rightarrow \chi_2, \Delta_1 \quad \vdash \Gamma_2, \chi_2 \Rightarrow \psi, \Sigma.$$

Therefore, again using the rules of the calculus and its closure under weakening,

$$\vdash \Gamma_1 \Rightarrow \chi_1 \wedge \chi_2, \Delta_1 \quad \vdash \Gamma_2, \chi_1 \wedge \chi_2 \Rightarrow \varphi \wedge \psi, \Sigma.$$

This shows that $\delta_{(\wedge_r)}(\chi_1, \chi_2, S^p) = \chi_1 \wedge \chi_2$ is an interpolant of S^p.

$R_\Box$: Consider instance

$$R_\Box \frac{\Gamma \Rightarrow \varphi}{\Pi, \Box\Gamma \Rightarrow \Box\varphi, \Delta}$$

Let $S^p = (\Pi_1, \Box\Gamma_1; \Pi_2, \Box\Gamma_2 \Rightarrow \Delta_1; \Delta_2)$ be a split of the conclusion. There are two cases, depending on whether $\Box\varphi$ belongs to Δ_1 or Δ_2. The second case is straightforward: $\Delta_2 = \Box\varphi, \Sigma$ for some Σ and the split premise is $\Gamma_1; \Gamma_2 \Rightarrow ; \varphi$. Thus for any interpolant χ of the split premise, $\Gamma_1 \Rightarrow \chi$ and $\Gamma_2, \chi \Rightarrow \varphi$ are derivable, and an application of $R_\Box$ to each of them shows that $\Box\chi$ is an interpolant of S^p.

We turn to the first case, where the split S^p and the inherited split of the premise of $R_\Box$ have respective forms

$$\Pi_1, \Box\Gamma_1; \Pi_2, \Box\Gamma_2 \Rightarrow \Box\varphi, \Sigma; \Delta_2 \qquad \Gamma_1; \Gamma_2 \Rightarrow \varphi; \ .$$

For any interpolant χ of the split premise we have $\vdash \Gamma_1 \Rightarrow \chi, \varphi$ and $\vdash \Gamma_2, \chi \Rightarrow$. Hence $\vdash \Gamma_1, \neg\chi \Rightarrow \varphi$ and $\vdash \Gamma_2 \Rightarrow \neg\chi$. An application of $R_\Box$ gives

$$\vdash \Pi_1, \Box\Gamma_1, \Box\neg\chi \Rightarrow \Box\varphi, \Sigma \qquad \vdash \Pi_2, \Box\Gamma_2 \Rightarrow \Box\neg\chi, \Delta_2.$$

This implies $\vdash \Pi_1, \Box\Gamma_1 \Rightarrow \neg\Box\neg\chi, \Box\varphi, \Sigma$ and $\vdash \Pi_2, \Box\Gamma_2, \neg\Box\neg\chi \Rightarrow \Delta_2$, as desired. ◀

▶ **Theorem 4.4.** *(Sequent interpolation)*
G3K *has sequent interpolation: If* $\vdash S$*, then any split of* S *has a split-interpolant.*

Proof. Suppose $\vdash S$ and consider an arbitrary split $S^p = (\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2)$ of S. We use induction to the depth d of the proof of S in G3K to show that S^p has a split-interpolant, using Lemma 4.3 which proves that for any rule R in G3K, δ_R is an interpolant function for R.

In case S is an instance of an axiom R, since δ_R is an interpolant function for R, $\delta(S^p)$ is an interpolant for S^p, and we are done. For $d > 1$ we consider the last rule of the proof, R, and the split of the premises according to Figure 6, to which we apply the induction hypothesis. This gives interpolant(s) χ (χ_1, χ_2) of the split premises. By the definition of interpolant functions, $\delta_R(\chi, S^p)$ $(\delta_R(\chi_1, \chi_2, S^p))$ is an interpolant of S^p. ◀

▶ **Theorem 4.5.** *(Interpolation for* K*)*
The modal logic K *has the Craig interpolation property.*

Proof. Suppose $\vdash_K \varphi \to \psi$. Then $\vdash_{\mathsf{G3K}} \varphi \Rightarrow \psi$. Consider split $(\varphi; \Rightarrow ; \psi)$. By Theorem 4.4, calculus G3K has sequent interpolation, thus there exists an interpolant χ in the common language of φ and ψ such that $\vdash_{\mathsf{G3K}} \varphi \Rightarrow \chi$ and $\vdash_{\mathsf{G3K}} \chi \Rightarrow \psi$. Thus χ is an interpolant for $\varphi \to \psi$ in K. ◀

4.7 Harvest from Sequent Interpolation

The constructive proof that G3K has sequent interpolation, Theorem 4.4, allows us to infer further strengthenings of the theorem. On the basis of the proof it is, for example, easy to observe that any interpolant of a sequent S does not contain a modal operator unless S does.

A less trivial and more important consequence of the constructive proof is that it shows that K has *Lyndon interpolation* by showing that G3K has *sequent Lyndon interpolation*. A logic has *Lyndon interpolation* if for any valid implication $\varphi \to \psi$ there exists an interpolant χ such that any propositional letter that occurs positively (negatively) in χ occurs positively (negatively) in both φ and ψ (the definition of positive and negative occurrences in formulas and sequents can be found in Section 4.1). A calculus has *sequent Lyndon interpolation* if for any split $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2$ of any derivable sequent S, there is a sequent interpolant χ such that any propositional letter that occurs positively (negatively) in χ occurs positively (negatively) in both $\Gamma_1 \Rightarrow \Delta_1$ and $\Gamma_2 \Rightarrow \Delta_2$. Inspection of the proofs of Lemma 4.3 and Theorem 4.4 shows that G3K has this form of interpolation, which clearly implies that K has Lyndon interpolation.

▶ **Corollary 4.6.** *(Lyndon interpolation)*
G3K *has sequent Lyndon interpolation and* K *has Lyndon interpolation.*

We conclude this section by listing some further advantages and disadvantages of the proof-theoretic method to prove interpolation:

- ✓ The proof-theoretic method can be used to establish Craig interpolation for many modal logics with a sequent calculus similar to calculus G3K for K.
- ✓ The proof-theoretic proof is modular: to establish that a certain extension of the calculus for K has sequent interpolation, it suffices to establish that the additional rules interpolate.
- ✓ The proof-theoretic method presented here provides a bound on the size of the interpolant of a sequent S in the size of the proofs of S in G3K.
- ✗ For many logics no sequent calculus is available that has all the necessary properties to apply the Maehara method.

5 Third Proof (Syntactic)

Recall from Chapter 1 how Craig interpolation for propositional logic can be proved syntactically via *disjunctive normal form* (DNF): to construct an interpolant for a propositional implication $\varphi \to \psi$, we first rewrite φ into disjunctive normal form (i.e., as a disjunction of consistent conjunctions of literals) and then we simply "drop" all (positive and negative) occurrences of proposition letters $p \in \mathrm{sig}(\psi) \setminus \mathrm{sig}(\varphi)$ from this DNF formula to obtain our Craig interpolant. Since converting a propositional formula to DNF incurs at most an exponential blowup, this yields an interpolant (in fact, a uniform interpolant) of exponential size. In this section, we will describe an interpolation method for modal logic that can be viewed as a modal analogue of this technique. It was used in [7] and can be summarized in one sentence as follows: *to obtain an interpolant for $\varphi \to \psi$, we bring φ in a suitable normal form, and then drop all occurrences of proposition letters $p \in \mathrm{sig}(\varphi) \setminus \mathrm{sig}(\psi)$ from φ.*

The precise normal form that we need is known as ∇-*normal form* [1, 23]. In order to define it, we must first introduce the ∇ operator. It takes a finite set of formulas as argument, and its semantics is as follows: $M, w \models \nabla\Phi$ holds if *each* $\varphi \in \Phi$ *is true in some successor and each successor satisfies some* $\varphi \in \Phi$. In other words, $\nabla\Phi$ is equivalent to $\bigwedge_{\varphi\in\Phi} \Diamond\varphi \wedge \Box \bigvee_{\varphi\in\Phi} \varphi$. Note that $\nabla\emptyset$ is equivalent to $\Box\bot$.

A modal formula is in ∇-normal form if it generated by the following grammar:

$$\varphi \quad ::= \quad \top \mid \bot \mid \pi \mid \nabla\Phi \mid \pi \wedge \nabla\Phi \mid (\varphi \vee \varphi)$$

where π is a consistent conjunction of literals (in other words, π is of the form $(\neg)p_1 \wedge \cdots \wedge (\neg)p_n$, with $n > 0$, where $p_1, \ldots, p_n$ are distinct proposition letters), and Φ is a finite set of formulas in ∇-form.

▶ **Example 5.1.** The modal formula $\Diamond p \wedge \Box(p \vee q)$ can be written in ∇-normal form as $\nabla\{p\} \vee \nabla\{p, q\}$. Similarly, the modal formula $\Diamond p \wedge \Box(q \vee r)$ can be written in ∇-normal form as $\nabla\{p \wedge q, q\} \vee \nabla\{p \wedge q, q, r\} \vee \nabla\{p \wedge r, r\} \vee \nabla\{p \wedge r, q, r\}$. Finally, the modal formula $\Box(p_1 \vee p_2) \wedge \Box(q_1 \vee q_2)$ can be written in ∇-normal form as

$$\bigvee_{\Phi\subseteq\{p_i\wedge q_j \mid i,j\in\{1,2\}\}} \nabla\Phi.$$

⌟

▶ **Theorem 5.2.** *Every modal formula* φ *is equivalent to a formula in* ∇*-normal form, which can be computed from* φ *in exponential time.*

Proof. We will use a normalization procedure that was given in [7]. Let φ be any modal formula. We will assume that φ is in negation normal from. For convenience in what follows we will treat conjunction as an operator that can take any number of arguments.

$$\begin{array}{lcll} \mathrm{nf}(\varphi) & = & \varphi & \text{for } \varphi \text{ of the form } (\neg)p, \top, \bot \\ \mathrm{nf}(\Diamond\varphi) & = & \nabla\{\mathrm{nf}(\varphi), \top\} & \\ \mathrm{nf}(\Box\varphi) & = & \nabla\{\mathrm{nf}(\varphi)\} \vee \nabla\emptyset & \\ \mathrm{nf}(\varphi \vee \psi) & = & \mathrm{nf}(\varphi) \vee \mathrm{nf}(\psi) & \end{array}$$

The only difficult case is for conjunction, i.e., if φ is of the form $\bigwedge\Phi$. If Φ contains a formula of the form $\top, \bot$ or a disjunction or conjunction, then we can apply the following simplification rules:

$$\begin{array}{lcl} \mathrm{nf}(\bigwedge(\Phi \cup \{\top\})) & = & \mathrm{nf}(\bigwedge\Phi) \\ \mathrm{nf}(\bigwedge(\Phi \cup \{\bot\})) & = & \bot \\ \mathrm{nf}(\bigwedge(\Phi \cup \{\varphi \vee \psi\})) & = & \mathrm{nf}(\bigwedge(\Phi \cup \{\varphi\})) \vee \mathrm{nf}(\bigwedge(\Phi \cup \{\psi\})) \\ \mathrm{nf}(\bigwedge(\Phi \cup \{\bigwedge\Psi\})) & = & \mathrm{nf}(\bigwedge(\Phi \cup \Psi)) \end{array}$$

This leaves us with the case where Φ consists of literals and formulas whose main connective is a $\Diamond$ or $\Box$. Let Φ_{lit} be the set of literals in Φ, let $\Phi_\Diamond$ be the set of formulas ψ for which $\Diamond\psi \in \Phi$ and let $\Phi_\Box$ be the set of formulas ψ for which $\Box\psi \in \Phi$. We distinguish three cases:

1. If Φ_{lit} contains a proposition letter and its negation, then clearly $\bigwedge \Phi$ is inconsistent, and we let $\mathrm{nf}(\bigwedge \Phi) = \bot$.
2. Otherwise, if $\Phi_\Diamond = \emptyset$ we define $\mathrm{nf}(\bigwedge \Phi)$ as $\bigwedge \Phi_{lit} \wedge (\nabla\{\mathrm{nf}(\bigwedge \Phi_\Box)\} \vee \nabla\emptyset)$
3. Otherwise, we define $\mathrm{nf}(\bigwedge \Phi)$ as $\bigwedge \Phi_{lit} \wedge \nabla(\{\mathrm{nf}(\psi \wedge \bigwedge \Phi_\Box) \mid \psi \in \Phi_\Diamond\} \cup \{\bigwedge \Phi_\Box\})$

It can be verified that $\mathrm{nf}(\varphi)$ is in ∇-normal form, $\mathrm{nf}(\varphi)$ is equivalent to φ, $|\mathrm{nf}(\varphi)| = 2^{O(|\varphi|^2)}$, and this size bound on $\mathrm{nf}(\varphi)$ also bounds the time needed to compute $\mathrm{nf}(\varphi)$. ◀

Intuitively, what makes the ∇-normal form special is that formulas can only contain a limited, "interaction-free" form of conjunction. This is reflected in the remarkable fact (which will not be used in our proof of Craig interpolation) that the satisfiability of a modal formula in ∇-normal form can be decided in linear time by a simple recursive procedure.

As we will see in Section 6 (cf. Remark 6.12), modal formulas in ∇-normal form correspond in an intuitive sense to non-deterministic tree automata.

Next, we define a "removal" operator $\tilde{\exists}$ that takes a modal formula in ∇-normal form and removes all occurrences of a specified set of proposition letters. Formally, given a formula φ in ∇-normal form and a set $\mathcal{P}$ of proposition letters, we define $\tilde{\exists}\mathcal{P}.\varphi$ as follows:

$$\begin{array}{lcl}
\tilde{\exists}\mathcal{P}.\top & = & \top \\
\tilde{\exists}\mathcal{P}.\bot & = & \bot \\
\tilde{\exists}\mathcal{P}.(\pi) & = & \pi' \\
\tilde{\exists}\mathcal{P}.(\pi \wedge \nabla\Psi) & = & \pi' \wedge \nabla\{\tilde{\exists}\mathcal{P}.\psi \mid \psi \in \Psi\} \\
\tilde{\exists}\mathcal{P}(\varphi \vee \psi) & = & \tilde{\exists}\mathcal{P}(\varphi) \vee \tilde{\exists}\mathcal{P}(\psi)
\end{array}$$

where π' is obtained from π by dropping all (positive and negative) occurrences of proposition letters in $\mathcal{P}$.

▶ **Proposition 5.3.** *For all modal formulas φ in ∇-normal form over propositional signature σ, sets of proposition letters $\mathcal{P}$, and pointed Kripke models (M, w), the following are equivalent:*

1. $M, w \models \tilde{\exists}\mathcal{P}.\varphi$,
2. *There is a pointed Kripke model $N, v \models \varphi$ such that $(M, w) \underline{\leftrightarrow}_{\sigma\setminus\mathcal{P}} (N, v)$.*

Proof. By induction on φ. The only interesting cases in the induction argument are for formulas φ of the form π or $\nabla\Psi$ or $\pi \wedge \nabla\Psi$. Of these three cases, we will discuss the latter (the former being simpler). Therefore, let $\varphi = \pi \wedge \nabla\Psi$. Recall that by definition, $\tilde{\exists}\mathcal{P}.\varphi = \pi' \wedge \nabla\{\tilde{\exists}\mathcal{P}.\psi \mid \psi \in \Psi\}$.

From 1 to 2: let $M, w \models \tilde{\exists}\mathcal{P}.\varphi$. Then, for each $\psi_i \in \Psi$, there is a successor w_i of w such that $M, w_i \models \tilde{\exists}\mathcal{P}.\psi_i$. By induction, there is a $N_i, v_i \models \psi_i$ such that $(M, w_i) \underline{\leftrightarrow}_{\sigma\setminus\mathcal{P}} (N_i, v_i)$. Similarly, for each successor u_j of w there is some $\psi_j \in \Psi$ such that $M, u_j \models \tilde{\exists}\mathcal{P}.\psi_j$, and, by induction hypothesis, there is some $N'_j, v'_j \models \psi_j$ such that $(M, u_j) \underline{\leftrightarrow}_{\sigma\setminus\mathcal{P}} (N'_j, v'_j)$. Let N, v be obtained by taking the disjoint union of all the the aforementioned pointed Kripke models N_i, v_i as well as N'_j, v'_j, adding a node v with an edge from v to each v_i and to each v'_j, and making π true at v. It is easy to see that $N, v \models \pi \wedge \nabla\Psi$ and that $(N, v) \underline{\leftrightarrow}_{\sigma\setminus\mathcal{P}} (M, w)$.

From 2 to 1: assume that $(M, w) \underline{\leftrightarrow}_{\sigma \setminus \mathcal{P}} (N, v)$ and $N, v \models \varphi$. In particular, each successor of v in N satisfies some $\psi \in \Psi$, and each $\psi \in \Psi$ is satisfied in N by a successor of v. It follows by the induction hypothesis (with the identity bisimulation) that each successor of v in N satisfies $\tilde{\exists}\mathcal{P}.\psi$ for some $\psi \in \Psi$, and for each $\psi \in \Psi$ there is a successor of v in N that satisfies $\tilde{\exists}\mathcal{P}.\psi$. It follows that $N, v \models \tilde{\exists}\mathcal{P}.\varphi$. Since $\tilde{\exists}\mathcal{P}.\varphi$ does not contain any occurrences of the proposition letters in $\mathcal{P}$, it is invariant for bisimulations over the signature without the proposition letters in $\mathcal{P}$. Therefore also $M, w \models \tilde{\exists}\mathcal{P}.\varphi$. ◀

Finally, we get our proof of interpolation:

Proof of Craig interpolation for modal logic. Let $\varphi \to \psi$ be a valid modal implication. We first rewrite φ to ∇-normal form. Then, we take $\vartheta = \tilde{\exists}\mathcal{P}.\varphi$ with $\mathcal{P} = \mathrm{sig}(\varphi) \setminus \mathrm{sig}(\psi)$ We claim that ϑ is a Craig interpolant. It is clear from the construction that $\mathrm{sig}(\vartheta) \subseteq \mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi)$. Therefore, it remains to show that $\varphi \to \vartheta$ and $\vartheta \to \psi$ are valid. Let $M, w \models \varphi$. Then, by Proposition 5.3 (choosing (N, v) to be (M, w) and using the identity bisimulation) it follows that $M, w \models \vartheta$. Next, let $M, w \models \vartheta$. By Proposition 5.3, there is a pointed Kripke model $N, v \models \varphi$ such that M, w and N, v are bisimilar w.r.t. all proposition letters except possibly those in $\mathcal{P}$. Therefore $N, v \models \psi$. It follows that $M, w \models \psi$, since ψ does not contain any occurrences of proposition letters in $\mathcal{P}$. ◀

Observe that this proof, in fact, yields something stronger than Craig interpolation, namely uniform interpolation. Furthermore, Proposition 5.3 shows that the "$\tilde{\exists}\mathcal{P}$" operation can be interpreted as expressing a form of *second-order quantification modulo bisimulations*. Indeed, this is known as *bisimulation quantifiers*. Bisimulation quantifiers were originally introduced in [37] in the context of intuitionistic logic, and then used in [20] and [46] as a tool to prove uniform interpolation for modal logic. It was subsequently used to prove uniform interpolation for the modal μ-calculus [15] and to prove a restricted form of uniform interpolation for the guarded fragment [12]. It remains unclear to what extent the method can be adapted to other logics. For instance:

▶ **Open Question 1.** *Can the above ∇-normal form based method be adapted and used to prove uniform interpolation for the basic tense logic K_t?*

In summary:

- ✓ This proof, in fact, yields something stronger than Craig interpolation, namely uniform interpolation.
- ✓ Since normalization to ∇-normal form can be performed in exponential time, it yields uniform interpolants of single exponential size.
- ✓ Lyndon interpolants can be obtained similarly by dropping only the positive (negative) occurrences of proposition letters that do not occur positively (negatively) in ψ). Note that the conversion to ∇-normal form preserves polarities of proposition letters.

✓ The same construction also works for the modal μ-calculus (although the proof of the ∇-normal form theorem is more involved here, cf. [44]).

✗ On the other hand, since this technique yields uniform interpolation, it does not allow us to prove Craig interpolation for logics that lack uniform interpolation (eg., S4, or the extension of modal logic with the global modality).

6 Fourth Proof (Automata-Theoretic)

A well-known consequence of bisimulation-invariance is the fact that modal logic has the *tree model property*: when studying properties of modal formulas, one can safely restrict attention to tree models. [6] This opens the possibility of using *tree automata*. Indeed, many results in modal logic have been obtained with the help of tree automata. Craig interpolation, in particular, can be proved via tree automata. The high-level outline of this approach is as follows: we first define a class of tree automata that have the same expressive power as modal logic, allowing us to translate back and forth between modal formulas and automata. Given a valid modal implication $\models \varphi \to \psi$, we then translate the formulas in question to tree automata, we construct a Craig interpolant as a tree automaton, by making use of the automata-theoretic operation of *projection*, and finally we translate the automaton back to a modal formula. As we will see (cf. Remark 6.12), this techniques turns out to be closely related to the syntactic approach from Section 5.

We present this approach in detail, here, for the basic modal logic K. As it turns out, for this case, we only need a rather simple type of tree automaton, which we will call a *modal automaton*. The input of such an automaton will be a tree-shaped Kripke model. Formally, let τ be any finite propositional signature. A *τ-tree* is a (possibly infinite) pointed Kripke model (M, w) such that, for every world v there exists exactly one directed path from w to v. Such a τ-tree is said to be *fat* if every non-root world has infinitely many bisimilar siblings, i.e., for each $(v, v') \in R$, there are infinitely many v'' such that $(v, v'') \in R$ and $(M, v') \underline{\leftrightarrow}_\tau (M, v'')$.

▶ **Example 6.1.** Consider the three Kripke models over propositional signature $\tau = \{p\}$ depicted in Figure 7. The pointed Kripke model (M_1, w_0) is *not* a τ-tree, whereas (M_2, v_0) is a non-fat τ-tree, and (M_3, u) is a fat τ-tree. All three pointed Kripke models are τ-bisimilar.

The following proposition explains why we may restrict attention to tree-shaped Kripke models.

▶ **Proposition 6.2.** *Let (M, w) be a pointed Kripke model over propositional signature τ. Then there is a fat τ-tree, which we will denote by $(unr(M), \langle w \rangle)$ such that $(M, w) \underline{\leftrightarrow}_\tau (unr(M), \langle w \rangle)$.*

[6] In the case of the modal logic K, one may even restrict attention to finite trees, although this is not relevant here.

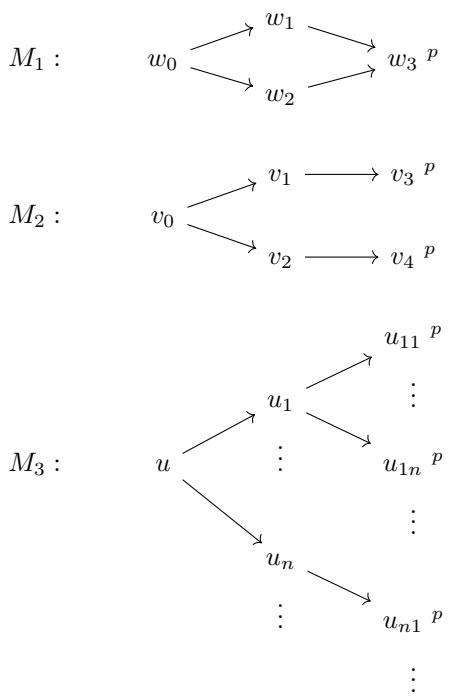


Figure 7 Three examples of Kripke models

Proof. The proof uses a variant of the standard tree unraveling construction, cf. [5]: let $M = (W, R, V)$. By a *path from* w we will mean a finite sequence $\langle w_0, k_1, w_1, k_2, w_2 \ldots, k_n, w_n\rangle$ $(n \geq 0)$ where $w_0 = w$, where $\langle w_i, w_{i+1}\rangle \in R$ for all $i < n$, and where each k_i is a natural number. By a *successor* of a path $\langle w_0, \ldots, w_n\rangle$ we will mean a path that extends it with one more number and world, or, in other words, a path of the form $\langle w_0, \ldots, w_n, k_{n+1}, w_{n+1}\rangle$. Let $unr(M) = (W', R', V')$ where W' is the set of all paths from w, R' is the successor relation on such paths, and $V'(p)$ is the set of all paths whose final world element belongs to $V(p)$. It can be verified that $(unr(M), \langle w\rangle)$ is a fat τ-tree and that $(M, w) \underline{\leftrightarrow}_\tau (unr(M), \langle w\rangle)$. Indeed, the relation Z that consists of all pairs (x, v) with x a path and v its final world element is a τ-bisimulation. ◀

For a propositional signature τ, we will denote by 2^τ the set of all truth assignments for the proposition letters in τ. To simplify notation, we will equate each truth assignments for τ with the subset of τ consisting of the proposition letters that are assigned to true. For a world w in a given Kripke model, $M = (W, R, V)$, we will denote by $lab_\tau(w) = \{p \in \tau \mid w \in V(p)\}$ the truth assignment that makes true precisely those $p \in \tau$ for which it holds that $w \in V(p)$.

This will allow us to view a τ-tree as a node-labeled tree (where 2^τ is the set of node labels).

▶ **Definition 6.3** (Modal automaton). *Let σ be any finite propositional signature. A modal automaton*[7] *for σ is a tuple $A = (\Sigma, Q, \delta, q_0, F)$ consisting of*

- *An* alphabet $\Sigma = 2^\sigma$,
- *A finite set of* states Q,
- *A* transition relation $\delta \subseteq Q \times \Sigma \times \wp(Q)$,
- *An* initial state $q_0 \in Q$, *and*
- *A set of* accepting states $F \subseteq Q$.

▶ Remark 6.4. As customary in automata theory, we will use the letter Q to denote the set of states of the automaton and we will use the letter q to denote an element of Q, that is, a state of the automaton. Therefore, in the remainder of this section, q will always denote a state (unlike elsewhere in this chapter, where we have been using q occasionally as a proposition letter).

In the same way that a modal formula evaluates to true or false in a given pointed Kripke model (M, w), a modal automaton can either *accept* or *reject* a given τ-tree.

▶ **Definition 6.5** (Acceptance). *An* accepting run *of a modal automaton $A = (\Sigma, Q, \delta, q_0, F)$ for τ on τ-tree (M, w) with $M = (W, R, V)$ is a function $\rho : W \to Q$ such that the following conditions are met:*

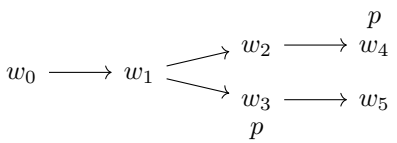
1. *$\rho(w) = q_0$,*
2. *whenever $\rho(v) = q$, either $q \in F$ or $(q, lab_\tau(v), \{\rho(v') \mid (v, v') \in R\}) \in \delta$.*

We say that (M, w) is accepted by A if there is an accepting run of A on (M, w).

Note that a run of a modal automaton may accepting even if an accepting state does not appear on every branch. Indeed, the final states can be dispensed with entirely, by introduce a special state q^* together with all transitions of the form $(q_f, \ell, \{q^*\}$ for $q_f \in F$ and $a \in \Sigma$, as well as all transitions of the form $(q^*, a, \{q^*\}$ and $(q^*, a, \emptyset\}$. However, having an explicit set of accepting states in the specification of automata will help us define "acyclicity" below.

▶ **Example 6.6.** Consider the propositional signature $\tau = \{p\}$. Consider the automaton A with two states, q_0, q_1, where q_0 is the initial state and q_1 is an accepting state, that has transitions $(q_0, \emptyset, \{q_0\})$, $(q_0, \{p\}, \emptyset)$ and $(q_0, \{p\}, \{q_1\})$.

Let (M, w_1) be the τ-tree depicted as follows:



[7] The name "modal automaton" we have chosen here, may be considered slightly misleading: as we will see in a moment (cf. Example 6.6), these automata are strictly more expressive than modal formulas. We will be adding a further restriction to obtain an exact equivalence.

Then A accepts (M, w_0). Indeed, the function ρ given by

$$\rho(v) = \begin{cases} q_0 & \text{for } v \in \{w_0, w_1, w_2, w_3, w_4\} \\ q_1 & \text{for } v \in \{w_5\} \end{cases}$$

is an accepting run. More generally, A accepts precisely the τ-trees in which *every path ending in a leaf contains a node that satisfies p*. There is no modal formula that defines the same property, although it can be expressed in extensions of modal logic such as CTL^* (by the formula $A(Fp \vee GF\top)$) and the modal μ-calculus (by the formula $\nu x(p \vee \Box x)$). ⌟

As the above example shows, there is not an exact correspondence between modal automata and modal formulas, since modal automata are too expressive. To solve this, we will restrict attention to acyclic automata. We say that a modal automaton is *acyclic* if there is a function $rank : Q \to \mathbb{N}$ such that whenever $(q, a, S) \in \delta$ and $q' \in S$, then $rank(q') < rank(q)$. If a modal automaton is acyclic, intuitively, this means that it does not allow for an infinite sequence of transitions. Note that the automaton in the above example is *not* acyclic.

▶ **Theorem 6.7.** *Fix any finite propositional signature σ.*

1. *For every modal formula χ, there is an acyclic modal automaton A_χ such that A_χ accepts precisely the σ-trees that satisfy χ.*
2. *Conversely, for every acyclic modal automaton A there is a modal formula χ_A such that a fat σ-tree satisfies χ_A iff it is accepted by A.*

Proof. 1. We may assume without loss of generality that χ is in negation normal form. Let X be the set of all subformulas of χ. For $\Psi, \Psi' \subseteq X$, we say that Ψ' is a *decisive refinement* of Ψ if

1. $\Psi \subseteq \Psi'$,
2. whenever $\chi_1 \wedge \chi_2 \in \Psi'$, then $\chi_1 \in \Psi'$ and $\chi_2 \in \Psi'$,
3. whenever $\chi_1 \vee \chi_2 \in \Psi'$, then $\chi_1 \in \Psi'$ or $\chi_2 \in \Psi'$, and
4. $\bot \notin \Psi'$,

We define the *rank* of $\Psi \subseteq X$ to be 0 if $\Psi = \emptyset$, and $1 + n$ otherwise, where n is the maximal modal depth of a formula in Ψ.

We now construct our automaton A_χ as follows:

- $\Sigma = 2^\sigma$
- $Q = \{q_\Psi \mid \Psi \subseteq X\}$.
- $(q_\Psi, a, S) \in \delta$ iff there is a decisive refinement Ψ' of Ψ such that
  - (i) all the literals in Ψ' are satisfied under the propositional valuation a;
  - (ii) for each formula in Ψ' of the form $\Diamond\psi$, there is some $q_{\Psi''} \in S$ with $\psi \in \Psi''$;
  - (iii) for each formula in Ψ' of the form $\Box\psi$ and for every $q_{\Psi''} \in S$, $\psi \in \Psi''$; and
  - (iv) for each $q_{\Psi''} \in S$, the rank of Ψ'' is strictly less than the rank of Ψ.
- The initial state q_0 is $q_{\{\chi\}}$,
- $F = \{q_\emptyset\}$.

Note that this automaton is, by construction, acyclic. Furthermore, it can be shown that A_χ accepts precisely those σ-trees that satisfy χ. More precisely, let us say that A_χ accepts a σ-tree (M, w) *from a state* q_Ψ if it accepts (M, w) when using q_Ψ (instead of $q_{\{\chi\}}$ as the initial state. Then, it can be shown that A_χ *accepts a* σ*-tree* (M, w) *from a state* q_Ψ *if and only if* $M, w \models \psi$ *for all* $\psi \in \Psi$. The proof, which is by induction on the rank of Ψ, is left as an exercise to the reader.

2. Let $A = (\Sigma, Q, \delta, q_0, F)$ be an acyclic modal automaton, with $\Sigma = 2^\sigma$. Let $rank : Q \to \mathbb{N}$ be a witness for the fact that A is acyclic. We will construct a modal formula χ_q for each $q \in Q$, by induction on the rank of q. If $q \in F$, we can pick χ_q to be $\top$. Otherwise, we can define χ_q as follows (by induction on rank):

$$\chi_q := \bigvee_{(q,a,S)\in\delta} \Big(\bigwedge_{p\in a} p \wedge \bigwedge_{p\in\sigma\setminus a} \neg p \wedge \nabla\{\chi_{q'} \mid q' \in S\}\Big)$$

Here, we conveniently make use of the ∇ operator that we defined in Section 5 (which can also be written out in terms of $\Diamond$ and $\Box$).

Claim: A fat σ-tree (M, w) satisfies χ_q if and only if it is accepted by A from state q.

The claim can be proved by induction on the rank of the state q. For the induction step, the "if" direction is entirely straightforward, while the "only if" direction relies on the following fact about fat σ-trees: whenever $\nabla\{\varphi_1, \ldots, \varphi_n\}$ is satisfied by at a world w in a fat σ-tree, then there are *distinct* successors $v_1, \ldots, v_n$ of w such that each v_i satisfies φ_i.

It follows from the above claim that $\chi_A = \chi_{q_0}$ is satisfied by those fat σ-trees that are accepted by A. ◀

One of the fundamental insights underlying much of automata theory, is that the languages accepted by finite-state automata are closed under natural operations such as *intersection*, *complementation*, and *projection*. Of these operations, *projection* turns out to be closely related to (uniform) Craig interpolation. For modal automata, the projection operation is most naturally defined as follows:

▶ **Definition 6.8** (Projection). *Let* $A = (2^\sigma, Q, \delta, q_0, F)$ *be a modal automaton over signature* σ*, and let* $\tau \subseteq \sigma$*. The* projection of A to τ*, denoted* $Proj_\tau(A)$*, is the modal automaton* $A = (2^\tau, Q, \delta', q_0, F)$ *where* $\delta = \{(q, a \cap \tau, S) \mid (q, a, S) \in \delta\}$.

For a Kripke model $M = (W, R, V)$ over a propositional signature σ and for $\tau \subseteq \sigma$, by the τ*-reduct* of M we will mean the Kripke model over signature τ obtained by restricting V to the proposition letters in τ.

▶ **Proposition 6.9.** *Let* A *be a modal automaton over signature* σ *and let* $\tau \subseteq \sigma$*. Then* $Proj_\tau(A)$ *accepts precisely those* τ*-trees that are the* τ*-reduct of a* σ*-tree accepted by* A.

Proof. Suppose A accepts (M, w), and let ρ be a witnesssing accepting run of A for (M, w). It follows immediately from the construction of $Proj_\tau(A)$ that the same function ρ is also an accepting run of $Proj_\tau(A)$ for the τ-reduct of (M, w).

Conversely, suppose that $Proj_\tau(A)$ accepts a σ-tree (M, w), where $M = (W, R, V)$. Let ρ be a witnessing accepting run. By definition, this means that, for each $v \in W$ with $\rho(v) \notin F$, the triple of the form $(\rho(v), lab_\tau(w), \{\rho(u) \mid (v, u) \in R\})$ belongs to the transition relation of $Proj_\tau(A)$. The transition relation of A must therefore contain a triple $(\rho(v), a_v, \{\rho(u) \mid (v, u) \in E\})$ such that $a_v \cap \tau = lab_\tau(v)$. Now, choose any such triple and let $M' = (W, R, V')$ where $V' : \sigma \to \wp(W)$ is given by $v \in V(p)$ iff $p \in a_v$. Note that this is only well defined for $\rho(v) \notin F$. For those $v \in W$ with $q(v) \in F$, we can simply let V' agree with V on all proposition letters. By construction, M is the τ-reduct of M', and ρ is an accepting run of A for (M', w). ◀

Proposition 6.9 can be viewed as an automata-theoretic analogue of Proposition 5.3. Note that the projection operation preserves the acyclicity of a modal automaton.

Proof of Craig interpolation for modal logic. Let $\models \varphi \to \psi$ be a valid modal implication. We construct ϑ by taking the modal automaton of φ, projecting it to $\tau = \mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi)$, and translating it back to a modal formula. That is, $\vartheta = \chi_{Proj_\tau(A_\varphi)}$. We claim that ϑ is a Craig interpolant. It is clear from the construction that $\mathrm{sig}(\vartheta) \subseteq \mathrm{sig}(\varphi) \cap \mathrm{sig}(\psi)$. Therefore, it remains to show that $\varphi \to \vartheta$ and $\vartheta \to \psi$ are valid.

Let M, w be any pointed Kripke model over $\mathrm{sig}(\varphi) \cup \mathrm{sig}(\psi)$ such that $M, w \models \varphi$. We may assume without loss of generality that M, w is a fat $\mathrm{sig}(\varphi) \cup \mathrm{sig}(\psi)$-tree. By Proposition 6.9, the τ-reduct of M, w satisfies ϑ and hence $M, w \models \vartheta$.

Next, let M, w be any pointed Kripke model over $\mathrm{sig}(\varphi) \cup \mathrm{sig}(\psi)$ such that $M, w \models \vartheta$. Again, we may assume without loss of generality that M, w is a fat $\mathrm{sig}(\varphi) \cup \mathrm{sig}(\psi)$-tree. Since $\mathrm{sig}(\vartheta) \subseteq \tau$, the τ-reduct of M, w already satisfies ϑ. It follows that the τ-reduct of M, w is accepted by $Proj_{\tau(A_\varphi)}$. By Proposition 6.9 and Theorem 6.7, the τ-reduct of M, w is equal to the τ-reduct of some $\mathrm{sig}(\varphi) \cup \mathrm{sig}(\psi)$-tree $N, v \models \varphi$. In other words, M, w and N, v are τ-isomorphic. Let $f : M \cong N$ be a τ-isomorphism. Next, we extend f to an $\mathrm{sig}(\psi)$-isomorphism by changing N. More precisely, let $M = (W^M, R^M, V^M)$ and $N = (W^N, R^N, V^N)$ and let let N' be a copy of N in which the valuation of each proposition letter $p \in \mathrm{sig}(\psi) \setminus \tau$ is replaced by $\{f(u) \mid u \in V^M(p)\}$. By construction, f is a $\mathrm{sig}(\psi)$-isomorphism between (M, w) and (N', v), and N' differs from N only on the interpretation of proposition letters that do not belong to τ. Since $N, v \models \varphi$ and N' differs from N only with respect to proposition letters that do not appear in φ, we have that $N', v \models \varphi$ and hence $N', v \models \psi$. This allows us to conclude, finally, that $M, w \models \psi$. ◀

▶ Remark 6.10. A close analysis of the proof of Theorem 6.7 reveals that A_χ can be constructed from χ in exponential time. Moreover a DAG representation of χ_A can be computed from A in polynomial time. Since the projection operation can be performed in polynomial time (in

fact, in linear time) this gives us a method for constructing interpolants (indeed, uniform interpolants in ∇-normal form) of singly exponential size if we allow DAG representations.

▶ Remark 6.11. Many different types of tree automata are in use, in the context of modal logic and elsewhere. These include *deterministic*, *non-deterministic* and *alternating* automata, that take as input either *ranked trees* (where each non-leaf node has a fixed number of children) or *unranked-trees*. Moreover, the trees in question can be either *finite* or *infinite*, and in the case of infinite trees, a variety of different *acceptance conditions* can be used to determine if a given run is accepting. The modal automata we used here are non-deterministic automata that run on (finite or infinite) unranked trees and that have a very simple type of acceptance condition. For more expressive extensions of the modal language, such as the modal μ-calculus, one typically needs more complex acceptance conditions, which speak about the states of the automaton that are allowed to appear infinitely often on the different branches of an infinite run. See [10] for a comprehensive introduction to different types of tree automata. The use of automata-theoretic techniques in the study of modal logics goes back to [43].

▶ Remark 6.12. The automata-theoretic proof of Craig interpolation we presented here is, at a technical level, very close to the proof of interpolation via ∇-normal form (Section 5). Indeed, the automata-theoretic proof produces a (uniform) interpolant that is in ∇-normal form. The reader might wonder what the advantage is of going through automata when there is a direct syntactic construction that already yields a uniform interpolant of single exponential size. The reason we have included the automata theoretic proof is that it helps us understand better what is happening at a conceptual level. Indeed, in some sense, a modal formula in ∇-normal form *is* just an acyclic modal automaton (and, moreover, this correspondence comes with an efficient translation if we allow a DAG-representation for the formula).

In light of Remark 6.12, we omit a list of pros and cons, as they correspond to the ones that were already given in Section 5.

7 Fifth Proof (via Quasi-Models and Type Elimination Sequences)

The interpolation method we will now describe is one that was successfully applied to the guarded-fragment and the guarded-negation fragment in [3] to compute interpolants of worst-case optimal size. Furthermore, in [26] a refinement of this approach was used to obtain decision procedures for the existence of interpolants in some extensions of modal logic that lack Craig interpolation. In both cases the constructions used are quite involved. Here, we illustrate the method in the simpler setting of the basic modal language.

We first review quasi-models (a.k.a. type models) and type elimination sequences as a technique for deciding validity (or, equivalently, satisfiability). This technique can be traced back to [39] where it was used to show that the satisfiability problem for Propositional Dynamic Logic (PDL) is in ExpTime. After this review, we will show how to extract Craig interpolants (and, in fact, Lyndon interpolants) from type elimination sequences.

A quasi-model is, essentially, a consistent collection of "types", where a type is a subset $X \subseteq Y$. Here, Y is some finite set of relevant formulas. For instance, Y may be the set of all subformulas of a given formula whose validity we are trying to determine.

Our presentation is specifically tailored to the case where the input formula (for which we want to test validity) is an implication between a pair of modal formulas. This will facilitate the interpolant-construction later. In addition, some of our definitions (e.g., the definition of SUBF below) are tailored in anticipation of the specific use case of constructing Lyndon interpolants.

Recall the *negation normal form (NNF)* for modal formulas (cf. Section 2).

▶ **Definition 7.1** (SUBF, and LITERALS)**.**

- *If χ is a formula in NNF, we denote by* $\text{SUBF}(\chi)$ *the following set:*

$$\begin{array}{lcl} \text{SUBF}(\alpha) & = & \{\alpha\} \text{ for all } \alpha \text{ of the form } p, \neg p, \top \text{ or } \bot \\ \text{SUBF}(\chi_1 \wedge \chi_2) & = & \text{SUBF}(\chi_1) \cup \text{SUBF}(\chi_2) \cup \{\chi_1 \wedge \chi_2\} \\ \text{SUBF}(\chi_1 \vee \chi_2) & = & \text{SUBF}(\chi_1) \cup \text{SUBF}(\chi_2) \cup \{\chi_1 \vee \chi_2\} \\ \text{SUBF}(\Diamond\chi) & = & \text{SUBF}(\chi) \cup \{\Diamond\chi\} \\ \text{SUBF}(\Box\chi) & = & \text{SUBF}(\chi) \cup \{\Box\chi\} \end{array}$$

  Note that $\text{SUBF}(\neg p)$ *does* not *include p.*
- *If χ is a formula in NNF, we denote by* $\text{LITERALS}(\chi)$ *the set of all formulas in* $\text{SUBF}(\chi)$ *of the form p or $\neg p$. For example,* $\text{LITERALS}(\Box p \wedge \Diamond\neg q) = \{p, \neg q\}$.

Fix a modal implication $\varphi \to \psi$. The next few definitions are all relative to the given choice of the modal formulas φ and ψ.

▶ **Definition 7.2** (Types; Overlap-Consistency)**.**

1. *By a* locally-consistent *subset of* $\text{SUBF}(\chi)$ *we mean a subset $X \subseteq \text{SUBF}(\chi)$ such that (i) whenever $\chi_1 \wedge \chi_2 \in X$, then $\chi_1, \chi_2 \in X$; (ii) whenever $\chi_1 \vee \chi_2 \in X$, then at least one of χ_1, χ_2 belongs to X; (iii) $\bot \notin X$; and (iv) it is not the case that $p, \neg p \in X$ for some proposition letter p.*
2. *An* L-type *is a locally-consistent subset of* $\text{SUBF}(\text{NNF}(\varphi))$*, and an* R-type *is a locally-consistent subset of* $\text{SUBF}(\text{NNF}(\neg\psi))$*. A combined* type *(or just* type *for short) is a pair $\tau = (\tau_L, \tau_R)$ where τ_L is an L-type and τ_R is an R-type.*
3. *A type $\tau = (\tau_L, \tau_R)$ is* overlap-consistent *if there does not exist a proposition letter p such that $p \in \tau_L$ and $\neg p \in \tau_R$ or vice versa.*

It is worth pointing out that the above definition of *local consistency* is closely related to the definition of *decisive refinements* used in the proof of Theorem 6.7. Moreover, both are variants of the concept of *Hintikka sets* that is typically used in completeness proof for semantic tableau systems.

Note that the "R" in *R-types* is not related to the accessibility relation in a Kripke model, which is coincidentally also usually denoted by R.

▶ **Definition 7.3** (Type of a world). *Given a Kripke model $M = (W^M, R^M, V^M)$, and $v \in W^M$, the type $\tau(v) = (\tau_L(v), \tau_R(v))$ is defined as follows: $\tau_L(v)$ is the set of formulas from* $\text{SUBF}(\text{NNF}(\varphi))$ *true at v, and $\tau_R(v)$ is the set of formulas from* $\text{SUBF}(\text{NNF}(\neg\psi))$ *true at v. By construction, this is an overlap-consistent type.*

▶ **Definition 7.4** (Viable successor). *Let τ, τ' be D-types, for $D \in \{L, R\}$. We say that τ' is a viable successor for τ (notation: $\tau \Rightarrow \tau'$) if for every formula of the form $\Box\chi$ belonging to τ, χ belongs to τ'. This definition extends naturally to combined types: we say that $\tau \Rightarrow \tau'$ holds if $\tau_L \Rightarrow \tau'_L$ and $\tau_R \Rightarrow \tau'_R$.*

▶ **Definition 7.5** (Quasi-model). *A* quasi-model *is a set $\mathcal{X}$ of (combined) types, such that*

1. *Every $\tau \in \mathcal{X}$ is overlap-consistent.*
2. *For all $\tau \in \mathcal{X}$ and $\Diamond\chi \in \tau_D$ (with $D \in \{L, R\}$), there is $\tau' \in \mathcal{X}$ such that $\tau \Rightarrow \tau'$ and $\chi \in \tau'_D$.*

Note that the empty set of types is, in particular, a quasi-model.

This completes the definition of quasi-models. The next theorem states that quasi-models can be used to decide validity of $\varphi \to \psi$.

▶ **Theorem 7.6. (Soundness and completeness of quasi-models)** *The modal formula $\varphi \wedge \neg\psi$ is satisfiable if and only if there is a quasi-model $\mathcal{X}$ with a type $\tau = (\tau_L, \tau_R) \in \mathcal{X}$ such that $\text{NNF}(\varphi) \in \tau_L$ and $\text{NNF}(\neg\psi) \in \tau_R$.*

Proof. (sketch) In one direction, let $M = (W^M, R^M, V^M)$ be a Kripke model and suppose $M, w \models \varphi \wedge \neg\psi$ for some $w \in W^M$. Let $\mathcal{X} = \{\tau(v) \mid v \in W^M\}$. Then it is easy to show that $\mathcal{X}$ is a quasi-model. Note that whenever vR^Mu, then $\tau(u)$ is indeed a viable successor for $\tau(v)$.

Conversely, suppose $\mathcal{X}$ is a quasi-model with a type $\tau = (\tau_L, \tau_R) \in \mathcal{X}$ such that $\text{NNF}(\varphi) \in \tau_L$ and $\text{NNF}(\neg\psi) \in \tau_R$. We construct a model as follows: for each $\tau' \in \mathcal{X}$ we create a world $w_{\tau'}$. The accessibility relation connects a pair of worlds $(w_{\tau'}, w_{\tau''})$ whenever $\tau' \Rightarrow \tau''$. Finally, a proposition letter p is set to true at $w_{\tau'}$ whenever $p \in \tau'_L \cup \tau'_R$. It can be shown by induction that the resulting model $M_{\mathcal{X}}$ satisfies the following truth lemma:

$$M_{\mathcal{X}}, w_{\tau'} \models \chi \text{ for all } \chi \in \tau'_L \cup \tau'_R$$

In particular, it follows that $M_{\mathcal{X}}, w_\tau \models \varphi \wedge \neg\psi$. ◀

Clearly, a quasi-model is a finite object. More precisely, since a type is a polynomial-sized object, a quasi-model is an object of singly exponential size (as a function of the size of φ and ψ). Therefore, Theorem 7.6 provides us with a decision procedure for testing satisfiability of $\varphi \wedge \neg\psi$. The immediate upper bound we get from this theorem is NExpTime (by non-deterministically guessing the quasi-model). In fact, we can do a little better.

▶ **Definition 7.7** (Type elimination sequence). *A* type elimination sequence *is a sequence* $\mathcal{X}_0, \ldots, \mathcal{X}_n$ *where*

1. $\mathcal{X}_0$ *is the set of all types (for the given formulas* φ, ψ*),*
2. *each* $\mathcal{X}_{i+1}$ *is obtained from* $\mathcal{X}_i$ *by removing a type* $\tau \in \mathcal{X}_i$ *that fails to satisfy condition 1 or 2 from the definition of quasi-models.*
3. $\mathcal{X}_n$ *is a quasi-model.*

It is easy to see that a type elimination sequence always exists (note that $\mathcal{X}_0$ is finite and that $\mathcal{X}_n$ may be empty). Even if, for a given choice of φ and ψ, there does not exist *any* type, there is still a type elimination sequence, namely where $\mathcal{X}_0 = \mathcal{X}_n = \emptyset$.

▶ **Theorem 7.8.**

1. *All type elimination sequences (for the given formulas* φ, ψ*) end in the same quasi-model* $\mathcal{X}_n$*, which can equivalently be characterized as the maximal quasi-model, and as the union of all quasi-models.*
2. *The modal formula* $\varphi \wedge \neg\psi$ *is satisfiable if and only if* $\mathcal{X}_n$ *contains a type* $\tau = (\tau_{\text{L}}, \tau_{\text{R}})$ *with* $\text{NNF}(\varphi) \in \tau_{\text{L}}$ *and* $\text{NNF}(\neg\psi) \in \tau_{\text{R}}$.

Proof. For Item 1, it suffices to observe that, if $\mathcal{X}_0, \ldots, \mathcal{X}_n$ is a type elimination sequence and $\mathcal{X}$ is any quasi-model, then, $\mathcal{X} \subseteq \mathcal{X}_i$ for all $i \leq n$. Indeed, this can be shown by a straightforward induction on i.

For Item 2, if $\varphi \wedge \neg\psi$ is satisfiable, then let $\mathcal{X}$ be the quasi-model given by Theorem 7.6. By Item 1, we have that $\mathcal{X} \subseteq \mathcal{X}_n$, and hence, $\mathcal{X}_n$ contains a type $\tau = (\tau_{\text{L}}, \tau_{\text{R}})$ with $\text{NNF}(\varphi) \in \tau_{\text{L}}$ and $\text{NNF}(\neg\psi) \in \tau_{\text{R}}$. If, on the other hand, $\varphi \wedge \neg\psi$ is *not* satisfiable, then it follows from Theorem 7.6 that there is no quasi-model containing a type τ with $\text{NNF}(\varphi) \in \tau_{\text{L}}$ and $\text{NNF}(\neg\psi) \in \tau_{\text{R}}$. In particular, $\mathcal{X}_n$ does not contain such a type. ◀

This puts the complexity in ExpTime: it suffices to construct an arbitrary type elimination sequence and inspect the final type set of the sequence. Note that the length of the sequence is single exponential because one type gets eliminated each step. Note that this complexity upper bound is still not optimal, because the satisfiability problem for the modal logic K is in PSpace. However, the quasi-model method is quite generic and can be adapted to various extensions of modal logic that are ExpTime-complete, such as the extension with the global modality, or Propositional Dynamic Logic (PDL).

Next, we explain how to construct interpolants from type elimination sequences. In fact we will show how to construct Lyndon interpolants. Recall that a *Lyndon interpolant* for a valid implication $\varphi \to \psi$ is a formula ϑ such that $\varphi \to \vartheta$ and $\vartheta \to \psi$ are valid, and such that every proposition letter occurring positively (negatively) in ϑ occurs positively (negatively) in both φ and ψ. In particular, a Lyndon interpolant is a Craig interpolant (but not vice versa).

Fix modal formulas φ, ψ such that $\varphi \to \psi$ is valid. By Theorem 7.8 there is a type elimination sequence $\mathcal{X}_0, \ldots, \mathcal{X}_n$ such that $\mathcal{X}_n$ does not contain any type τ with $\text{NNF}(\varphi) \in \tau_{\text{L}}$ and $\text{NNF}(\neg\psi) \in \tau_{\text{R}}$. For the remainder of this section, we can fix such a sequence.

The core result is:

▶ **Theorem 7.9.** *If a type $\tau = (\tau_L, \tau_R)$ gets eliminated in the sequence, then there is a modal formula ϑ_τ, such that*

- $\models (\bigwedge \tau_L) \to \vartheta_\tau$ *and*
- $\models \vartheta_\tau \to \neg(\bigwedge \tau_R)$ *and*
- *Every proposition letter occurring positively (negatively) in ϑ_τ occurs positively (negatively) in both φ and ψ.*

Proof. The proof is by induction on the stage at which the type gets eliminated. If a type τ is eliminated, this is for one of two reasons:

1. τ is not overlap-consistent. If $p \in \tau_L$ and $\neg p \in \tau_R$, we pick $\vartheta_\tau = p$. Note that $p \in \text{LITERALS}(\varphi)$ and $\neg p \in \text{LITERALS}(\text{NNF}(\neg\psi))$. Since polarity of proposition letter occurrences is preserved when bringing formulas into negation normal form, and polarity of proposition letter occurrences is inverted when negating a formula, it follows that p occurs positively in both φ and ψ. The case where $\neg p \in \tau_L$ and $p \in \tau_R$ is similar, except that we choose $\vartheta_\tau = \neg p$. By analogous reasoning, in this case, p must occur negatively in both φ and ψ.

2. For some $D \in \{L, R\}$, τ_D contains a formula of the form $\Diamond\chi$, and every type $\tau' = (\tau'_L, \tau'_R)$ satisfying $\tau \Rightarrow \tau'$ and $\chi \in \tau'_D$ has already been eliminated earlier on in the elimination sequence. In this case, we proceed as follows. If $D = L$, we take

$$\vartheta_\tau \quad = \quad \Diamond \bigvee_{\substack{X \text{ an L-type with} \\ \chi \in X \text{ and } \tau_L \Rightarrow X}} \bigwedge_{\substack{\tau'_R \text{ an R-type with} \\ \tau_R \Rightarrow Y}} \vartheta_{(X,Y)}$$

We need to show that (i) $\models \bigwedge \tau_L \to \vartheta_\tau$ and (ii) $\models \vartheta_\tau \to \neg(\bigwedge \tau_R)$.

For (i), let $M, w \models \bigwedge \tau_L$, where $M = (W^M, R^M, V^M)$. Then $M, w \models \Diamond\chi$. Let v be a witnessing successor, i.e., $(w, v) \in R^M$ and $M, v \models \chi$. Let $X = \tau_L(v)$ be the L-type of v. Note that, by construction, $\tau_L \Rightarrow X$. By induction hypothesis, we have that $\models (\bigwedge X) \to \vartheta_{(X,Y)}$ for all R-types Y such that $\tau' = (X, Y)$ got eliminated previously. It follows that

$$M, v \models \bigwedge_{Y \text{ an R-type with } \tau_R \Rightarrow Y} \vartheta_{(X,Y)}$$

Hence, $M, w \models \vartheta_\tau$.

For (ii), by contraposition, let $M, w \models \bigwedge \tau_R$. We need to show that $M, w \not\models \vartheta_\tau$. Let v be any successor of w, and let $Y = \tau_R(v)$ be the R-type of v. Observe that $\tau_R \Rightarrow Y$. It follows by the induction hypothesis that $M, v \not\models \vartheta_{(X,Y)}$, for all X such that (X, Y) got eliminated. Hence, $M, w \not\models \vartheta_\tau$.

This concludes the argument for $D = L$. If $D = R$, the argument is analogous, except that we reason dually, and we now take

$$\vartheta_\tau \quad = \quad \Box \bigvee_{\substack{X \text{ an L-type with} \\ \tau_L \Rightarrow X}} \bigwedge_{\substack{Y \text{ an R-type with} \\ \chi \in Y \text{ and } \tau_R \Rightarrow Y}} \vartheta_{(X,Y)}$$

We omit the details, which are straightforward. ◀

From this theorem we get, as a corollary, the Lyndon interpolation theorem.

▶ **Corollary 7.10** (Lyndon interpolation). *If* $\models \varphi \to \psi$*, then*

$$\vartheta = \bigvee_{\substack{X \text{ an L-type} \\ \text{with } \mathrm{NNF}(\varphi) \in X}} \bigwedge_{\substack{Y \text{ an R-type} \\ \text{with } \mathrm{NNF}(\neg\psi) \in Y}} \vartheta_{(X,Y)}$$

is a Lyndon interpolant for $\varphi \to \psi$*.*

Proof. If $\varphi \wedge \neg\psi$ is unsatisfiable, then the type elimination sequence ends in a quasi-model that does not contain a type $\tau = (\tau_{\mathrm{L}}, \tau_{\mathrm{R}})$ with $\varphi \in \tau_{\mathrm{L}}$ and $\mathrm{NNF}(\neg\psi) \in \tau_{\mathrm{R}}$. Therefore, every such type gets eliminated. This shows that the above formula ϑ is indeed well defined.

To see that $\models \varphi \to \vartheta$, let $M, w \models \varphi$. Let $X = \tau_{\mathrm{L}}(w)$ be the L-type of w. By construction, $M, w \models \bigwedge X$. It follows from Theorem 7.9 that, for all R-types Y containing $\mathrm{NNF}(\neg\psi)$, because (X, Y) got eliminated, $M, w \models \vartheta_{(X,Y)}$. It follows that $M, w \models \vartheta$.

To see that $\models \vartheta \to \psi$, by contraposition, let $M, w \not\models \psi$. Let $Y = \tau_{\mathrm{R}}(w)$ be the R-type of w. By construction, $M, w \models \bigwedge Y$. It follows by Theorem 7.9 that $M, w \not\models \vartheta_{(X,Y)}$ for all L-types X containing $\mathrm{NNF}(\varphi)$. Therefore, $M, w \not\models \vartheta$. ◀

By a careful inspection of the above procedure it can be shown that this yields a method for constructing interpolants in exponential time, provided that the interpolants are represented succinctly using a DAG-style representation of formulas.

The method described here generalizes to the guarded-negation fragment [3] and yields an effective procedure for interpolant construction that is optimal in terms of formula length and running time, in that context. It seems likely that the above method can be adapted to other modal logics, as well as extensions of the basic modal language, that admit filtration. Also, unlike the model-theoretic approach (cf. Section 3), the technique presented here is not inherently restricted to fragments of *first-order* logic. We leave it as an open question whether the present technique be used to prove a general interpolation result for a larger class of modal logics that admit filtration.

In summary:

- ✓ The present technique is likely to generalize to other modal logics that admit filtration. It also generalizes to richer languages such as the guarded-negation fragment of first-order logic.

- ✓ The present proof technique yields an algorithm to compute Lyndon interpolants for K in singly exponential time and of singly-exponential size, if a DAG-representation is allowed.

✓ The present proof technique can also be adapted for deciding the existence of Craig interpolants for certain extensions of modal logic that lack Craig interpolation [26].

✗ This proof technique does not extend in any obvious way to prove uniform interpolation.

8 Sixth Proof (via Modal Algebras and Duality Theory)

Many notions of modal logic can be translated into algebraic concepts in terms of modal algebras. In particular, as we will see below, a modal logic L has the Craig interpolation property iff the class of L-algebras—modal algebras corresponding to this logic—have the superamalgamation property. Thus, for showing that the basic modal logic K has Craig interpolation, it is sufficient to show that its algebraic models, modal algebras, have the superamalgamation property. However, in order to show that the class of modal algebras has the superamalgamation property, the easiest is to work with duals of modal algebras—modal spaces. Using the spatial intuition of modal spaces we can construct algebras that ensure that modal algebras have the superamalgamation property. As a corollary, we obtain that the basic modal logic K has Craig interpolation. In this section, we spell out this approach.

8.1 Algebraic Semantics of Modal Logic

Recall that a *Boolean algebra* is a set A equipped with binary operations $\vee$ and $\wedge$, a unary operation $-$, and designated elements 0 and 1, satisfying the commutative and associative laws for $\vee$ and $\wedge$, distribution of $\vee$ over $\wedge$ and vice versa, and a few other equations. A *modal algebra* is a Boolean algebra with an additional unary operation $\Diamond$, such that $\Diamond$ commutes with finite joins (i.e., $\Diamond(a \vee b) = \Diamond a \vee \Diamond b$) and $\Diamond 0 = 0$. We will denote such modal algebras by $\mathfrak{A} = (A, \vee, \wedge, -, 0, 1, \Diamond,)$ or just $\mathfrak{A} = (A, \Diamond)$. Note also that on each Boolean algebra we can also define the Boolean implication $a \to b = -a \vee b$ and on a modal algebra we can define the modal Box: $\Box a = -\Diamond - a$. A *homomorphism* $h : \mathfrak{A}_1 \to \mathfrak{A}_2$, as usual in universal algebra, is a map from the underlying set of $\mathfrak{A}_1$ to the underlying set of $\mathfrak{A}_2$ that commutes with the operations (i.e., $\vee$, $\wedge$, $-$, and $\Diamond$) and that maps the designated elements of $\mathfrak{A}_1$ (i.e., the 0 and 1) to the corresponding designated elements of $\mathfrak{A}_2$.

Recall from Section 2 that a *normal modal logic* is a set L of modal formulas that includes all substitution instances of propositional tautologies as well as all formulas of the form $\Box(\chi_1 \to \chi_2) \to (\Box\chi_1 \to \Box\chi_2)$ and is closed under modus ponens, necessitation, and substitution; and that K is the least normal modal logic.

Let $(A, \Diamond)$ be a modal algebra. A *valuation* on A is a map $v : \mathsf{Prop} \to A$, where Prop is the set of propositional variables. This map can be extended to the set of all modal formulas in an obvious way. We say that a modal formula φ is *valid* on a modal algebra $(A, \Diamond)$ (written $(A, \Diamond) \models \varphi$) if $v(\varphi) = 1$ for any valuation v on A. A formula φ is *valid* on a class of algebras $\mathcal{C}$ (written $\mathcal{C} \models \varphi$) if φ is valid on every algebra in $\mathcal{C}$.

A normal modal logic L is sound and complete with respect to a class $\mathcal{C}$ of modal algebras if for each modal formula φ we have

$$\vdash_L \varphi \text{ iff } \mathcal{C} \models \varphi$$

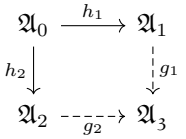
For any normal modal logic L, an *L-algebra* is a modal algebra that validates all the formulas in L. In particular, for the basic modal logic K the class of K-algebras coincides with the class of all modal algebras. For the next result we refer to e.g., [5, Chapter 5] or [9, Chapter 7].

▶ **Theorem 8.1** (Algebraic completeness)**.** *Every normal modal logic L is sound and complete with respect to the class of L-algebras.*

We note, on the other hand, that there exist modal logics that are Kripke incomplete [9, Chapter 6], [5, Chapter 4]. In fact, there are continuum many such logics [30].

In order to state the next result we need to introduce the notion of *amalgamable* and *superamalgamable* classes of algebras. Recall that an *embedding* is an injective homomorphism.

▶ **Definition 8.2.** *Let $\mathcal{C}$ be a class of algebras. We say that $\mathcal{C}$ is* amalgamable *if for each $\mathfrak{A}_0$, $\mathfrak{A}_1, \mathfrak{A}_2 \in \mathcal{C}$ such that $\mathfrak{A}_0$ is embedded into $\mathfrak{A}_1$ via a homomorphisms h_1 and $\mathfrak{A}_0$ is embedded into $\mathfrak{A}_2$ via a homomorphism h_2, there is an algebra $\mathfrak{A}_3 \in \mathcal{C}$ such that $\mathfrak{A}_1$ is embedded into $\mathfrak{A}_3$ via a homomorphism g_1 and $\mathfrak{A}_2$ is embedded into $\mathfrak{A}_3$ via a homomorphism g_2 such that for each $a \in \mathfrak{A}_0$ we have $g_1(h_1(a)) = g_2(h_2(a))$.*



Note that the above definition of amalgamation applies to any class of universal algebras. If the conditions of Definition 8.2 are satisfied, the triple $\mathfrak{A}_0, \mathfrak{A}_1, \mathfrak{A}_2$ is called an *amalgamable triple*. The algebra $\mathfrak{A}_3$ is called the *amalgam* of the diagram or the *algebra amalgamating the diagram*.

The next definition of superamalgamation presumes that the algebras in the class are lattice-based with $\leq$ being a lattice induced partial order. We note that for modal algebras $a \leq b$ iff $a \wedge b = a$ iff $a \vee b = b$ iff $a \to b = 1$. We will denote by $\leq_i$ the lattice order of an algebra $\mathfrak{A}_i$.

▶ **Definition 8.3.** *Let $\mathcal{C}$ be a class of modal algebras. We say that $\mathcal{C}$ is* superamalgamable *if it is amalgamable and, in addition, if for each amalgamable triple $\mathfrak{A}_0$, $\mathfrak{A}_1$, $\mathfrak{A}_2 \in \mathcal{C}$ such that $\mathfrak{A}_0$ is embedded into $\mathfrak{A}_1$ via h_1 and $\mathfrak{A}_0$ is embedded into $\mathfrak{A}_2$ via h_2, amalgamable by $\mathfrak{A}_3$ via the embeddings $g_1 : \mathfrak{A}_1 \to \mathfrak{A}_3$ and $g_2 : \mathfrak{A}_2 \to \mathfrak{A}_3$, and for each $a \in \mathfrak{A}_i$, $b \in \mathfrak{A}_j$, $\{i, j\} = \{1, 2\}$ we have*

$$g_i(a) \leq_3 g_j(b) \text{ implies } \exists c \in \mathfrak{A}_0 \text{ such that } a \leq_i h_i(c) \text{ and } h_j(c) \leq_j b.$$

Clearly, every class $\mathcal{C}$ which is supermalgamable, is also amalgamable. The next theorem connects superamalgamability with the Craig interpolation property.

▶ **Theorem 8.4.** *A normal modal logic L has the Craig interpolation property iff the class of all L-algebras is superamalgamable.*

Proof. For the proof we refer to [9, Chapter 14], [18, Chapter 7], and [45, Section 6.3]. We only sketch here why superamalgamation of the class of all L-algebras entails that a modal logic L has the interpolation property. We rely on the construction of the Lindenbaum–Tarski algebra for a modal logic L. The Lindenbaum–Tarski algebra $F_L(\vec{p})$ is the algebra of formulas in variables $\vec{p}$ moded out by L-equivalence. Moreover, it is well known that $\vdash_L \varphi(\vec{p}) \to \psi(\vec{p})$ iff $F_L(\vec{p}) \models \varphi(\vec{p}) \to \psi(\vec{p})$ iff $[\varphi(\vec{p})] \leq [\psi(\vec{p})]$ in $F_L(\vec{p})$, where $[\varphi]$ and $[\psi]$ are the elements of $F_L(\vec{p})$ corresponding to formulas φ and ψ, respectively.

Now suppose $\vdash_L \varphi(\vec{p}, \vec{q}) \to \psi(\vec{q}, \vec{r})$. Let $F_L(\vec{q})$, $F_L(\vec{p}, \vec{q})$ $F_L(\vec{q}, \vec{r})$ be the Lindenbaum–Tarski algebras of L (see e.g., [5, 28, 9]) in the variables $\vec{q}$, and $\vec{p}, \vec{q}$ and $\vec{q}, \vec{r}$, respectively. There is a natural embedding of $F_L(\vec{q})$ into $F_L(\vec{p}, \vec{q})$ and of $F_L(\vec{q})$ into $F_L(\vec{q}, \vec{r})$. Then it can be shown that the algebra $\mathfrak{A}_3$ amalgamating this diagram, is a subalgebra of $F_L(\vec{p}, \vec{q}, \vec{r})$—the Lindenbaum–Tarski algebra of L in the variables $\vec{p}$, $\vec{q}$ and $\vec{r}$. We omit the details.

Finally, by just unfolding the superamalgamation property we obtain that there is a formula $\chi(\vec{q})$ in the Lindenbaum-Tarski algebra $F_L(\vec{q})$ (formally speaking an equivalence class of formulas in this algebra), such that $\vdash_L \varphi(\vec{p}, \vec{q}) \to \chi(\vec{q})$ and $\vdash_L \chi(\vec{q}) \to \psi(\vec{q}, \vec{r})$, entailing that L has the interpolation property. ◀

Thus, in order to show that K has the interpolation property, it is sufficient to show that the class of all modal algebras has the superamalgamation property. For this, we will make use of duality theory, a powerful technique that has many applications all over algebraic logic.

8.2 Topological Duality

In this section we briefly recall the basic definitions and facts underlying the topological duality for modal algebras. We start by reviewing Stone duality. For more information we refer to [13, 24].

A *topological space* is a pair (X, τ) where X is a set and τ is a collection of subsets of X (called *open sets*) satisfying the usual axioms: the empty set and X itself are open; arbitrary unions of open sets are open; and finite intersections of open sets are open. A subset $C \subseteq X$ is *closed* if its complement $X \setminus C$ is open. A space is *compact* if every open cover has a finite subcover, and *Hausdorff* if any two distinct points have disjoint open neighborhoods. A *Stone space* is a compact Hausdorff space that is *totally separated*, meaning that distinct points are separated by a clopen (simultaneously closed and open) set.

Stone spaces provide the topological side of *Stone duality*, which establishes a correspondence between Boolean algebras and such spaces. Given a Boolean algebra A, its *Stone space*

(or *Stone dual*) A_* is the set of all *ultrafilters* on A, that is, maximal proper filters $x \subseteq A$ satisfying $a \in x$ iff $\neg a \notin x$. The topology on A_* is generated by basic open sets of the form

$$\widehat{a} = \{\, x \in A_* : a \in X \,\},$$

for $a \in A$. Each $\widehat{a}$ is clopen, which ensures that B_* is totally separated. Conversely, every Stone space X gives rise to a Boolean algebra $\mathsf{Clop}(X)$ of clopen subsets, and these two constructions establish a dual equivalence between the categories of Boolean algebras and Stone spaces.

Here we will also like to highlight a useful separation property of Stone spaces, which we will use below in an essential way (see e.g., [16]).

▶ **Proposition 8.5.** *Let X be a Stone space. For each pair of disjoint closed sets $F, G \subseteq X$, there exists a clopen set $C \subseteq X$ such that $F \subseteq C$ and $G \cap C = \varnothing$.*

The crucial idea of the Jónsson–Tarski duality, is that modal algebras can be represented as algebras of clopen sets of modal spaces—Stone spaces equipped with a binary relation. We refer to [5, 9, 28, 45] for the basic theory of modal algebras, modal spaces, and the dual equivalence between modal algebras and modal spaces.

A *modal space* (or *descriptive Kripke frame*) is a pair $\mathfrak{X} = (X, R)$, where X is a Stone space and R is a binary relation on X satisfying

$$R[x] := \{y \in X : xRy\}$$

is closed for each $x \in X$ and

$$R^{-1}[U] := \{x \in X : \exists y \in U \text{ with } xRy\}$$

is clopen for each clopen U of X. A *bounded morphism* (or *p-morphism*) between two modal spaces is a continuous map f such that $f(R[x]) = R[f(x)]$. Let MS be the category of modal spaces and bounded morphisms.

It is a well-known theorem in modal logic that MA is dually equivalent to MS. The functors $(-)_* : \mathsf{MA} \to \mathsf{MS}$ and $(-)^* : \mathsf{MS} \to \mathsf{MA}$ that establish this dual equivalence are constructed as follows. For a modal algebra $\mathfrak{A} = (A, \Diamond)$, let $\mathfrak{A}_* = (A_*, R)$, where A_* is the Stone space of A and

$$xRy \quad \text{iff} \quad \forall a \in A,\ (a \in y \Rightarrow \Diamond a \in x).$$

We call R the *dual* of $\Diamond$. For a modal homomorphism $h : A \to B$, its dual $h_* : B_* \to A_*$ is given by h^{-1}. For a modal space $\mathfrak{X} = (X, R)$, let $\mathfrak{X}^* = (X^*, \Diamond_R)$, where X^* is the Boolean algebra of clopens of X and

$$\Diamond_R(U) = R^{-1}[U].$$

For a bounded morphism $f : X \to Y$, its dual $f^* : Y^* \to X^*$ is given by f^{-1}, the map that maps each set U to its pre-image $f^{-1}(U)$. Moreover, f is surjective iff f^* is injective and conversely f is injective iff f^* is surjective.

The *canonical frame* of a modal logic L (or of its Lindenbaum–Tarski algebra) is a special instance of this general dual construction. The underlying set of the canonical frame consists precisely of the ultrafilters of the Lindenbaum-Tarski algebra (or, equivalently, maximally L-consistent sets), and the accessibility relation is defined as above. Thus, the familiar canonical model of L in modal logic is just the Jónsson–Tarski dual of its Lindenbaum-Tarski algebra.

Note that one can also construct the powerset algebra from a modal space, or any Kripke frame (X, R) in general. The powerset algebra of (X, R) is a the pair $(\mathcal{P}(X), \Diamond)$, where $\mathcal{P}(X)$ is the powerset Boolean algebra and $\Diamond_R$ is defined as $\Diamond_R(U) = R^{-1}[U]$ for each $U \in \mathcal{P}(X)$. It is easy to see that the powerset algebra of every Kripke frame is a modal algebra.

This topological perspective allows one to transfer algebraic properties such as the *superamalgamation property* into the corresponding dual notions, as we will see below.

8.3 Superamalgamation via Duality

We will now show how the above duality techniques can be put to use for proving that the modal logic K has the Craig interpolation property.

▶ **Theorem 8.6.** *The class of all modal algebras is superamalgamable.*

Proof. Consider the diagram of modal algebras $\mathfrak{A}_1$, $\mathfrak{A}_2$ and $\mathfrak{A}_3$, where h_1 and h_2 are embeddings

$$\begin{array}{ccc} \mathfrak{A}_0 & \xrightarrow{h_1} & \mathfrak{A}_1 \\ {\scriptstyle h_2}\downarrow & & \\ \mathfrak{A}_2 & & \end{array}$$

Let (X_i, R_i) be the modal space dual to $\mathfrak{A}_i$. Let $f_1 : X_1 \to X_0$ be the surjective bounded morphism dual to h_1 and $f_2 : X_2 \to X_0$ be the surjective bounded morphism dual to h_2. Then we get the dual diagram

$$\begin{array}{ccc} X_0 & \xleftarrow{f_1} & X_1 \\ {\scriptstyle f_2}\uparrow & & \\ X_2 & & \end{array}$$

Let

$$Y = \{(x, y) : x \in X_1, y \in X_2, f_1(x) = f_2(y)\}$$

we also let $(x, y)R(x', y')$ if xR_1x' and yR_2y'.

$$\begin{array}{ccc} X_0 & \xleftarrow{f_1} & X_1 \\ {\scriptstyle f_2}\uparrow & & \uparrow{\scriptstyle \pi_1} \\ X_2 & \xleftarrow[\pi_2]{} & Y \end{array}$$

If (X_0, R_0), (X_1, R_1) and (X_2, R_2) were just relational structures this would do the job—(Y, R) would co-amalgamate the above diagram, where π_1 and π_2 are projections from Y to X_1 and X_2 respectively. However, (Y, R) has to be a modal space and while Y is a closed subset of $X_1 \times X_2$, and hence a Stone space, the space (Y, R) may not be a modal space—it would be if Y were a generated subframe, i.e., also closed under the product relation $R_1 \times R_2$.

However, one can still use Y to prove the superamalgamation property. We will take the powerset algebra $(\mathcal{P}(Y), \Diamond_R)$. We let $\mathfrak{A}_3 = (\mathcal{P}(Y), \Diamond_R)$ and we show that $\mathfrak{A}_3$ superamalgamates the diagram. Let $\pi : Y \to X_1$ and $\pi_2 : Y \to X_2$ be the two projection maps defined by $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$.

$$\begin{array}{ccc} \mathfrak{A}_0 & \xrightarrow{h_1} & \mathfrak{A}_1 \\ {\scriptstyle h_2}\downarrow & & \downarrow{\scriptstyle g_1} \\ \mathfrak{A}_2 & \xrightarrow[g_2]{} & \mathfrak{A}_3 \end{array}$$

Keeping in mind that $\mathfrak{A}_i$ is isomorphic to $(\mathsf{Clop}(X_i), \Diamond_{R_i})$ for $i = 1, 2, 3$, we also have maps $g_1 : \mathfrak{A}_1 \to \mathfrak{A}_3$ and $g_2 : \mathfrak{A}_2 \to \mathfrak{A}_3$ defined by $g_1(U) = \pi_1^*(U) = \pi_1^{-1}(U)$ and $g_2(U) = \pi_2^*(U) = \pi_2^{-1}(U)$.

We thus have the diagram:

$$\begin{array}{ccc} (\mathsf{Clop}(X_0), \Diamond_{R_0}) & \xrightarrow{f_1^*} & (\mathsf{Clop}(X_1), \Diamond_{R_1}) \\ {\scriptstyle f_2^*}\downarrow & & \downarrow{\scriptstyle \pi_1^*} \\ (\mathsf{Clop}(X_2), \Diamond_{R_2}) & \xrightarrow[\pi_2^*]{} & (\mathcal{P}(Y), \Diamond_R) \end{array}$$

We will now verify the superamalgamation property. Our proof will be topological in nature and it connects the existence of a superamalgam and hence of the Craig interpolant with the topological separation property of Stone spaces highlighted in Proposition 8.5—in Stone spaces disjoint closed sets can be separated by a clopen set. We will show that the existence of this clopen set dually corresponds to the existence of the Craig interpolant.

We first note that, since $f_1(\pi_1(x, y)) = f_1(x) = f_2(y) = f_2(\pi_2(x, y))$, it is easy to see that for each $U \in \mathsf{Clop}(X_0)$ we have $\pi_1^{-1}(f_1^{-1}(U)) = \pi_2^{-1}(f_2^{-1}(U))$, that is, $\pi_1^*(f_1^*(U)) = \pi_2^*(f_2^*(U))$. This shows that modal algebras enjoy the amalgamation property.

Now assume that $\pi_i^*(a) \leq \pi_j^*(b)$. for some $i, j = 1, 2$. Without the loss of generality we may assume that $i = 1$ and $j = 2$. The other case is analogous. So let $\pi_1^*(U) \leq \pi_2^*(V)$ for some $U \in \mathsf{Clop}(X_1)$ and $V \in \mathsf{Clop}(X_2)$. This means that $\pi_1^{-1}(U) \leq \pi_2^{-1}(V)$. Note that U and $X_3 \setminus V$ are clopen and hence closed sets. As every continuous map between compact

Hausdorff spaces is closed (i.e., the direct image of a closed set is closed) [16], $f_1[U]$ and $f_2[X_2 \setminus V]$ are closed subsets of X_0. We claim that they are also disjoint. Suppose not. Then there is $x \in f_1[U] \cap f_2[X_2 \setminus V]$. This yields that there are $y \in U$ and $z \in X_3 \setminus V$ such that $f_1(y) = f_2(z)$. But then $(y, z) \in Y$ and $(y, z) \in \pi_1^{-1}(U)$. Hence, by assumption, $(y, z) \in \pi_2^{-1}(V)$; but then $z \in V$, which contradicts the fact that $z \notin V$.

By Proposition 8.5, there exists a clopen set C such that $f_1[U] \subseteq C$ and $C \cap f_2[X_2 \setminus V] = \varnothing$. We will now show that $U \subseteq_1 f_1^*(C)$ and $f_2^*(C) \subseteq_2 V$. If not, then there is $x \in U$ such that $x \notin f_2^{-1}(C)$. Since $f_1[U] \subseteq C$, we have that $U \subseteq f_1^{-1}f_1[U] \subseteq f_1^{-1}(C)$. On the other hand, $C \cap f_2[X_2 \setminus V] = \varnothing$ implies that $f_1^{-1}(C) \cap f_2^{-1}(f_2[X_2 \setminus V]) = \varnothing$. This yields that $f_2^{-1}(C) \cap (X_2 \setminus V)) = \varnothing$ and so $f_2^{-1}(C) \subseteq V$, which finishes the proof (via duality) that the class of modal algebras are superamalgamble. ◀

Thus we obtain the following direct corollary to Theorems 8.1, 8.4 and 8.6.

▶ **Corollary 8.7.** *The modal logic* K *has the Craig interpolation property.*

The method described here is algebraic and uses duality between modal algebras and modal spaces, it also uses essentially topological intuitions and topological properties of Stone spaces. In fact, it shows that the interpolant corresponds topologically to the clopen set obtained via a separation property, as a "separator" of two disjoint closed sets.

The proof of Theorem 8.6 shows that for a modal logic L and the amalgamable triple of L-algebras $\mathfrak{A}_0$ $\mathfrak{A}_1$ and $\mathfrak{A}_2$, if the algebra $\mathfrak{A}_3 = (\mathcal{P}(Y), \Diamond_R)$, constructed in the proof of the theorem, is an L-algebra, then the class of L-algebras is superamalgamable and hence the logic L has the Craig interpolation property. Note that the frame (Y, R) is very similar to the bisimulation product constructed in the model-theoretic proof of Section 3. Also, similarly to the model theoretic case, if a logic L is axiomatized by modal formulas whose first-order correspondents are universal Horn sentences (see Section 3), then it is easy to see that the algebra $\mathfrak{A}_3 = (\mathcal{P}(Y), \Diamond_R)$ is an L-algebra, and hence L has the Craig interpolation property (see also [45, Section 6.3]). Thus, the two proofs—algebraic via duality and the model theoretic one—can be seen as two sides of the same coin.

- The present proof technique is semantic in nature. The algebraic approach towards interpolation via amalgamation and superamalgamation apply to many systems of non-classical logic admitting algebraic semantics, even if they do not possess Kripke semantics (see Chapter 3 and Chapter 8 in this volume).

- As the semantic condition of superamalgamation is necessary and sufficient for the logic to have the Craig interpolation, this method could also be used to show negative results, that is, that a given logic (or a class of logics) does not have the interpiolation property. For example, using the algebraic and duality techniques Maksimova [32, 33] showed that there are exactly 7 consistent superintuitionistic logics and that among the extensions of the modal logic S4 there are at most 37 logics with the interpolation property (see also Chapter 3 in this volume).

⊗ The interpolants constructed in this section are highly non-constructive. Thus, the method does not provide an algorithm for finding the interpolants.

9 Lower Bounds

As we have seen, some Craig interpolation techniques (such as the syntactic approach using ∇-normal form) yield interpolants of exponential formula length, and some others (such as the approach using quasi-models) yield interpolants that, although not necessarily of exponential length, admit a DAG-representation of singly exponential size.[8] In this section, we present a matching size lower bound. Specifically, the following result, due to Frank Wolter *(personal communication)*, shows that Craig interpolants for modal implications are in the worst case necessarily exponentially large, even when the size of a formula is measured by its DAG-representation. Note that no such unconditional lower bound is known for propositional logic (cf. Chapter 2).

▶ **Theorem 9.1.** *There is a sequence of valid modal implications* $(\varphi_n \to \psi_n)_{n=1,2,\ldots}$, *such that*

1. $|\varphi_n|$ *and* $|\psi_n|$ *are bounded by a polynomial function in* n,
2. *Every Craig interpolant* ϑ *for* $\varphi_n \to \psi_n$ *has DAG-size* $|\vartheta|_{DAG} \geq 2^n$.

Proof. Take the following modal formulas with proposition letters $\{p_1, \ldots, p_n, s, q_1, \ldots, q_n\}$:

$$\varphi_n = \Diamond s \wedge \bigwedge_{i=1\ldots n} ((p_i \to \Box(s \to p_i)) \wedge (\neg p_i \to \Box(s \to \neg p_i)))$$

$$\psi_n = \Big(\bigwedge_{i=1\ldots n} ((p_i \to \Box q_i) \wedge (\neg p_i \to \Box \neg q_i)) \Big) \to \Diamond \bigwedge_{i=1\ldots n} (p_i \leftrightarrow q_i)$$

$$\chi_n = \bigvee_{X \subseteq \{1,\ldots,n\}} (\tau_X \wedge \Diamond \tau_X) \qquad \text{where } \tau_X = \bigwedge_{i \in X} p_i \wedge \bigwedge_{i \notin X} \neg p_i.$$

Note that the formulas φ_n, ψ_n are indeed of size polynomial in n. It is also easy to see that $\varphi_n \to \psi_n$ is a valid modal implication and that χ_n is a Craig interpolant for it. Therefore, it remains only to establish the following two claims.

Claim 1: Every Craig interpolant for $\models \varphi_n \to \psi_n$ is equivalent to χ_n

Proof of claim: Let θ_n be a Craig interpolant for $\varphi_n \to \psi_n$. We must show that θ_n is equivalent to χ_n. We do this by showing that every pointed Kripke model satisfying χ_n can be expanded so that it satisfies φ_n, while each pointed Kripke model satisfying $\neg\chi_n$ can

[8] It is not inconceivable that a more fine-grained analysis would show that the approaches that yield an exponential size DAG representation in fact yield an exponential length formula.

be expanded so that it satisfies $\neg\psi_n$. Since χ_n lies sandwiched between φ_n and ψ_n, it then follows that χ_n and θ_n are equivalent.

More precisely, let M, w be any pointed Kripke model in the propositional signature $\{p_1, \ldots, p_n\}$. If $M, w \models \chi_n$, then we can expand M, w into a model M', w over the extended signature $\{p_1, \ldots, p_n, s\}$ by making s true (only) at some R-successor of w that satisfies the same proposition letters as w. We then have that $M', w \models \varphi_n$ and hence $M', w \models \theta_n$. Since s does not occur in θ_n, it follows that $M, w \models \theta_n$. If, on the other hand, $M, w \not\models \chi_n$, we can expand M, w into a model M'', w over the extended signature $\{p_1, \ldots, p_n, q_1, \ldots, q_n\}$ by making q_i true everywhere if p_i is true at w and making q_i true nowhere otherwise. In this way, we have that $M'', w \not\models \psi_n$ and hence $M'', w \not\models \theta_n$. Since $q_1, \ldots, q_n$ do not occur in θ_n, it follows that $M, w \not\models \theta_n$.

Claim 2: Every modal formula equivalent to χ_n has DAG-size at least 2^n.

Proof of claim: Assume for the sake of a contradiction that χ_n is equivalent to ϑ_n for some modal formula ϑ_n of DAG-size strictly less than 2^n. Consider the Kripke model $M_n = (W_n, R_n, V_n)$ with

- $W_n = \{w\} \cup \{v_X \mid X \subseteq \{1, \ldots, n\}\} \cup \{u_X \mid X \subseteq \{1, \ldots, n\}\}$
- $R_n = \{(w, v_X) \mid X \subseteq \{1, \ldots, n\}\} \cup \{(v_X, u_Y) \mid X, Y \subseteq \{1, \ldots, n\}\}$; and
- $V_n(p_i) = \{v_X \mid i \in X\} \cup \{u_X \mid i \in X\}$.

By construction, we have that $M_n, w \models \Box\chi_n$, and for every submodel M'_n of M_n obtained by deleting one or more worlds of the form u_X, we have that $M'_n, w \not\models \Box\chi_n$. At the same time, since ϑ_n has DAG-size strictly smaller than 2^n, it has strictly less than 2^n subformulas, and therefore, there exists a world u_X such that every subformula of ϑ_n true in u_X is also true in $u_{X'}$ for some $X' \neq X$. It follows that deleting the world u_X from the model M_n does not affect the truth of $\Box\vartheta_n$, and hence of $\Box\chi_n$, at w, a contradiction. ◀

The same argument is used in [29, Lemma 22] to establish an exponential succinctness gap between modal logic and the two-variable fragment of first-order logic. Additional recent results on the size of Craig interpolants and uniform interpolants are reported in [8].

Acknowledgments

The authors are very grateful to Roman Kuznets and Patrick Koopman for the careful reading of the drafts of this paper and for their many helpful comments and corrections. They also wish to thank Rodrigo Almeida, Konstantinos Papafilippou, and Frank Wolter for valuable discussions and suggestions.

References

1 Jon Barwise and Lawrence Moss. *Vicious Circles*, volume 60 of *CSLI Lecture Notes*. CSLI Publications, 1996.

2 Jon Barwise and Johan van Benthem. Interpolation, preservation, and pebble games. *The Journal of Symbolic Logic*, 64(2):881–903, 1999. URL: `http://www.jstor.org/stable/2586507`.

3 Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. Effective interpolation and preservation in guarded logics. *ACM Trans. Comput. Logic*, 17(2), 2015. `doi:10.1145/2814570`.

4 Christoph Berkholz, Dietrich Kuske, and Christian Schwarz. Modal Logic Is More Succinct Iff Bi-Implication Is Available in Some Form. In *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.STACS.2024.12`.

5 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, 2001.

6 Balder ten Cate. *Model Theory for Extended Modal Languages*. PhD thesis, University of Amsterdam, 2005.

7 Balder ten Cate, Willem Conradie, Maarten Marx, and Yde Venema. Definitorially complete description logics. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 79–89. AAAI Press, 2006. URL: `http://www.aaai.org/Library/KR/2006/kr06-011.php`.

8 Balder ten Cate, Louwe Kuijer, and Frank Wolter. The size of interpolants in modal logics, 2026. Under submission. `arXiv:2511.04577`.

9 Alexander Chagrov and Michael Zakharyaschev. *Modal Logic*. The Clarendon Press, New York, 1997.

10 Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. 2008. URL: `https://inria.hal.science/hal-03367725`.

11 William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957. `doi:10.2307/2963594`.

12 Giovanna D'Agostino and Giacomo Lenzi. Bisimulation quantifiers and uniform interpolation for guarded first order logic. *Theoretical Computer Science*, 563:75–85, 2015. `doi:10.1016/j.tcs.2014.08.015`.

13 Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order, Second Edition*. Cambridge University Press, 2002.

14 Hand van Ditmarsch, Jie Fan, Wiebe van der Hoek, and Petar Iliev. Some exponential lower bounds on formula-size in modal logic. In *Advances in Modal Logic*, 2014. URL: `https://api.semanticscholar.org/CorpusID:17911954`.

15 Giovanna D'Agostino and Marco Hollenberg. Logical questions concerning the μ-calculus. *Journal of Symbolic Logic*, 65:310–332, 2000.

16 Ryszard Engelking. *General topology*. PWN—Polish Scientific Publishers, Warsaw, 1977.

17 Dov M. Gabbay. Craig's interpolation theorem for modal logics. In Wilfrid Hodges, editor, *Conference in Mathematical Logic — London '70*, pages 111–127, Berlin, Heidelberg, 1972. Springer Berlin Heidelberg.

18 Dov M. Gabbay and Larisa Maksimova. *Interpolation and Definability: Modal and Intuitionistic Logic*, volume 41 of *Oxford Logic Guides*. Clarendon Press, Oxford, 2005.

19 Guido Gherardi, Paolo Maffezioli, and Eugenio Orlandelli. Interpolation in extensions of first-order logic. *Studia Logica*, 108:619–648, 2020. URL: https://doi.org/10.1007/s11225-019-09867-0.

20 Silvio Ghilardi and Marek Zawadowski. *Sheaves, Games and Model Completions: A Categorical Approach to Non Classical Propositional Logics*. Trends in Logic. Kluwer, Dordrecht, 2002.

21 Wilfrid Hodges. *Model Theory*. Encyclopedia of mathematics and its applications. Cambridge University Press, 1992. URL: https://books.google.nl/books?id=wo3RvQEACAAJ.

22 Eva Hoogland. *Definability and interpolation: model-theoretic investigations*. PhD thesis, University of Amsterdam, 2000.

23 David Janin and Igor Walukiewicz. Automata for the modal μ-calculus and related results. In *Mathematical foundations of computer science 1995 (Prague)*, volume 969 of *Lecture Notes in Computer Science*, pages 552–562, Berlin, 1995. Springer.

24 Peter Johnstone. *Stone spaces*, volume 3. Cambridge University Press, 1982.

25 Jean Christoph Jung and Jedrzej Kolodziejski. Modal separation of fixpoint formulae. In Olaf Beyersdorff, Michal Pilipczuk, Elaine Pimentel, and Kim Thang Nguyen, editors, *42nd International Symposium on Theoretical Aspects of Computer Science, STACS 2025, March 4-7, 2025, Jena, Germany*, volume 327 of *LIPIcs*, pages 55:1–55:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.STACS.2025.55.

26 Jean Christoph Jung, Andrea Mazzullo, and Frank Wolter. More on interpolants and explicit definitions for description logics with nominals and/or role inclusions. In *Proceedings of DL2022*, 2022.

27 Stephen Cole Kleene. *Introduction to Meta-Mathematics*. Elsevier North-Holland, 1952.

28 Marcus Kracht. *Tools and Techniques in Modal Logic*. North-Holland Publishing Co., Amsterdam, 1999.

29 Louwe Kuijer, Tony Tan, Frank Wolter, and Michael Zakharyaschev. Separation and definability in fragments of two-variable first-order logic with counting. In *Proceedings of LICS 2025*, 2025.

30 Tadeusz Litak. A continuum of incomplete intermediate logics. *Rep. Math. Logic*, 36:131–141, 2002.

31 Roger C. Lyndon. An interpolation theorem in the predicate calculus. *Pacific Journal of Mathematics*, 9(1):129 – 142, 1959. doi:pjm/1103039458.

32 Larisa Maksimova. Craig's theorem in superintuitionistic logics and amalgamable varieties of pseudo-boolean algebras. *Algebra and Logic*, 16(6):427–455, 1977.

33 Larisa Maksimova. Interpolation theorems in modal logics and amalgamable varieties of topological boolean algebras. *Algebra and Logic*, 18(5):348–370, 1979.

34 Larisa Maksimova. Amalgamation and interpolation in normal modal logics. *Studia Logica: An International Journal for Symbolic Logic*, 50(3/4):457–471, 1991. URL: http://www.jstor.org/stable/20015597.

35 Maarten Marx. *Algebraic Relativization and Arrow Logic*. PhD thesis, University of Amsterdam, 1995.

36 Maarten Marx and Yde Venema. *Multi-dimensional modal logic*, volume 4 of *Applied logic series*. Kluwer, 1997.

37 Andrew M. Pitts. On an interpretation of second order quantification in first order intuitionistic propositional logic. *The Journal of Symbolic Logic*, 57(1):33–52, 1992. URL: http://www.jstor.org/stable/2275175.

38 Francesca Poggiolesi. *Gentzen Calculi for Modal Propositional Logic*, volume 32 of *Trends in Logic*. Springer, 2011.

39 Vaughan R. Pratt. Models of program logics. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 115–122, 1979. doi:10.1109/SFCS.1979.24.

40 Benjamin Rossman. Formulas versus circuits for small distance connectivity. *SIAM Journal on Computing*, 47(5):1986–2028, 2018. doi:10.1137/15M1027310.

41 Gaisi Takeuti. *Proof Theory - 2nd edition*. Mineola, New York: Dover Publications, 1987.

42 Anne S. Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1996.

43 Moshe Y. Vardi and Pierre Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32(2):183–221, 1986. doi:10.1016/0022-0000(86)90026-7.

44 Yde Venema. Lectures on the modal µ-calculus. Technical report, ILLC, University of Amsterdam, 2004.

45 Yde Venema. Algebras and coalgebras. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of modal logic*, pages 331–426. Elsevier, 2007.

46 Albert Visser. Uniform interpolation and layered bisimulation. In *Gödel '96*, volume 6 of *Lecture Notes in Logic*, pages 139–164. Springer, Heidelberg, 1996. Brno, 1996.

Chapter 5

Interpolation in Proof Theory

Iris van der Giessen
University of Amsterdam, The Netherlands

Raheleh Jalali
University of Bath, UK

Roman Kuznets
Institute of Computer Science of the Czech Academy of Sciences, Czechia

Abstract

This chapter provides a comprehensive overview of proof-theoretic methods for establishing interpolation properties across a range of logics, including classical, intuitionistic, modal, and substructural logics. Central to the discussion are two foundational techniques: Maehara's method for Craig interpolation and Pitts' method for uniform interpolation. The chapter demonstrates how these methods lead to results on the existence of well-behaved proof systems in the contemporary framework of universal proof theory and how they provide a road map for constructing interpolation proofs using modern proof formalisms. The emphasis of the chapter is on constructive, modular, and syntax-driven techniques that illuminate deeper connections between interpolation properties and proof systems.

Contents

1 Introduction 150

2 Preliminaries 151

2.1 Propositional and Modal Logic 151

2.2 Sequent Calculi 153

3 Interpolation Using Sequent Calculi 155

3.1 Craig and Lyndon Interpolation via Maehara's Method 155

3.2 Restricted Cuts and Interpolation 162

3.3 Uniform Interpolation via Pitts' Method 163

4 Interpolation in Universal Proof Theory 168

4.1 Semi-analytic Rules and Interpolation 169

4.2 Interpolation and (Non-)Existence of Sequent Calculi 171



© Iris van der Giessen, Raheleh Jalali, and Roman Kuznets;
licensed under Creative Commons License CC BY 4.0

In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 5, pages 149–196. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.e

5 Interpolation Using Generalizations of Sequent Calculi 177
5.1 Labelled Sequents 178
5.2 Labelled Interpolation 180
5.3 Interpolation Results for Labelled, Hyper-, and Nested Sequents 188
6 Conclusion 190
Acknowledgments 191
References 191

1 Introduction

Proof-theoretic techniques towards interpolation originating from the 1950/60's [19, 64] are well-established and play an important role in the contemporary research on interpolation. The aim of this chapter is to provide a nice balance between (1) the *basics* of the praised proof-theoretic methods based on sequent calculi to prove interpolation properties (IPs), (2) the recent *applications* of such methods to the existence of proof systems within the emerging field of *universal proof theory*, and (3) the *development* of similar methods for more powerful contemporary proof systems such as labelled sequents.

The proof-theoretic method to prove the Craig interpolation property (CIP) is known as Maehara's method [64]. It provides an algorithm that, given a sequent proof of $\varphi \to \psi$, produces a Craig interpolant θ for it. This method is *constructive*: it explicitly constructs an interpolant, instead of merely proving its existence. Sometimes, the constructed interpolants preserve the polarity of propositional variables, which then shows the so-called Lyndon interpolation property (LIP). Maehara's method is used for a wide range of logics, including classical and intuitionistic first-order logic [81, 84, 86], (intuitionistic) modal logic [25, 36], linear logic [79], and substructural logic [75].

Pitts [76] has developed a syntactic method to prove the uniform interpolation property (UIP) for the intuitionistic propositional logic IPC (see Chapter 9, the main chapter on uniform interpolation). At the time, this was a rather surprising result as it shows that propositional quantification can be interpreted in IPC itself. Pitts uses a strongly terminating sequent calculus for IPC to prove the UIP. Pitts' idea was generalized to various propositional and (intuitionistic) modal logics [3, 11, 23, 44, 45] and formalized in Coq/Rocq [23, 24] enabling the automatic computation of uniform interpolants.

Maehara's method and Pitts' method are based on sequent calculi. With the development of more expressive sequent structures, such as labelled sequents, hypersequents, and nested sequents, related methods were designed to prove interpolation properties using such structures [38, 39, 56, 59, 63]. The new methods reveal that a more robust type of 'interpolants' is needed where it does not suffice to define them as formulas, but rather as structures that mimic the type of sequent structure at hand. Interestingly, such calculi seem more suitable to prove the LIP than Maehara's method via analytic sequent calculi.

The proof-theoretic approach to interpolation is not only useful in light of its constructive nature. The approach can also establish general results in proof theory itself. In the field of *universal proof theory*, structural conditions on sequent rules are identified that guarantee the CIP or UIP [3, 5, 7, 45]. Such sequent rules are called *semi-analytic* and capture the intuitive notion of 'nice' rules. Given the fact that interpolation properties are rare among intermediate logics, most such logics cannot have such 'nice' sequent calculi.

The advantages of the proof-theoretic method are the following (we already mentioned a couple): it provides a *constructive* proof of interpolation properties; the complexity of a Craig interpolant is often *linear* in the size of the proof; it is flexible across different classes of logics; and it finds applications in *proof complexity* and *universal proof theory*. There are also some disadvantages. Although the method is *constructive*, it does not immediately provide an algorithm for computing an interpolant based on a provable implication $\varphi \to \psi$ because it requires the construction of a sequent proof first. Also, Maehara's method is not complete meaning that it might be that given $\varphi \to \psi$ some of its interpolants cannot be constructed via this method [42].

Some applications and developments of the proof-theoretic approach to interpolation can be found in other chapters of this book. Chapter 6 on *proof complexity* discusses the feasible extraction of interpolants from proofs, which relates to the NP vs. coNP problem. Chapter 11 discusses new Maehara-type methods for *cyclic proofs*.

This chapter is structured as follows. We choose to illustrate the methods for the simple propositional and modal logics introduced in Section 2. The basics of Maehara's method and Pitts' method are explained in Section 3. The modularization of these methods and its impact on universal proof theory is discussed in Section 4. In Section 5 the proof-theoretic method and results for labelled sequents, hypersequents, and nested sequents is discussed. In Section 6 we provide some concluding remarks.

2 Preliminaries

2.1 Propositional and Modal Logic

Fix a countable set of *propositional variables* $\mathsf{Prop} = \{p, q, \dots\}$, also called *atoms*. For propositional logic we work with the language $\mathcal{L}_{\mathsf{p}} := \{\bot, \top, \vee, \wedge, \to\}$ and use symbols $\varphi, \psi, \dots$ to denote formulas generated from Prop and $\mathcal{L}_{\mathsf{p}}$ as usual. The set of all formulas is denoted by $\mathsf{Form}_{\mathcal{L}_{\mathsf{p}}}$. We use the standard abbreviation $\neg\varphi := \varphi \to \bot$. The base logics used in this chapter are *classical propositional logic* CPC and *intuitionistic propositional logic* IPC, where IPC can be defined as CPC without the law of the excluded middle $\varphi \vee \neg\varphi$. An *intermediate logic* is any logic L such that $\mathsf{IPC} \subseteq \mathsf{L} \subseteq \mathsf{CPC}$, where we consider a logic as a set of formulas closed under substitution and modus ponens. The interested reader is referred to [86, 87].

For modal logic we work with the language $\mathcal{L}_{\Box} := \mathcal{L}_{\mathsf{p}} \cup \{\Box\}$. As usual we define $\Diamond\varphi := \neg\Box\neg\varphi$. We define $\Box^n\varphi$ inductively as $\Box^0\varphi := \varphi$ and $\Box^{n+1}\varphi := \Box(\Box^n\varphi)$. Figure 1 presents standard modal axioms and rules, and the left column of Figure 3 presents common modal logics. Semantically, modal logics can be characterized by Kripke frames, also known

(k)	$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$	(5)	$\neg\Box\varphi \to \Box\neg\Box\varphi$	(MP)	$\dfrac{\varphi \to \psi \quad \varphi}{\psi}$
(t)	$\Box\varphi \to \varphi$	(d)	$\Box\varphi \to \Diamond\varphi$		
(4)	$\Box\varphi \to \Box\Box\varphi$	(wlöb)	$\Box(\Box\varphi \to \varphi) \to \Box\varphi$	(Nec)	$\dfrac{\varphi}{\Box\varphi}$

Figure 1 Modal axioms and the inference rules modus ponens (MP) and necessitation (Nec).

$\mathcal{M}, w \models p$	iff	$p \in V(w)$;
$\mathcal{M}, w \models \top$;		
$\mathcal{M}, w \not\models \bot$;		
$\mathcal{M}, w \models \varphi \to \psi$	iff	$\mathcal{M}, w \not\models \varphi$ or $\mathcal{M}, w \models \psi$;
$\mathcal{M}, w \models \varphi \vee \psi$	iff	$\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$;
$\mathcal{M}, w \models \varphi \wedge \psi$	iff	$\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$;
$\mathcal{M}, w \models \Box\varphi$	iff	for all v such that wRv, we have $\mathcal{M}, v \models \varphi$.

Figure 2 Definition of the modal satisfaction relation $\models$ between Kripke model $\mathcal{M} = (W, R, V)$, world $w \in W$, and formula φ.

$\mathsf{K} := \mathsf{CPC} + (\mathsf{k})$	class of all models
$\mathsf{T} := \mathsf{K} + (\mathsf{t})$	class of all reflexive models
$\mathsf{D} := \mathsf{K} + (\mathsf{d})$	class of all serial models
$\mathsf{K4} := \mathsf{K} + (4)$	class of all transitive models
$\mathsf{S4} := \mathsf{K} + (\mathsf{t}) + (4)$	class of all transitive and reflexive models
$\mathsf{S5} := \mathsf{K} + (\mathsf{t}) + (5)$	class of all total models
$\mathsf{GL} := \mathsf{K} + (\mathsf{wlöb})$	class of all transitive and conversely well-founded models

Figure 3 Axiomatizations of classical modal logics (each should be understood as closed under rules (MP) and (Nec)) with their soundness and completeness results.

as *linear transition systems*. A *Kripke model* is a tuple $\mathcal{M} = (W, R, V)$ with a non-empty set W of *worlds*, a binary relation $R \subseteq W \times W$ called the *accessibility relation*, and a function $V\colon W \to \mathcal{P}(\mathsf{Prop})$ called a *valuation*. Figure 2 provides the inductive definition of the *modal satisfaction relation* $\models$ between worlds in a model and formulas. We say that formula φ is *valid* in model $\mathcal{M} = (W, R, V)$ iff $\mathcal{M}, w \models \varphi$ for all worlds $w \in W$. Well-known soundness and completeness results are summarized in Figure 3, where φ is in a logic whenever it is valid in all respective models of the indicated classes. We refer to [14,18] for full introductions on modal logic.

In general, a language $\mathcal{L}$ is a set of symbols and we write $\mathsf{Form}_{\mathcal{L}}$ to denote the set of formulas generated from Prop and $\mathcal{L}$. For every language, we use symbols $\varphi, \psi, \ldots$ to denote

formulas. We write $\mathsf{Sub}(\varphi)$ for the set of all subformulas occurring in formula φ. For a logic L, we write $\vdash_{\mathsf{L}} \varphi$ to mean $\varphi \in \mathsf{L}$.

2.2 Sequent Calculi

Sequent calculi are rule-based descriptions of a logic and originate from Gentzen [32,33]. Here we introduce sequent calculi for propositional logic and modal logic [32, 81, 84, 86].

A *sequent* is an expression of the form $\Gamma \Rightarrow \Delta$ where Γ and Δ are finite multisets of formulas. Γ is called the *antecedent* of such a sequent, and Δ is called its *succedent* or *conclusion*. If Δ contains at most one formula, the sequent is called a *single-conclusion* sequent. We use capital letters $S, S', S_1, S_2, \ldots$ to refer to sequents. We use the notation Γ, φ for the multiset $\Gamma \cup \{\varphi\}$ and Γ, Δ for $\Gamma \cup \Delta$. We define $\mathsf{Sub}(\Gamma) := \bigcup_{\varphi \in \Gamma} \mathsf{Sub}(\varphi)$ for the set of all subformulas of formulas occurring in Γ and $\Box\Gamma := \{\Box\varphi \mid \varphi \in \Gamma\}$.

The intended meaning of a sequent $\Gamma \Rightarrow \Delta$ is the formula $\bigwedge_{\varphi \in \Gamma} \varphi \rightarrow \bigvee_{\psi \in \Delta} \psi$ also called the *formula interpretation* of the sequent. We adopt the convention that the empty conjunction is $\top$ and the empty disjunction is $\bot$.

An *inference rule* is an expression of the form on the left below where S and S_i's are sequents. Each inference rule gives rise to different *rule instances*, namely for every substitution σ, the expression on the right is an *instance* of the rule.

$$\frac{S_1 \quad S_2 \quad \ldots \quad S_n}{S} \qquad\qquad \frac{\sigma S_1 \quad \sigma S_2 \quad \ldots \quad \sigma S_n}{\sigma S} \tag{1}$$

This is a rather formal way of presenting sequent rules needed in Section 4.1, but for now we adopt a standard practice in proof theory of leaving the substitutions implicit and write rules as *rule schemas* (e.g., Figure 4), in which so-called *multiset variables* (resp. formula variables and atom variables) in the rules can represent any multiset (resp. formula and atom).

The upper sequent(s) of a rule are called the *premise(s)* and the lower sequent is called the *conclusion*. A rule with no premises is called an *axiom*, a rule with one premise is a *unary* rule, and a rule with two premises is a *binary* rule. When considering rule schemas as in Figure 4, the distinguished formula in the conclusion is called the *principal formula* and the multisets like Γ, Δ, etc. are called the *context*. The formulas in the premises that are not in the context are called the *active* formulas of the rule. Axiom (id) differs in that it has two principal formulas but no active ones. In the cut rule (Figure 4) we call φ the *cut formula*.

A *sequent calculus* is a finite set of inference rules and a *proof* in a given sequent calculus $\mathbf{G}$ is a finite rooted tree in which the nodes are sequents generated inductively from instances of the inference rules in $\mathbf{G}$. This means that in a proof all leaves are instances of axioms. The root of the tree is called the *endsequent* and is the sequent proved by the sequent proof. We write $\vdash_{\mathbf{G}} \Gamma \Rightarrow \Delta$ iff there is a proof in sequent calculus $\mathbf{G}$ with endsequent $\Gamma \Rightarrow \Delta$.

Sequent calculi are used to represent the provable formulas of a logic via soundness and completeness theorems. This is reflected in the following definition:

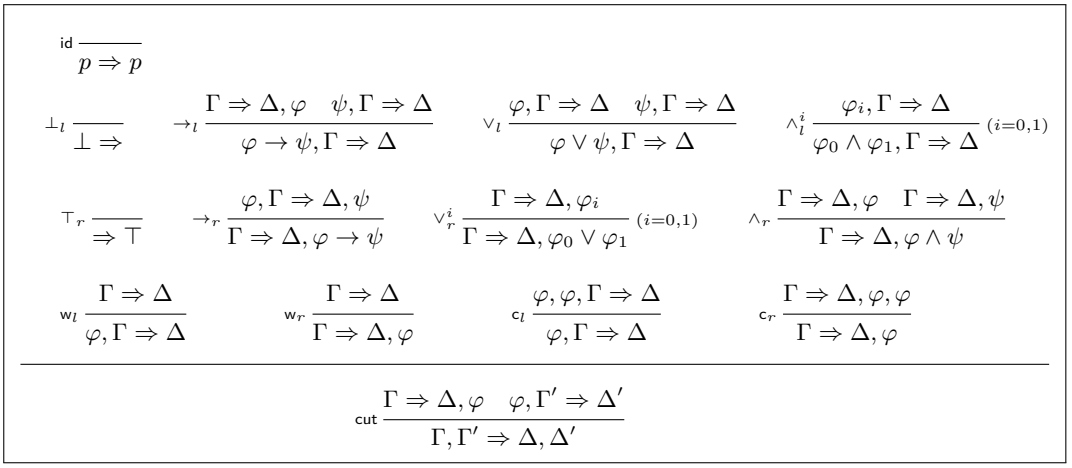


Figure 4 Top: Sequent calculus **LK**. Subscripts l and r stand for *left* and *right* rule respectively. The *structural rules* are *weakening* (w_l) and (w_r) and *contraction* (c_l) and (c_r). Sequent calculus **LJ** is obtained by restricting these rules to single-conclusion sequents, i.e., $\Delta = \varnothing$ in the right rules, and in the left rules Δ contains at most one formula, with the extra condition for ($\rightarrow_l$) that Δ is not present in the left premise, and the right contraction rule (c_r) is omitted. **Bottom:** The cut rule. For single-conclusion calculi we require that $\Delta = \varnothing$ and that Δ' contains at most one formula.

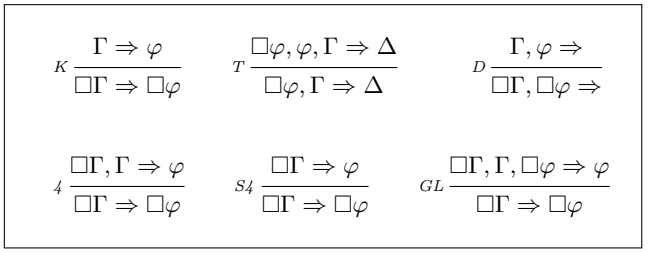


Figure 5 Modal sequent rules.

$$\begin{aligned}
\mathbf{G.K} &:= \mathbf{LK} + (K)\\
\mathbf{G.T} &:= \mathbf{G.K} + (T)\\
\mathbf{G.D} &:= \mathbf{G.K} + (D)\\
\mathbf{G.K4} &:= \mathbf{LK} + (4)\\
\mathbf{G.S4} &:= \mathbf{LK} + (S4) + (T)\\
\mathbf{G.GL} &:= \mathbf{LK} + (GL)
\end{aligned}$$

Figure 6 Modal sequent calculi.

▶ **Definition 1.** *Let* $\mathbf{G}$ *be a sequent calculus and let* L *be a logic over the same language* $\mathcal{L}$*. We say that* $\mathbf{G}$ *is a* sequent calculus for L*, or that* L *is* the logic of $\mathbf{G}$*, iff for every sequent* $\Gamma \Rightarrow \Delta$*, we have* $\vdash_{\mathbf{G}} \Gamma \Rightarrow \Delta$ *iff* $\vdash_{\mathsf{L}} \bigwedge \Gamma \rightarrow \bigvee \Delta$*.*

Figure 4 defines the well-known sequent calculus **LK** for classical logic CPC and the single-conclusion calculus **LJ** for intuitionistic logic IPC. We define the calculi without the cut rule (Figure 4). Figure 5 presents some well-known modal rules, and Figure 6 defines calculi for classical modal logics. We denote sequent calculi in bold font, using the name of the logic, sometimes prefixed with **G**., to identify the system. Thus, in Figure 5 we mean that **G.K** is a sequent calculus for modal logic K, etc.

Important to note is that the cut rule is admissible in the presented calculi. It means that the rule can safely be added to the calculi without deriving new sequents. Formally, a rule with premises $S_1, \ldots, S_n$ and conclusion S is *admissible* in a sequent calculus $\mathbf{G}$ iff for any substitution σ, if $\vdash_{\mathbf{G}} \sigma S_i$ for all i, then $\vdash_{\mathbf{G}} \sigma S$.

▶ **Theorem 2** (Cut-admissibility). *Rule* (cut) *is admissible in all calculi from Figures 4 and 6.*

Cut-admissibility theorems feature prominently in proof theory. Indeed, the cut rule allows for easy completeness results, but the possibility to eliminate cut has far reaching applications such as consistency and decidability results. Cut-free calculi are used in Maehara's proof-theoretic method for the CIP [64]. We will see in Section 3.2 that some forms of cut can be adopted in Maehara's method, but may create problems for the LIP.

▶ Remark 3 (Design choices of sequent calculi). Multiple choices can be made when designing a sequent calculus. Two common choices are: the type of sequents used (e.g., lists, multisets, or sets), and the choice between explicit or implicit structural rules. Generally, these choices make no difference for Maehara's method to prove the CIP, only the cut rule may present a problem. Pitts' method for the UIP is more sensitive to such design choices where contraction rules create a problem (Section 3.3). ⌟

▶ Remark 4 (Sequent calculi vs tableaux). The intended reading of the sequent rules is from top to bottom: assuming the validity of the premises, the conclusion is valid. Dual to sequent calculi are tableau calculi [25]. These can be viewed as sequent calculi 'upside down', where the unsatisfiability of the upper formulas is preserved for the lower formulas. Maehara style methods also exist for tableau calculi [25]. See Chapter 1 of this book for more details. ⌟

3 Interpolation Using Sequent Calculi

In this section we discuss the basics of the proof-theoretic methods of interpolation properties based on propositional and modal logics.

3.1 Craig and Lyndon Interpolation via Maehara's Method

There are different types of Craig interpolation depending on what consequence is under consideration (see other chapters in this book). Maehara's method is very suitable for the *local* Craig interpolation property where we interpolate the local consequence relation represented by $\rightarrow$. It is also suitable for the local Lyndon interpolation property, which takes into account the polarity of propositional variables.

▶ **Definition 5** (Signature). *The* signature *of a formula φ, denoted $\mathsf{Var}(\varphi)$, is the set of propositional variables occurring in φ.*

For the propositional language $\mathcal{L}_{\mathsf{p}}$ and modal language $\mathcal{L}_{\Box}$ we inductively define the sets $\mathsf{Var}^+(\varphi)$ and $\mathsf{Var}^-(\varphi)$ of propositional variables occurring in φ with positive *and* negative polarity *respectively:* $\mathsf{Var}^+(p) := \{p\}$ *and* $\mathsf{Var}^-(p) := \varnothing$ *for* $p \in \mathsf{Prop}$*;* $\mathsf{Var}^+(\bot) = \mathsf{Var}^+(\top) =$

$\mathsf{Var}^-(\top) = \mathsf{Var}^-(\bot) := \varnothing$*; connectives* $\vee$ *and* $\wedge$ *preserve polarity; connective* $\to$ *flips polarity in the antecedent and preserves polarity in the succedent; modality* $\Box$ *preserves polarity.*

For a multiset Γ*, we define* $\mathsf{Var}(\Gamma) := \bigcup_{\varphi\in\Gamma}\mathsf{Var}(\varphi)$*, and analogously for* $\mathsf{Var}^+(\Gamma)$ *and* $\mathsf{Var}^-(\Gamma)$*. For a sequent* $\Gamma \Rightarrow \Delta$*, we define* $\mathsf{Var}^+(\Gamma \Rightarrow \Delta) := \mathsf{Var}^-(\Gamma) \cup \mathsf{Var}^+(\Delta)$ *and* $\mathsf{Var}^-(\Gamma \Rightarrow \Delta) := \mathsf{Var}^+(\Gamma) \cup \mathsf{Var}^-(\Delta)$*, where polarity flips in the antecedent similarly to* $\to$*.*

It is clear from the definition that $\mathsf{Var}(\varphi) = \mathsf{Var}^+(\varphi) \cup \mathsf{Var}^-(\varphi)$ for any formula φ from $\mathcal{L}_\mathsf{p}$ or $\mathcal{L}_\Box$, and similarly for multisets and sequents.

▶ **Example 6.** $\mathsf{Var}^+((\Box p \to q) \to r) = \{p, r\}$ and $\mathsf{Var}^-((\Box p \to q) \to r) = \{q\}$. Similarly, $\mathsf{Var}^+((\Box p \to q) \to q) = \{p, q\}$ and $\mathsf{Var}^-((\Box p \to q) \to q) = \{q\}$. ⌟

▶ Remark 7. We define the signature to consist of only propositional variables. One can also consider modalities as part of the signature. In the setting of first-order logic, the signature consists of relational and function symbols occurring in a first-order formula φ. ⌟

▶ **Definition 8** (Craig interpolation)**.** *A logic* L *over a language* $\mathcal{L}$ *has the* Craig Interpolation Property (CIP)*, iff whenever* $\vdash_\mathsf{L} \varphi \to \psi$*, there is a formula* $\theta \in \mathsf{Form}_\mathcal{L}$ *such that:*
1. $\mathsf{Var}(\theta) \subseteq \mathsf{Var}(\varphi) \cap \mathsf{Var}(\psi)$*,*
2. $\vdash_\mathsf{L} \varphi \to \theta$ *and* $\vdash_\mathsf{L} \theta \to \psi$*.*

Such formula θ *is called a* (Craig) interpolant *of* $\varphi \to \psi$ *in* L*.*

▶ **Definition 9** (Lyndon interpolation)**.** *The* Lyndon Interpolation Property (LIP) *for logics over language* $\mathcal{L}_\mathsf{p}$ *or* $\mathcal{L}_\Box$ *is defined analogously to the CIP replacing condition 1 in Definition 8 with the following condition:*
1'. $\mathsf{Var}^+(\theta) \subseteq \mathsf{Var}^+(\varphi) \cap \mathsf{Var}^+(\psi)$ *and* $\mathsf{Var}^-(\theta) \subseteq \mathsf{Var}^-(\varphi) \cap \mathsf{Var}^-(\psi)$*.*
In this case θ *is called a* Lyndon interpolant *of* $\varphi \to \psi$ *in* L*.*

It is clear from the definition that the LIP implies the CIP.

▶ **Example 10** (Non-unique interpolants)**.** Given a provable implication $\varphi \to \psi$, there may exist multiple provably non-equivalent Craig/Lyndon interpolants. For example in CPC, let $\varphi = (p \wedge q) \vee (\neg r \wedge s)$ and $\psi = t \vee p \vee q \vee \neg r$. We have $\vdash_\mathsf{CPC} \varphi \to \psi$. There are multiple provably non-equivalent interpolants for it, namely $(p \wedge q) \vee \neg r$, $p \vee \neg r$, $q \vee \neg r$, and $p \vee q \vee \neg r$. ⌟

Now we introduce Maehara's method and show how sequent calculi can be used to prove the CIP and LIP. *Maehara's method* is commonly described as the proof-theoretic method of proving the CIP by an induction on a sequent proof (or proof in another suitable proof system). This is a helpful intuition. However, the method is more refined and interesting because of its more complex underlying process. The idea is to formulate a sequent-based interpolation property that is at least as strong as the CIP and that relies on a calculus with richer sequents called split sequents.

▶ **Definition 11** (Split sequent)**.** *The expression of the form* $\Gamma; \Gamma' \Rightarrow \Delta; \Delta'$ *is called a* split sequent*. Multisets* Γ *and* Δ *belong to the* left split*, and* Γ' *and* Δ' *to the* right split*.*

One can also think of sequent $\Gamma \Rightarrow \Delta$ being the left split and sequent $\Gamma' \Rightarrow \Delta'$ being the right split of split sequent $\Gamma; \Gamma' \Rightarrow \Delta; \Delta'$.

▶ **Definition 12** (Sequent Craig interpolation). *Let* $\mathbf{G}$ *be a sequent calculus over a language* $\mathcal{L}$. *We say that* $\mathbf{G}$ *has the* Sequent Craig Interpolation Property (SCIP) *iff whenever we have* $\vdash_{\mathbf{G}} \Gamma, \Gamma' \Rightarrow \Delta, \Delta'$, *there exists a formula* $\theta \in \mathsf{Form}_{\mathcal{L}}$ *such that*

1. $\mathsf{Var}(\theta) \subseteq \mathsf{Var}(\Gamma, \Delta) \cap \mathsf{Var}(\Gamma', \Delta')$, *and*
2. $\vdash_{\mathbf{G}} \Gamma \Rightarrow \Delta, \theta$ *and* $\vdash_{\mathbf{G}} \theta, \Gamma' \Rightarrow \Delta'$.

If above properties hold we write $\vdash_{\mathbf{G}} \Gamma; \Gamma' \overset{\theta}{\Longrightarrow} \Delta; \Delta'$ *and we call* θ *a* (Craig) interpolant *for split sequent* $\Gamma; \Gamma' \Rightarrow \Delta; \Delta'$ *in* $\mathbf{G}$. *When dealing with a single-conclusion calculus we require* $\Delta = \varnothing$ *and* Δ' *to contain at most one formula.*

The intuition of the SCIP is that formula θ interpolates between the left and right side of the split, rather than between the antecedent and succedent of the sequent. The splits are essential as explained after Theorem 14.

For sequent Lyndon interpolation we take into account the polarity of the variables:

▶ **Definition 13** (Sequent Lyndon interpolation). *The* Sequent Lyndon Interpolation Property (SLIP) *for logics over language* $\mathcal{L}_{\mathsf{p}}$ *or* $\mathcal{L}_{\Box}$ *is defined analogously to the SCIP replacing condition 1 in Definition 12 with:*

1'. $\mathsf{Var}^{+}(\theta) \subseteq \mathsf{Var}^{-}(\Gamma \Rightarrow \Delta) \cap \mathsf{Var}^{+}(\Gamma' \Rightarrow \Delta')$ *and* $\mathsf{Var}^{-}(\theta) \subseteq \mathsf{Var}^{+}(\Gamma \Rightarrow \Delta) \cap \mathsf{Var}^{-}(\Gamma' \Rightarrow \Delta')$.

In this case θ *is called a* Lyndon interpolant *for split sequent* $\Gamma; \Gamma' \Rightarrow \Delta; \Delta'$ *in* $\mathbf{G}$.

The interpolation property of a sequent calculus implies interpolation of its logic:

▶ **Theorem 14.** *If a sequent calculus* $\mathbf{G}$ *has the SCIP/SLIP, then its logic* L *has the CIP/LIP.*

Proof sketch. Assume that $\mathbf{G}$ has the SCIP/SLIP and let $\vdash_{\mathsf{L}} \varphi \to \psi$. By Definition 1, $\vdash_{\mathbf{G}} \varphi \Rightarrow \psi$. For any Craig/Lyndon interpolant θ for the split $\varphi; \Rightarrow ; \psi$ of this sequent in $\mathbf{G}$, one can prove that θ is a Craig/Lyndon interpolant of $\varphi \to \psi$ in L. ◀

The idea of Maehara's method is to present the sequent rules using split sequents, where each sequent rule has multiple counterparts in the split sequent calculus depending on whether the principal formula is on the left or on the right side of the split. Figure 7 presents the split system of $\mathbf{LK}$. Given a split sequent proof of $\vdash_{\mathbf{G}} \varphi; \Rightarrow ; \psi$, one assigns Craig interpolants to the leaves and inductively defines the interpolants when going down the tree yielding an interpolant for $\vdash_{\mathbf{G}} \varphi; \Rightarrow ; \psi$. The splits ensure the variable property 1 of Definition 12. Indeed, bottom-up, the splits keep track of those formulas originating from φ on the left and those originating from ψ on the right side of the split. For example, in the following instance of rule $R(\to_r)$, formula ψ_1 should be stored on the right side of the split in the antecedent of the premise:

$$R(\to_r)\ \frac{\varphi; \psi_1 \Rightarrow ; \psi_2}{\varphi; \Rightarrow ; \psi_1 \to \psi_2}$$

Split axioms:

$$RR(\mathsf{id})\ \frac{}{;p \overset{\top}{\Longrightarrow} ;p} \qquad LR(\mathsf{id})\ \frac{}{p; \overset{p}{\Rightarrow} ;p} \qquad RL(\mathsf{id})\ \frac{}{;p \overset{\neg p}{\Longrightarrow} p;} \qquad LL(\mathsf{id})\ \frac{}{p; \overset{\bot}{\Longrightarrow} p;}$$

$$L(\bot_l)\ \frac{}{\bot; \overset{\bot}{\Longrightarrow} ;} \qquad R(\bot_l)\ \frac{}{;\bot \overset{\top}{\Longrightarrow} ;} \qquad L(\top_l)\ \frac{}{; \overset{\bot}{\Longrightarrow} \top;} \qquad R(\top_r)\ \frac{}{; \overset{\top}{\Longrightarrow} ;\top}$$

Some unary split rules:

$$R(\wedge_l^i)\ \frac{\Gamma;\Gamma', \varphi_i \overset{\theta}{\Rightarrow} \Delta;\Delta'}{\Gamma;\Gamma', \varphi_0 \wedge \varphi_1 \overset{\theta}{\Rightarrow} \Delta;\Delta'} \qquad R(\to_r)\ \frac{\Gamma;\Gamma', \varphi \overset{\theta}{\Rightarrow} \Delta;\Delta', \psi}{\Gamma;\Gamma' \overset{\theta}{\Rightarrow} \Delta;\Delta', \varphi \to \psi} \qquad L(\mathsf{c}_r)\ \frac{\Gamma;\Gamma' \overset{\theta}{\Rightarrow} \Delta, \varphi, \varphi;\Delta'}{\Gamma;\Gamma' \overset{\theta}{\Rightarrow} \Delta, \varphi;\Delta'}$$

Binary split rules:

$$L(\vee_l)\ \frac{\varphi, \Gamma;\Gamma' \overset{\theta_1}{\Longrightarrow} \Delta;\Delta' \quad \psi, \Gamma;\Gamma' \overset{\theta_2}{\Longrightarrow} \Delta;\Delta'}{\varphi \vee \psi, \Gamma;\Gamma' \overset{\theta_1 \vee \theta_2}{\Longrightarrow} \Delta;\Delta'} \qquad R(\vee_l)\ \frac{\Gamma;\Gamma', \varphi \overset{\theta_1}{\Longrightarrow} \Delta;\Delta' \quad \Gamma;\Gamma', \psi \overset{\theta_2}{\Longrightarrow} \Delta;\Delta'}{\Gamma;\Gamma', \varphi \vee \psi \overset{\theta_1 \wedge \theta_2}{\Longrightarrow} \Delta;\Delta'}$$

$$L(\wedge_r)\ \frac{\Gamma;\Gamma' \overset{\theta_1}{\Longrightarrow} \varphi, \Delta;\Delta' \quad \Gamma;\Gamma' \overset{\theta_2}{\Longrightarrow} \psi, \Delta;\Delta'}{\Gamma;\Gamma' \overset{\theta_1 \vee \theta_2}{\Longrightarrow} \varphi \wedge \psi, \Delta;\Delta'} \qquad R(\wedge_r)\ \frac{\Gamma;\Gamma' \overset{\theta_1}{\Longrightarrow} \Delta;\Delta', \varphi \quad \Gamma;\Gamma' \overset{\theta_2}{\Longrightarrow} \Delta;\Delta', \psi}{\Gamma;\Gamma' \overset{\theta_1 \wedge \theta_2}{\Longrightarrow} \Delta;\Delta', \varphi \wedge \psi}$$

$$L(\to_l)\ \frac{\Gamma;\Gamma' \overset{\theta_1}{\Longrightarrow} \Delta, \varphi;\Delta' \quad \psi, \Gamma;\Gamma' \overset{\theta_2}{\Longrightarrow} \Delta;\Delta'}{\varphi \to \psi, \Gamma;\Gamma' \overset{\theta_1 \vee \theta_2}{\Longrightarrow} \Delta;\Delta'} \qquad R(\to_l)\ \frac{\Gamma;\Gamma' \overset{\theta_1}{\Longrightarrow} \Delta;\Delta', \varphi \quad \Gamma;\Gamma', \psi \overset{\theta_2}{\Longrightarrow} \Delta;\Delta'}{\Gamma;\Gamma', \varphi \to \psi \overset{\theta_1 \wedge \theta_2}{\Longrightarrow} \Delta;\Delta'}$$

Figure 7 Craig interpolant construction with split sequent calculus for **LK**. Capital letters L and R denote whether the principal formula is on the left or right side of the split respectively. The figure presents only three of the unary split rules, because all other unary split rules have the same shape in which the active formulas in the premise stay on the same side of the split as the principal formula of the conclusion and the interpolant of the conclusion is preserved from the premise.

In the split sequent calculus of **LK** we see that active formulas are always on the same side as the principal formula, which is not necessarily the case for sequent calculi for other logics, see Remark 21.

▶ Remark 15 (Monotone connectives). When working with a language with only *monotone connectives*, Maehara's split is redundant as the left and right sides can be taken as the left and right sides of the sequent arrow $\Rightarrow$ (cf. symmetric sequents in [25]). ⌟

We present Maehara's method for classical propositional logic:

▶ **Theorem 16. LK** *has the SCIP and SLIP, and hence* CPC *enjoys the CIP and LIP.*

Proof. Suppose $\vdash_{\mathbf{LK}} \Gamma;\Gamma' \Rightarrow \Delta;\Delta'$ and let π be its split sequent proof with rules from Figure 7. We will construct an interpolant θ by induction on the structure of π. For both the base step and the induction step, there are several subcases based on the lowest split sequent rule of π. The interpolant construction for each case is summarized in Figure 7. Here we will check two cases. As most textbooks only consider Craig interpolants, here we show that this construction also satisfies the stronger variable condition for Lyndon interpolants.

- $RR(\mathsf{id})$: Suppose
$$\pi \;=\; \dfrac{}{;p \Rightarrow ;p}\,RR(\mathsf{id})$$
We claim that $\theta = \top$. Indeed, the variable condition $1'$ from Definition 13 holds because $\mathsf{Var}(\top) = \varnothing$. Also, $\vdash_{\mathbf{LK}} \Rightarrow \top$ by $(\top_r)$ and $\vdash_{\mathbf{LK}} \top, p \Rightarrow p$ by (id) and (w_l) as desired.
- $L(\to_l)$: Suppose that the bottommost rule is an instance of $(\to_l)$ with the principal formula on the left side of the split. For the premises the appropriate splits are chosen.
$$\pi \;=\; \dfrac{\overset{\pi_1}{\triangledown} \qquad \overset{\pi_2}{\triangledown}}{\Gamma;\Gamma' \Rightarrow \Delta,\varphi;\Delta' \qquad \psi,\Gamma;\Gamma' \Rightarrow \Delta;\Delta'}\,L(\to_l)$$
$$\varphi \to \psi,\Gamma;\Gamma' \Rightarrow \Delta;\Delta'$$
By the induction hypothesis (IH) on π_1 and π_2, there are Lyndon interpolants θ_1 and θ_2 such that $\Gamma;\Gamma' \overset{\theta_1}{\Longrightarrow} \Delta,\varphi;\Delta'$ and $\psi,\Gamma;\Gamma' \overset{\theta_2}{\Longrightarrow} \Delta;\Delta'$. We prove that $\theta = \theta_1 \vee \theta_2$ is the desired interpolant. Out of the two variable conditions from Definition 13, we show how to check the first (the other is symmetric):
$$\begin{aligned}
&\mathsf{Var}^+(\theta_1 \vee \theta_2) = \mathsf{Var}^+(\theta_1) \cup \mathsf{Var}^+(\theta_2)\\
&\quad\subseteq \Big(\mathsf{Var}^-(\Gamma \Rightarrow \Delta,\varphi) \cap \mathsf{Var}^+(\Gamma' \Rightarrow \Delta')\Big) \cup \Big(\mathsf{Var}^-(\psi,\Gamma \Rightarrow \Delta) \cap \mathsf{Var}^+(\Gamma' \Rightarrow \Delta')\Big)\ \text{(IH)}\\
&\quad= \Big(\mathsf{Var}^-(\Gamma \Rightarrow \Delta,\varphi) \cup \mathsf{Var}^-(\psi,\Gamma \Rightarrow \Delta)\Big) \cap \mathsf{Var}^+(\Gamma' \Rightarrow \Delta')\\
&\quad= \Big(\mathsf{Var}^+(\psi,\Gamma) \cup \mathsf{Var}^-(\Delta,\varphi)\Big) \cap \mathsf{Var}^+(\Gamma' \Rightarrow \Delta')\\
&\quad= \Big(\mathsf{Var}^+(\varphi \to \psi,\Gamma) \cup \mathsf{Var}^-(\Delta)\Big) \cap \mathsf{Var}^+(\Gamma' \Rightarrow \Delta')\\
&\quad= \mathsf{Var}^-(\varphi \to \psi,\Gamma \Rightarrow \Delta) \cap \mathsf{Var}^+(\Gamma' \Rightarrow \Delta').
\end{aligned}$$
It remains to check that $\vdash_{\mathbf{LK}} \varphi \to \psi,\Gamma \Rightarrow \Delta,\theta_1 \vee \theta_2$ and $\vdash_{\mathbf{LK}} \theta_1 \vee \theta_2,\Gamma' \Rightarrow \Delta'$. Indeed:
$$\dfrac{\dfrac{\dfrac{\overset{IH}{\triangledown}}{\Gamma \Rightarrow \Delta,\theta_1,\varphi}}{\Gamma \Rightarrow \Delta,\theta_1,\theta_2,\varphi}\,\mathsf{w}_r \qquad \dfrac{\overset{IH}{\triangledown}}{\psi,\Gamma \Rightarrow \Delta,\theta_2}\,\mathsf{w}_r}{\varphi \to \psi,\Gamma \Rightarrow \Delta,\theta_1,\theta_2}\,{\to_l}$$
$$\dfrac{\varphi \to \psi,\Gamma \Rightarrow \Delta,\theta_1,\theta_2}{\varphi \to \psi,\Gamma \Rightarrow \Delta,\theta_1 \vee \theta_2}\,\mathsf{c}_r,\vee_r^0,\vee_r^1$$
$$\dfrac{\overset{IH}{\triangledown} \qquad \overset{IH}{\triangledown}}{\theta_1,\Gamma' \Rightarrow \Delta' \qquad \theta_2,\Gamma' \Rightarrow \Delta'}\,\vee_l$$
$$\theta_1 \vee \theta_2,\Gamma' \Rightarrow \Delta'$$
We leave the other cases to the reader. ◀

▶ Remark 17 (Computation and complexity). Maehara's method is *constructive*: it provides an algorithm to compute interpolants. The complexity of an interpolant of $\varphi \to \psi$ is *linear* in the number of sequents occurring in the proof-tree of $\varphi \Rightarrow \psi$. Different proofs of a sequent may yield different interpolants. ⌟

▶ Remark 18 (Incompleteness). Interestingly, it has been shown that Maehara's method is *incomplete* in the sense that there are implications $\varphi \to \psi$ for which there are interpolants that cannot be computed by Maehara's method [42] (see also the next example). ⌟

▶ **Example 19.** Recall Example 10 with $\varphi = (p \wedge q) \vee (\neg r \wedge s)$ and $\psi = t \vee p \vee q \vee \neg r$. We work in the split calculus **LK** (Figure 7). Craig interpolant $p \vee \neg r$ is derived as follows:

$$\dfrac{\dfrac{\dfrac{}{p;\overset{p}{\Longrightarrow};p}\,LR(\mathsf{id})}{p,q;\overset{p}{\Longrightarrow};t,p,q,\neg r}\,L(\mathsf{w}_l),R(\mathsf{w}_r) \qquad \dfrac{\dfrac{\dfrac{}{;r\overset{\neg r}{\Longrightarrow}r;}\,RL(\mathsf{id})}{\overset{\neg r}{\Longrightarrow}r;\neg r}\,R(\to_r),R(\mathsf{w}_r) \qquad \dfrac{}{\bot;\overset{\bot}{\Longrightarrow};}\,L(\bot_l)}{\neg r;\overset{\neg r\vee\bot(=\neg r)}{\Longrightarrow};\neg r}\,L(\to_l)}{\neg r,s;\overset{\neg r}{\Longrightarrow};t,p,q,\neg r}\,L(\mathsf{w}_l),R(\mathsf{w}_r)}{p\wedge q;\overset{p}{\Longrightarrow};t,p,q,\neg r}\,L(\wedge_l) \qquad \dfrac{\dfrac{\dfrac{}{\bot;\overset{\bot}{\Longrightarrow};}\,L(\bot_l)}{\bot;\overset{\bot}{\Longrightarrow};\neg r}\,R(\mathsf{w}_r)}{\neg r\wedge s;\overset{\neg r}{\Longrightarrow};t,p,q,\neg r}\,L(\wedge_l)}{(p\wedge q)\vee(\neg r\wedge s);\overset{p\vee\neg r}{\Longrightarrow};t\vee p\vee q\vee\neg r}\,L(\vee_l),R(\vee_r)$$

A similar split sequent proof with the left branch starting with axiom $q;\overset{q}{\Longrightarrow};q$ yields interpolant $q \vee \neg r$. Interpolant $(p \vee q) \vee \neg r$ can be obtained by Maehara's method using contraction rules. However, interpolant $(p \wedge q) \vee \neg r$ cannot be obtained by Maehara's method in **LK** (see [42]). ⌟

▶ Remark 20 (Other types of sequent calculi). Maehara's method is very flexible and has been used for different types of sequent calculi (recall Remark 3). We worked with two-sided (multiset) sequent calculus **LK**. It equally works for sequents based on lists or sets, and when working with implicit weakening and contraction rules [86]. The method also works for *one-sided sequent* calculi based on formulas in negation normal form, where a one-sided split sequent has the form $\Delta_l; \Delta_r$ with left split Δ_l and right split Δ_r. Recall from Remark 4 that sequent calculi can be viewed as the dual of tableau calculi. For that reason, Maehara's method can be adjusted to tableaux (see Chapter 1.10 of this book). ⌟

▶ Remark 21 (Single-conclusion calculi). Maehara's method also applies to single-conclusion sequent calculi such as **LJ**. The main difference is that sequents have only one formula in the succedent, so there is no need to split the succedent; equivalently, one may view the succedent as always belonging to the "right" side of the split. For **LJ**, the construction of interpolants

$$L(K)\ \frac{\Gamma;\Gamma' \overset{\theta}{\Rightarrow} \varphi;}{\Box\Gamma;\Box\Gamma' \overset{\Diamond\theta}{\Longrightarrow} \Box\varphi;} \qquad R(K)\ \frac{\Gamma;\Gamma' \overset{\theta}{\Rightarrow};\varphi}{\Box\Gamma;\Box\Gamma' \overset{\Box\theta}{\Longrightarrow};\Box\varphi} \qquad L(D)\ \frac{\varphi,\Gamma;\Gamma' \overset{\theta}{\Rightarrow};}{\Box\varphi,\Box\Gamma;\Box\Gamma' \overset{\Box\theta}{\Longrightarrow};} \qquad R(D)\ \frac{\Gamma;\Gamma',\varphi \overset{\theta}{\Rightarrow};}{\Box\Gamma;\Box\Gamma',\Box\varphi \overset{\Box\theta}{\Longrightarrow};}$$

$$L(GL)\ \frac{\Box\Gamma,\Gamma,\Box\varphi;\Box\Gamma',\Gamma' \overset{\theta}{\Rightarrow} \varphi;}{\Box\Gamma;\Box\Gamma' \overset{\Diamond\theta}{\Longrightarrow} \Box\varphi;} \qquad R(GL)\ \frac{\Box\Gamma,\Gamma;\Box\Gamma',\Gamma',\Box\varphi \overset{\theta}{\Rightarrow};\varphi}{\Box\Gamma;\Box\Gamma' \overset{\Box\theta}{\Longrightarrow};\Box\varphi} \qquad L(T)\ \frac{\varphi,\Box\varphi,\Gamma;\Gamma' \overset{\theta}{\Rightarrow} \Delta;\Delta'}{\Box\varphi,\Gamma;\Gamma' \overset{\theta}{\Rightarrow} \Delta;\Delta'}$$

Figure 8 Interpolant transformations for modal rules. When the principal formula is on the left side of the split in rules (K) and (GL), it is a '$\Diamond$-transformation', and when it is on the right it is a '$\Box$-transformation'. It similarly works for rules (4) and $(S4)$ from Figure 5. For rule (T), the same interpolant suffices for both splits. Both transformations for (D) are '$\Box$-transformation' because the only role φ plays there is to ensure that the antecedent is not empty. See Examples 70 and 80.

proceeds much as in the **LK** case described in Figure 7, except that rules such as $RL(\mathsf{id})$ and $R(\wedge_r)$ are omitted, and the split rule $L(\rightarrow_l)$ is adapted as follows:

$$L(\rightarrow_l)\ \frac{\Gamma';\Gamma \overset{\theta_1}{\Longrightarrow} \varphi \quad \psi,\Gamma;\Gamma' \overset{\theta_2}{\Longrightarrow} \Delta'}{\varphi \rightarrow \psi,\Gamma;\Gamma' \overset{\theta_1 \rightarrow \theta_2}{\Longrightarrow} \Delta'}$$

where the formula $\varphi \rightarrow \psi$ is on the left side of the split in the conclusion, but φ is forced to be on the 'right' side of the split in the left premise. This forces us to swap the sides of Γ' and Γ in this premise, to make sure that Γ and φ remain on the same side of the split. ⌟

In light of the previous remark we have the following theorem.

▶ **Theorem 22.** *Logic* IPC *has the CIP and LIP.*

For modal logics, we provide the interpolant transformations for the sequent calculi in Figure 6, and show that logics K, T, D, K4, S4, and GL have the CIP. In fact, all of these except for GL admit the LIP. When considering split rules $L(GL)$ and $R(GL)$ from Figure 8, φ and both $\Box\varphi$'s are placed on the same side of the split. Thus, the Craig condition in Definition 12(1) is satisfied, but the stronger Lyndon one in Definition 13(1′) cannot be guaranteed: $\Box\varphi$ in the premise antecedent yields the opposite polarities from $\Box\varphi$ in the conclusion succedent. Consequently, sequent calculus **G.GL** can be used to prove the CIP but not the LIP, despite the fact that logic GL does have the LIP [83]. An existing proof-theoretic proof of the LIP for GL [82] uses cyclic proofs, which are discussed in Chapter 11. More generally, the existence of a cut-free sequent calculus is no guarantee that the logic has the CIP, nor the logic having the CIP is a guarantee that a cut-free sequent calculus can be used to prove it.

Although Maehara's method is flexible across many types of logics, it does not mean that many logics of each type enjoy the CIP. In fact, interpolation is rare among, e.g., intermediate and modal logics [29, 35], which has crucial impact on proof theory (see Section 4.2).

$$\textsf{an-cut}\ \frac{\Gamma \Rightarrow \Delta, \varphi \quad \varphi, \Gamma' \Rightarrow \Delta'}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\ \varphi \in \mathsf{Sub}(\Gamma, \Gamma', \Delta, \Delta')$$

$$\Box\textsf{-semi-cut}\ \frac{\Gamma \Rightarrow \Delta, \Box^n \psi \quad \Box^n \psi, \Gamma' \Rightarrow \Delta'}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\ \psi \in \mathsf{Sub}(\Gamma, \Gamma', \Delta, \Delta')$$

Figure 9 Examples of restricted cut rules.

$$T\ \frac{\Box\varphi, \varphi, \Gamma \Rightarrow \Delta}{\Box\varphi, \Gamma \Rightarrow \Delta}$$

$$5\ \frac{\Box\Gamma \Rightarrow \varphi, \Box\Delta}{\Sigma, \Box\Gamma \Rightarrow \Box\varphi, \Box\Delta}$$

Figure 10 Modal rules for $\mathbf{G.S5}$: $\mathbf{G.S5} := \mathbf{LK} + (\textsf{an-cut}) + (T) + (5)$.

3.2 Restricted Cuts and Interpolation

In many textbooks on proof theory [84,86], Maehara's method is presented as an application of the cut-admissibility theorem. In the previous section we assumed to have a cut-free calculus, but it is well-known that special forms of cut can be admitted for Maehara's method to prove the CIP (though not necessarily the LIP). This is especially useful for logics that do not admit a cut-free sequent calculus, but that have a complete calculus including restricted cut rules.

Restricted cut rules circumvent the problem of the general cut rule for Maehara's method where one has no grip on the propositional variables occurring in the cut formula. We present three types of restricted cuts.

- *Analytic cut*: The *analytic cut rule* (an-cut) is presented in Figure 9. The cut formula is restricted to subformulas of the conclusion sequent. The rule preserves the *global subformula property* stating that in any proof each formula occurring in it is a subformula of the endsequent. Calculi with the subformula property are often called *analytic*, justifying the name of the rule. Examples of Maehara's method with the analytic cut rule are for modal logic S5 [74] and bi-intuitionistic logic [52].
- *Semi-analytic cut rules*: *semi-analyticity* is the concept in which the propositional variables of the cut formula occur in *one* formula from the conclusion. In other words, the cut formula is any (arbitrarily complex) combination of subformulas of a formula from the conclusion, justifying the name *semi-analytic*. An example is ($\Box$-semi-cut) from Figure 9 introduced in [25] in a tableau calculus. In Section 4.1 we discuss a slightly different notion of semi-analytic rules, which provides a general setting in which Maehara's method can be applied.
- *Monochromatic cut* (e.g., [17]): this type of cut is more general than semi-analytic cuts where the cut formula may be any combination of subformulas of *multiple* formulas in the conclusion. One has to be careful which *multiple* formulas are allowed and, therefore, this type of cut is only defined for split sequents: an application of the cut rule is *monochromatic* in split sequent proof π with endsequent $\Gamma_1; \Gamma_2 \Rightarrow \Delta_1; \Delta_2$ iff the cut formula φ satisfies the following condition: $\mathsf{Var}(\varphi) \subseteq \mathsf{Var}(\Gamma_1) \cup \mathsf{Var}(\Delta_1)$ or $\mathsf{Var}(\varphi) \subseteq \mathsf{Var}(\Gamma_2) \cup \mathsf{Var}(\Delta_2)$, i.e., tracing the propositional variables only through the left or right side of Maehara's split.

To illustrate the use of such cuts in Maehara's method, we consider modal logic S5 as an example from [74]. S5 is not known to have a cut-free sequent calculus. Sequent calculus $\mathbf{G.S5} := \mathbf{LK} + (\mathsf{an\text{-}cut}) + (T) + (5)$, including the analytic cut rule, is complete for S5 (cf. Figures 4, 9, and 10 for all rules).

▶ **Theorem 23.** *Modal logic* S5 *has the CIP.*

Proof. The proof can be found in [74] and follows a Maehara-style argument such as in the proof of Theorem 16. We only state the interpolant constructions for the analytic cut rules and leave all other details to the reader. The interpolant construction for (an-cut) depends on the formula in the conclusion whose subformula the cut formula φ is. There are two cases:

$$L(\mathsf{an\text{-}cut})\ \frac{\Gamma_1;\Gamma_1' \overset{\theta_1}{\Longrightarrow} \Delta_1,\varphi;\Delta_1' \quad \varphi,\Gamma_2;\Gamma_2' \overset{\theta_2}{\Longrightarrow} \Delta_2;\Delta_2'}{\Gamma_1,\Gamma_2;\Gamma_1',\Gamma_2' \overset{\theta_1\vee\theta_2}{\Longrightarrow} \Delta_1,\Delta_2;\Delta_1',\Delta_2'}\ \varphi \in \mathsf{Sub}(\Gamma_1,\Gamma_2,\Delta_1,\Delta_2)$$

$$R(\mathsf{an\text{-}cut})\ \frac{\Gamma_1;\Gamma_1' \overset{\theta_1}{\Longrightarrow} \Delta_1;\Delta_1',\varphi \quad \Gamma_2;\Gamma_2',\varphi \overset{\theta_2}{\Longrightarrow} \Delta_2;\Delta_2'}{\Gamma_1,\Gamma_2;\Gamma_1',\Gamma_2' \overset{\theta_1\wedge\theta_2}{\Longrightarrow} \Delta_1,\Delta_2;\Delta_1',\Delta_2'}\ \varphi \in \mathsf{Sub}(\Gamma_1',\Gamma_2',\Delta_1',\Delta_2')$$

The variable condition 1 from Definition 12 is fulfilled because of the subformula requirements. ◀

In fact, S5 also admits the LIP. This cannot be shown using sequent calculus **G.S5**, but can be obtained by using a hypersequent calculus for it (see Section 5.3).

3.3 Uniform Interpolation via Pitts' Method

The UIP is a beautiful strengthening of the CIP (see also Chapter 9, the main chapter on uniform interpolation). Here, we are interested in Pitts' [76] important result that *propositional quantification* can be interpreted in intuitionistic propositional logic IPC (i.e., only using the propositional language). To show this, Pitts provides a proof-theoretic method showing that the UIP holds for IPC. The method was later adapted to (intuitionistic) modal logics [5,11,44]. The purpose of this section is to explain the main concepts in Pitts' method.

A Craig interpolant of $\varphi \to \psi$ is *uniform* if it *only* relies on φ (or only on ψ) meaning that it is a Craig interpolant that uniformly works for any provable implication $\varphi \to \psi'$ with ψ' in a specified vocabulary. Formally, we define the following, where a formula φ is *p-free* iff $p \notin \mathsf{Var}(\varphi)$, *$p^+$-free* iff $p \notin \mathsf{Var}^+(\varphi)$, and *$p^-$-free* iff $p \notin \mathsf{Var}^-(\varphi)$:

▶ **Definition 24** (Uniform interpolation)**.** *Fix a logic* L *over a language* $\mathcal{L}$*. We say that* L *has the* Uniform Interpolation Property (UIP) *iff for any formula* $\varphi \in \mathsf{Form}_{\mathcal{L}}$ *and propositional variable* $p \in \mathsf{Prop}$*, there are formulas* $\exists p\varphi, \forall p\varphi$ *over the same language* $\mathcal{L}$ *such that the following hold:*

1. $\exists p\varphi$ *and* $\forall p\varphi$ *are* p*-free, and* $\mathsf{Var}(\exists p\varphi) \subseteq \mathsf{Var}(\varphi)$ *and* $\mathsf{Var}(\forall p\varphi) \subseteq \mathsf{Var}(\varphi)$*;*
2. $\vdash_{\mathsf{L}} \varphi \to \exists p\varphi$ *and* $\vdash_{\mathsf{L}} \forall p\varphi \to \varphi$*;*
3. *For all* p*-free formulas* ψ*:*

   if $\vdash_{\mathsf{L}} \varphi \to \psi$*, then* $\vdash_{\mathsf{L}} \exists p\varphi \to \psi$.

   if $\vdash_{\mathsf{L}} \psi \to \varphi$*, then* $\vdash_{\mathsf{L}} \psi \to \forall p\varphi$.

Formula $\exists p\varphi$ *is called the* uniform post-interpolant *of* φ *w.r.t.* p*, and* $\forall p\varphi$ *is the* uniform pre-interpolant *of* φ *w.r.t.* p *in* L*. When* $\exists p\varphi$ *(resp.* $\forall p\varphi$*) exists for all formulas* $\varphi \in \mathsf{Form}_{\mathcal{L}}$ *and* $p \in \mathsf{Prop}$*, then we say that* L *has the* Post-Uniform Interpolation Property (PostUIP) *(resp.* Pre-Uniform Interpolation Property (PreUIP)*).*

The notations $\exists p\varphi$ and $\forall p\varphi$ are suggestive: the formulas do not really contain quantifiers, but they are defined over the (propositional/modal) language $\mathcal{L}$. However, the notation is justified by Pitts' result stating that uniform interpolants *interpret propositional quantifiers* of second order intuitionistic logic into the propositional language. Uniform interpolants are also connected to so-called *bisimulation quantifiers* (see Chapter 9 of this book). In the literature, and particularly in Chapter 1, post-uniform and pre-uniform interpolants are known as right-uniform and left-uniform interpolants respectively.

▶ Remark 25. In classical logic, we can define $\exists p\varphi := \neg\forall p(\neg\varphi)$. ⌟

The proof of the following property can be found, e.g., in [36].

▶ **Theorem 26** (Uniqueness)**.** *Let* χ_1 *and* χ_2 *both be uniform pre-interpolants for* φ *w.r.t.* p *in logic* L*. Then* $\vdash_{\mathsf{L}} \chi_1 \to \chi_2$ *and* $\vdash_{\mathsf{L}} \chi_2 \to \chi_1$*. The same holds for uniform post-interpolants.*

Recently, Kurahashi [54] introduced the notion of uniform Lyndon interpolation:

▶ **Definition 27** (Uniform Lyndon interpolation)**.** *A logic* L *over a language* $\mathcal{L}$ *has the* Uniform Lyndon Interpolation Property (ULIP)*, iff for any* $\varphi \in \mathsf{Form}_{\mathcal{L}}$ *and* $p \in \mathsf{Prop}$*, there are formulas* $\forall^{+}p\,\varphi$*,* $\forall^{-}p\,\varphi$*,* $\exists^{+}p\,\varphi$*, and* $\exists^{-}p\,\varphi$ *in* $\mathsf{Form}_{\mathcal{L}}$ *that satisfy the properties from Definition* 24*, except that each occurrence of* p*-free is replaced with* p^{+}*-free for* $\forall^{+}p\,\varphi$ *and* $\exists^{+}p\,\varphi$ *and with* p^{-}*-free for* $\forall^{-}p\,\varphi$ *and* $\exists^{-}p\,\varphi$*.*

▶ **Theorem 28** ([54])**.** *If a logic* L *has the UIP/ULIP, then it has the CIP/LIP.*

In fact, uniform interpolants are the *weakest* and *strongest* Craig interpolants.

Now we turn to the proof-theoretic method by Pitts. The ingredients for Pitts' method in comparison to Maehara's method are summarized in Table 1. Because of technical reasons we will not prove the UIP for IPC, as in the original proof by Pitts [76], but we will explain Pitts' method for modal logic K as done by Bílková in [11]. Recall Remark 25: we only focus on the construction of pre-interpolants $\forall p\varphi$.

Towards a sequent based formulation of the UIP we have the following in mind. A pre-interpolant for φ w.r.t. p works almost as a Craig interpolant for all implications $\psi \to \varphi$

Table 1 General overview of similarities and differences between Maehara's and Pitts' method. The term 'cut-free' is in quotes, because cut-free systems are nice for Maehara's method, but certainly not necessary (recall Section 3.2). Similarly, for Pitts' method, strongly terminating calculi are a nice feature to prove the UIP, but not necessary (see Remark 34).

Maehara's method	**Pitts' method**
Craig interpolation	Uniform interpolation
Sequent formulation of the Craig IP	Sequent formulation of the uniform IP
'Cut-free' sequent calculus	'Strongly terminating' sequent calculus
Proof of $\varphi \to \psi$	Proof search of φ

where ψ is p-free (almost, because it might violate the Craig variable condition). So φ on the right is fixed, but ψ on the left is flexible. This idea will be present in the split sequents in the following definition, where $\Gamma \Rightarrow \Delta$ is fixed on the right of the split, and the p-free contexts Π and Σ are flexible and always on the left side of the split.

▶ **Definition 29.** *A sequent calculus* $\mathbf{G}$ *over a language* $\mathcal{L}$ *has the* Sequent Pre-Uniform Interpolation Property (SPreUIP) *iff for any sequent* $\Gamma \Rightarrow \Delta$ *and* $p \in \mathsf{Prop}$*, there exists a formula* $A_p(\Gamma \Rightarrow \Delta) \in \mathsf{Form}_{\mathcal{L}}$ *such that the following properties hold:*

1. $\mathsf{Var}(A_p(\Gamma \Rightarrow \Delta)) \subseteq \mathsf{Var}(\Gamma, \Delta) \setminus \{p\}$*;*
2. $\vdash_{\mathbf{G}} \Gamma, A_p(\Gamma \Rightarrow \Delta) \Rightarrow \Delta$*;*
3. $\vdash_{\mathbf{G}} \Pi \Rightarrow A_p(\Gamma \Rightarrow \Delta), \Sigma$ *for any* p*-free multisets* Π *and* Σ *such that* $\vdash_{\mathbf{G}} \Pi, \Gamma \Rightarrow \Sigma, \Delta$*.*

▶ Remark 30. We only focus on modal logic K for which above sequent formulation of the UIP is sufficient because of Remark 25. When considering intuitionistic logics one additionally has to define a formula $E_p(\Gamma)$ that satisfies dual-like properties (not exactly dual since the construction of $E_p(\Gamma)$ depends on $A_p(\Gamma \Rightarrow \varphi)$). Here we leave out all such details and refer the reader to [5, 76]. ⌟

▶ Remark 31. Here we only define a sequent version for the UIP. One can also provide a sequent version for the ULIP, see [3]. ⌟

▶ **Theorem 32.** *Let* $\mathbf{G}$ *be a sequent calculus for a logic* L*. If* $\mathbf{G}$ *has the SPreUIP, then* L *has the PreUIP.*

Proof. Define $\forall p\varphi := A_p(\ \Rightarrow \varphi)$. We leave it to the reader to check the details. ◀

$A_p(\Gamma \Rightarrow \Delta)$ is constructed using the sequent calculus. Whereas for Maehara's method we are given a *provable* split sequent $\Gamma; \Gamma' \Rightarrow \Delta; \Delta'$, here we are given a sequent $\Gamma \Rightarrow \Delta$ that might be *unprovable*. So we cannot look at a *proof* of $\Gamma \Rightarrow \Delta$ because there simply might not be one. The trick is to construct the uniform interpolant via a *proof search* of $\Gamma \Rightarrow \Delta$. A proof

$$(\mathsf{id})^*\ \frac{}{p, \Gamma \Rightarrow \Delta, p} \qquad (K)^*\ \frac{\Gamma \Rightarrow \varphi}{\Pi, \Box\Gamma \Rightarrow \Delta, \Box\varphi}$$

$$(\bot_l)^*\ \frac{}{\bot, \Gamma \Rightarrow \Delta} \qquad \to_l\ \frac{\Gamma \Rightarrow \Delta, \varphi \quad \psi, \Gamma \Rightarrow \Delta}{\varphi \to \psi, \Gamma \Rightarrow \Delta} \qquad \vee_l\ \frac{\varphi, \Gamma \Rightarrow \Delta \quad \psi, \Gamma \Rightarrow \Delta}{\varphi \vee \psi, \Gamma \Rightarrow \Delta} \qquad \wedge_l\ \frac{\varphi, \psi, \Gamma \Rightarrow \Delta}{\varphi \wedge \psi, \Gamma \Rightarrow \Delta}$$

$$(\top_r)^*\ \frac{}{\Gamma \Rightarrow \Delta, \top} \qquad \to_r\ \frac{\varphi, \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \to \psi} \qquad \vee_r\ \frac{\Gamma \Rightarrow \Delta, \varphi, \psi}{\Gamma \Rightarrow \Delta, \varphi \vee \psi} \qquad \wedge_r\ \frac{\Gamma \Rightarrow \Delta, \varphi \quad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi}$$

Figure 11 Sequent calculus **G3K**.

search of a sequent is a tree in which, bottom-up, rules of the sequent calculus are applied in an attempt to find a proof. A key feature that we want of this proof search is that it is *finite*. This helps ensure that the algorithm constructing the uniform pre-interpolant terminates.

▶ **Definition 33** (Strong termination). *A sequent calculus is said to be* strongly terminating *iff for each sequent, any order of bottom-up applications of the rules results in a finite tree, i.e., it stops with leaves that are either axioms or unprovable sequents to which no rule can be applied.*

▶ Remark 34. We focus on strongly terminating calculi, but weaker forms of termination may also work, though they typically require additional machinery, e.g., for modal logic T [11], for a hypersequent calculus to prove the UIP for S5 [39], or in cyclic proofs [1]. ⌟

Many sequent calculi are not strongly terminating. For example, sequent calculus **LJ** is not strongly terminating due to contraction rule (c_l) (see Figure 7), but even when the explicit weakening and contraction rules are subsumed as in **G3ip** from [86], strong termination still fails, due to the intuitionistic version of rule ($\to_l$) where contraction is embedded as follows:

$$(\to_l)^*\ \frac{\varphi \to \psi, \Gamma \Rightarrow \varphi \quad \psi, \Gamma \Rightarrow \Delta}{\varphi \to \psi, \Gamma \Rightarrow \Delta}$$

where Δ has at most one formula. Consequently, Pitts used a strongly terminating calculus **G4ip** for IPC where $(\to_l)^*$ is replaced with four other implication rules [22, 43, 89, 90], an idea later extended to intuitionistic modal logics [46].

We look at modal logic K. Sequent calculus **G.K** is not strongly terminating because of the contraction rules. Instead, we will consider calculus **G3K** from Figure 11 with implicit weakening and contraction.

▶ **Theorem 35.** *Sequent calculus* **G3K** *for* K *is strongly terminating.*

Proof. We define the weight $w(\varphi)$ of a formula φ to be the number of symbols occurring in it counted as a multiset. For sequents, we define $w(\Gamma \Rightarrow \Delta) := \Sigma_{\varphi \in \Gamma} w(\varphi)$. For all rules in **G3K** it holds that the weight of each premise is smaller than the weight of the conclusion, meaning that the calculus is strongly terminating. ◀

Table 2 Uniform interpolant construction for **G3K**, where Φ and Ψ do not contain boxed formulas.

rule	$\Gamma \Rightarrow \Delta$ *matches*	$A_p(\Gamma \Rightarrow \Delta)$ *equals*
$(\mathsf{id})^*$	$r, \Gamma' \Rightarrow \Delta', r$	$\top$
$(\top_r)^*$	$\Gamma \Rightarrow \Delta', \top$	$\top$
$(\bot_l)^*$	$\bot, \Gamma' \Rightarrow \Delta$	$\top$
$(\wedge_l)$	$\Gamma', \psi_1 \wedge \psi_2 \Rightarrow \Delta$	$A_p(\Gamma', \psi_1, \psi_2 \Rightarrow \Delta)$
$(\vee_l)$	$\Gamma', \psi_1 \vee \psi_2 \Rightarrow \Delta$	$A_p(\Gamma', \psi_1 \Rightarrow \Delta) \wedge A_p(\Gamma, \psi_2 \Rightarrow \Delta)$
$(\rightarrow_l)$	$\Gamma', \psi_1 \rightarrow \psi_2 \Rightarrow \Delta$	$A_p(\Gamma' \Rightarrow \Delta, \psi_1) \wedge A_p(\Gamma, \psi_2 \Rightarrow \Delta)$
$(\wedge_r)$	$\Gamma \Rightarrow \Delta', \psi_1 \wedge \psi_2$	$A_p(\Gamma \Rightarrow \Delta', \psi_1) \wedge A_p(\Gamma \Rightarrow \Delta', \psi_2)$
$(\vee_r)$	$\Gamma \Rightarrow \Delta', \psi_1 \vee \psi_2$	$A_p(\Gamma \Rightarrow \Delta', \psi_1, \psi_2)$
$(\rightarrow_r)$	$\Gamma \Rightarrow \Delta', \psi_1 \rightarrow \psi_2$	$A_p(\Gamma, \psi_1 \Rightarrow \Delta', \psi_2)$
$(K)^*$ or no rule applies	$\Box\Gamma', \Phi \Rightarrow \Box\Delta', \Psi$ where $\Gamma' \neq \varnothing$	$\Diamond A_p(\Gamma' \Rightarrow \varnothing) \vee \bigvee_{q \in \Phi \setminus \{p\}} q \vee \bigvee_{r \in \Psi \setminus \{p\}} \neg r \vee \bigvee_{\psi \in \Delta'} \Box A_p(\Gamma' \Rightarrow \psi)$
$(K)^*$ or no rule applies	$\Phi \Rightarrow \Box\Delta', \Psi$	$\bigvee_{q \in \Phi \setminus \{p\}} q \vee \bigvee_{r \in \Psi \setminus \{p\}} \neg r \vee \bigvee_{\psi \in \Delta'} \Box A_p(\Rightarrow \psi)$

▶ Remark 36. Modal sequent rules (4) and (T) from Figure 5 prevent strong termination, because rule (4) duplicates the context Γ and rule (T) duplicates principal formula φ in the premise. In fact, logics K4 and S4 do not have the UIP [11, 34]. ⌟

▶ **Theorem 37.** *Sequent calculus* **G3K** *has the SPreUIP, and hence logic* K *enjoys the UIP.*

Proof idea. By recursion on $\Gamma \Rightarrow \Delta$ we construct the formula $A_p(\Gamma \Rightarrow \Delta)$ and prove that it acts as the pre-interpolant of $\Gamma \Rightarrow \Delta$ w.r.t. p. The algorithm starts a proof search bottom-up for sequent $\Gamma \Rightarrow \Delta$ and inductively defines the uniform interpolants. The algorithm follows the steps from Table 2 according to the shape of the sequents with priority given to the higher rows in the table. The left column of the table indicates the rule applied in the proof search, the middle column the shape of the conclusion of that rule, and the right column defines the pre-interpolant using the premise(s) of that rule. Rows 2–7 correspond to so-called *invertible rules*. We stress that for each invertible rule, the pre-interpolant for the conclusion can always be taken to be a conjunction of the pre-interpolants of the premises. The last row is a special case corresponding to the *non-invertible* rule (K) providing the $\Box$-clauses in the result, but it also corresponds to the possibility in which the proof search 'applies rule (K)' on a boxed formula of the extra context Σ on the 'left split' from condition 3 of Definition 29 providing the $\Diamond$-clause in the result. The algorithm terminates in the same ordering on weights w in which sequent calculus **G3K** terminates (Theorem 35). Indeed, all rows except for the latter reflect the rules of the calculus, and also the recursion in the last row reduces in the same way.

It remains to check the correctness of this algorithm, i.e., we need to check the conditions of Definition 29. This is based on syntactic arguments for which we refer to [11]. ◀

▶ **Example 38.** Let us apply the uniform interpolant algorithm for **G3K** to formula $\Box p \vee \Box \neg p$. We present it in a proof search where we compute $A_p(\cdot)$ of each sequent denoted in parentheses:

$$\vee_r \dfrac{\dfrac{\Rightarrow p \;\; (\bot) \qquad \to_r \dfrac{p \Rightarrow \bot \;\; (\bot)}{\Rightarrow \neg p \;\; (\bot)}}{\Rightarrow \Box p, \Box \neg p \;\; (\Box\bot \vee \Box\bot)}\ \text{Proof search on rule } (K)}{\Rightarrow \Box p \vee \Box \neg p \;\; (\Box\bot \vee \Box\bot)}$$

For the top sequents, no rule of **G3K** applies, and the last row of Table 2 applies with the empty disjunction yielding $\bot$. We leave it to the reader to check the conditions of Definition 29. ⌟

Similar proof-theoretic proofs of uniform interpolation have been provided for classical modal logics GL, S4.Grz, T [11], and D [44], as well as for intuitionistic modal logics iK, iKD [45], and iSL [23]. The method has been extended to general forms of calculi providing results for a wide range of intermediate and substructural logics as discussed in the next section.

The uniform interpolation algorithm has been formalized in the interactive theorem prover Coq/Rocq for logics IPC, K, GL, and iSL [23,24]. This has resulted in an online calculator of uniform interpolants [23,24]. This calculator has been useful to prove that certain second-order intuitionistic connectives are non-definable [50].

4 Interpolation in Universal Proof Theory

Universal proof theory, initiated by Iemhoff [44,45] and extended in later works [2–7], investigates general questions about the existence, equivalence, and characterization of well-behaved proof systems across broad classes of logics. It offers a uniform and modular method to establish logical properties, such as the CIP and UIP and the admissibility of Visser rules [6], by linking them to the existence of certain structurally constrained sequent calculi. This framework applies to many logics, notably intuitionistic, modal, and substructural ones, by defining syntactic conditions that guarantee these properties.

The method yields both positive and negative results. Positively, if a logic admits a (terminating) *semi-analytic* calculus satisfying the required structural conditions, it enjoys the CIP (and UIP). In Section 4.1 we introduce these calculi and see that they provide a general form for axioms and rules, such that as a result, the framework provides a systematic way to prove interpolation properties. Negatively, and more significantly, since these properties are rare, these results indicate that most logics cannot have a semi-analytic (resp. terminating semi-analytic) calculus which we will discuss in Section 4.2. In this sense, it not only provides a systematic way to prove interpolation results but also demonstrates the non-existence of well-behaved calculi for large classes of logics.

Although interpolation properties were the first to be investigated within this line of research, [6] introduces a new logical invariant: the admissibility of the Visser rules. This extension significantly broadens the scope of universal proof theory, moving beyond interpolation to encompass further proof-theoretic properties. Such a generalization is particularly valuable for obtaining negative results, as it provides new tools to capture the limitations of proof systems. In [6], a constructive sequent calculus is introduced, and one interesting result is that no intermediate logic other than IPC admits a constructive sequent calculus.

4.1 Semi-analytic Rules and Interpolation

The following definition introduces the so-called *semi-analytic* rules, a concept that is slightly weaker than the analyticity property of rules: a rule is *analytic* when the formulas in the premises are subformulas of the principal formula in the conclusion. As a result, these rules are broader and more general than the analytic ones.

▶ **Definition 39** (Semi-analytic rules, [5, 7])**.** *For $1 \leq i \leq n$ and $1 \leq j \leq m$, let Γ_i's, Δ_i's, and Π_j's be pairwise disjoint families of pairwise distinct multiset variables, $\overline{\varphi}_{ir}$'s, $\overline{\psi}_{js}$'s, and $\overline{\chi}_{js}$'s be multisets of formulas and φ a formula, where $1 \leq r \leq k_i$ and $1 \leq s \leq l_j$.*

A rule is called left single-conclusion semi-analytic *if it has the form*

$$\frac{\{\Pi_j, \overline{\psi}_{js} \Rightarrow \overline{\chi}_{js} \mid 1 \leq j \leq m, 1 \leq s \leq l_j\} \qquad \{\Gamma_i, \overline{\varphi}_{ir} \Rightarrow \Delta_i \mid 1 \leq i \leq n, 1 \leq r \leq k_i\}}{\Pi_1, \ldots, \Pi_m, \Gamma_1, \ldots, \Gamma_n, \varphi \Rightarrow \Delta_1, \ldots, \Delta_n}$$

and it satisfies the following variable condition:

$$\bigcup_{ir} \mathsf{Var}(\overline{\varphi}_{ir}) \cup \bigcup_{js} \mathsf{Var}(\overline{\psi}_{js}) \cup \bigcup_{js} \mathsf{Var}(\overline{\chi}_{js}) \subseteq \mathsf{Var}(\varphi).$$

In each instance of the rule, at most one of Δ_i's can be substituted by a formula and the rest must be empty, as the rule is single-conclusion.

A rule is called right single-conclusion semi-analytic *if it has the form*

$$\frac{\{\Gamma_i, \overline{\varphi}_{ir} \Rightarrow \overline{\chi}_{ir} \mid 1 \leq i \leq n, 1 \leq r \leq k_i\}}{\Gamma_1, \ldots, \Gamma_n \Rightarrow \varphi}$$

and the following variable condition holds:

$$\bigcup_{ir} \mathsf{Var}(\overline{\varphi}_{ir}) \cup \bigcup_{ir} \mathsf{Var}(\overline{\chi}_{ir}) \subseteq \mathsf{Var}(\varphi).$$

A multi-conclusion semi-analytic *rule can be defined similarly, when we allow the rule to be multi-conclusion (see [7, Definition 4.3]). A rule is called* semi-analytic *if it is a single-conclusion or multi-conclusion left or right semi-analytic rule.*

▶ **Example 40.** A generic example of a left single-conclusion semi-analytic rule is the following:

$$\frac{\Pi, \nu_{11}^1, \nu_{11}^2 \Rightarrow \rho_{11} \quad \Pi, \nu_{12} \Rightarrow \rho_{12} \quad \Gamma, \mu_1^1, \mu_1^2, \mu_1^3 \Rightarrow \Delta}{\Pi, \Gamma, \mu \Rightarrow \Delta}$$

where $\mathsf{Var}(\mu_1^1) \cup \mathsf{Var}(\mu_1^2) \cup \mathsf{Var}(\mu_1^3) \cup \mathsf{Var}(\nu_{11}^1) \cup \mathsf{Var}(\nu_{11}^2) \cup \mathsf{Var}(\nu_{12}) \cup \mathsf{Var}(\rho_{11}) \cup \mathsf{Var}(\rho_{12}) \subseteq \mathsf{Var}(\mu)$.

The left and middle premises are in the same family with the context Π in the antecedent. Thus, one copy of Π appears in the antecedent of the conclusion. ⌟

▶ **Example 41.** Concrete examples of multi-conclusion semi-analytic rules are the rules in Figure 4, except for the cut rule. The single-conclusion version of these rules are single-conclusion semi-analytic rules. Concrete non-examples of multi-conclusion semi-analytic rules include the cut rule, as it violates the variable condition. Considering the modal rules in Figure 5, rules (T) and $(S4)$ are single-conclusion semi-analytic, (K) and (D) are non-examples as the contexts do not remain intact from the premises to the conclusion, and (4) and (GL) are also non-examples, as context Γ disappears from the premise to the conclusion. Non-examples for single-conclusion semi-analytic rules are provided by the single-conclusion version of the mentioned rules, i.e., (K), (D), (4), (GL), and (cut). For another non-example, consider the following rule in the calculus **KC** introduced in [15] for the logic $\mathsf{KC} = \mathsf{IPC} + \neg\varphi \vee \neg\neg\varphi$:

$$\frac{\Gamma, \varphi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \to \psi, \Delta} \qquad \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \psi, \Pi \Rightarrow \Lambda}{\Gamma, \varphi \to \psi, \Pi \Rightarrow \Delta, \Lambda}$$

in which Δ must be substituted by multisets of *strictly negative formulas* (not containing all formulas in the language; see [15]). This rule is not semi-analytic as its context Δ is not free for arbitrary substitutions of multisets. ⌟

▶ Remark 42. Note that the semi-analytic cut rules introduced in Section 3.1 do not fall in the category of semi-analytic rules as per Definition 39. The reason is the simple fact that in a cut rule, there is no principal formula. However, it is possible to generalize the notion to also cover semi-analytic cut rules, a direction we leave for future work. ⌟

Now, we define a general form for axioms.

▶ **Definition 43** (Focused axioms, [7])**.** *A meta-sequent is a* focused axiom*, if it has one of the forms:*

$$\varphi \Rightarrow \varphi \qquad \Rightarrow \bar{\psi} \qquad \bar{\theta} \Rightarrow \qquad \Gamma, \bar{\varphi} \Rightarrow \Delta \qquad \Gamma \Rightarrow \bar{\varphi}, \Delta$$

where for each axiom $\mathsf{Var}(\mu) = \mathsf{Var}(\nu)$*, for any* $\mu, \nu \in \bar{\eta}$ *and* $\eta \in \{\varphi, \psi, \theta\}$*.*

Note that the term *focused* is different from the notion of 'focused proofs' used in proof-search, as in the works [62, 68].

▶ **Example 44.** The axioms in sequent calculi **LK** and **LJ** are all focused. More examples:

$$p, \neg p \Rightarrow \qquad \Rightarrow p, \neg p \qquad \Gamma, \neg\top \Rightarrow \Delta \qquad \Gamma \Rightarrow \Delta, \neg\bot$$

For a non-example, consider the axiom $(p, \neg p, q \Rightarrow)$, where p and q are distinct atoms. This axiom is not focused, as the set of the variables of p and q are not equal. ⌟

▶ **Definition 45.** *A calculus* $\mathbf{G}$ *is* (single-conclusion) multi-conclusion semi-analytic *if each rule in* $\mathbf{G}$ *is (single-conclusion) multi-conclusion semi-analytic and its axioms are (single-conclusion versions of) focused axioms. In case the language contains the modality* $\Box$*, any union of the sets of rules* $\{(K)\}, \{(4)\}, \{(K), (D)\}, \{(K), (T)\}$ *are also allowed to appear in* $\mathbf{G}$*.*

We finish this subsection by defining what it means for a calculus to be fully terminating. This is a stronger notion than *strong termination* introduced in Definition 33.

▶ **Definition 46** (Fully terminating sequent calculi)**.** *Let* $\mathbf{G}$ *be a calculus and* $\preceq$ *be a well-founded order on sequents. We say that* $\mathbf{G}$ *is* fully terminating with respect to $\preceq$ *if for any sequent* S *there are at most finitely many instances of the rules in* $\mathbf{G}$ *with the conclusion* S*. Moreover, we require the following to be strictly less than* S *with respect to the order* $\preceq$*:*

- *the premises of all instances of a rule where the conclusion is* S*,*
- *proper subsequents of* S*, and*
- *in case that the language contains the modality* $\Box$*, any* $S' = (\Gamma, \Pi \Rightarrow \Delta, \Lambda)$*, where* $\Pi \cup \Lambda \neq \varnothing$ *and* $S = (\Gamma, \Box\Pi \Rightarrow \Delta, \Box\Lambda)$*.*

The calculus $\mathbf{G}$ *is* fully terminating *if there is an order* $\preceq$ *such that* $\mathbf{G}$ *is fully terminating with respect to* $\preceq$*.*

▶ Remark 47. In modal logics, a fully terminating semi-analytic calculus cannot contain the modal rules (4) and (T) presented in Figure 5. This means that any fully semi-analytic modal sequent calculus can contain the set of modal rules $\{(K)\}$ and $\{(K), (D)\}$. ⌟

A useful rule of thumb for interpolation is that if a logic L has a semi-analytic calculus $\mathbf{G}$, then L enjoys the CIP, and if $\mathbf{G}$ is fully terminating, it also enjoys the UIP. The proofs essentially follow Maehara's approach for the CIP and Pitts' method for the UIP. Of course, there are a lot of technical details to check for which we refer to [5, 7].

4.2 Interpolation and (Non-)Existence of Sequent Calculi

In this section, we apply the uniform methodology introduced in the context of universal proof theory and present a rich number of positive and negative results based on the CIP and UIP across different logics. For a positive result we first observe that a given logic has a (fully terminating) semi-analytic calculus, which allows us to conclude that this logic has the CIP (UIP). For a negative result we use the contraposition: if we know that a logic does not have the CIP (UIP), we immediately conclude that this logic cannot have a (fully terminating) semi-analytic calculus.

4.2.1 Intermediate and Modal Logics

The following theorem is the main positive result with regard to proving the CIP and UIP.

▶ **Theorem 48** ([5, 7]). *Let* L *be a logic over a language* $\mathcal{L} \in \{\mathcal{L}_{\mathsf{p}}, \mathcal{L}_{\Box}\}$ *and* $\mathbf{G}$ *be a single-conclusion (resp. multi-conclusion) semi-analytic calculus for* L.

- *If* $\mathcal{L} = \mathcal{L}_{\mathsf{p}}$ *and if* $\mathsf{IPC} \subseteq \mathsf{L}$ *(resp.* $\mathsf{L} = \mathsf{CPC}$*), then* L *has the CIP. Moreover, if* $\mathbf{G}$ *is fully terminating, then* L *has the UIP.*
- *If* $\mathcal{L} = \mathcal{L}_{\Box}$ *and if* $\mathsf{K} \subseteq \mathsf{L}$*, then* L *has the CIP. Moreover, if* $\mathbf{G}$ *is fully terminating, then* L *has the UIP.*

The theorem in its current form does not yield strong general positive results for two reasons. First, in the realm of intermediate logic it is known that only seven consistent intermediate logics have the CIP and UIP, namely IPC, LC, KC, Bd_2, Sm, GSc, and CPC [65]. Second, for modal calculi, being semi-analytic depends on an ad-hoc examination of rules, as we currently lack a general form for a semi-analytic modal rule. However, for modal logics there are many proof-theoretic proofs of the CIP, e.g., for modal logic GL [11] and intuitionistic modal logics in [45] (see also Section 3).

To state our negative results (the following corollaries), we rely on the nonexistence of the CIP and UIP in a wide range of logics [12, 34, 35, 65, 66]. Consequently, we get the nonexistence of (fully terminating) calculi consisting only of semi-analytic rules and focused axioms.

▶ **Corollary 49** (Negative results, CIP). *The following results hold:*

- *Except for logics* IPC, LC, KC, Bd_2, Sm, GSc, *and* CPC*, no intermediate logic has a single- or multi-conclusion semi-analytic sequent calculus.*
- *With the possible exception of at most 37 for* $\mathsf{S4}$ *and 6 for* Grz*, none of their consistent extensions have a single- or multi-conclusion semi-analytic sequent calculus.*

▶ **Corollary 50** (Negative results, UIP). *With the possible exception of at most six extensions of* $\mathsf{S4}$*, none of the extensions of either* $\mathsf{S4}$ *or* $\mathsf{K4}$ *have a single- or multi-conclusion fully terminating semi-analytic sequent calculus.*

4.2.2 (Modal) Substructural and Linear Logics

The results of the previous section are generalized to (modal) substructural and linear logics. Consider the *bounded* language $\mathcal{L}^b := \{\wedge, \vee, \rightarrow, *, 0, 1, \top, \bot\}$ and the *unbounded* language $\mathcal{L}^u := \mathcal{L}^b \setminus \{\top, \bot\}$. Denote $\mathcal{L}^b_! := \mathcal{L}^b \cup \{!\}$ and $\mathcal{L}^u_! := \mathcal{L}^u \cup \{!\}$. We recall some well-known calculi for substructural and linear rules in Figures 12 and 13. The left column of Figure 14 presents well-known sequent calculi using these rules, the weakening and contraction rules from Figure 4, and $(\bot_l)^*$ and $(\top_r)^*$ from Figure 11. The single-conclusion versions of the calculi in the left column are $\mathbf{FL_e}$, $\mathbf{FL_{ew}}$, $\mathbf{FL_{ec}}$, **IMALL**, **AIMALL**, and **RIMALL** respectively.

For the language $\mathcal{L}^b_!$ that contains '!', we present sequent calculi for several logics on the right column of Figure 14, using the rules in Figure 13. It is worth noting that by denoting ! in

$$\mathsf{id}_\varphi\ \frac{}{\varphi \Rightarrow \varphi} \qquad 0_l\ \frac{}{0 \Rightarrow} \qquad 1_r\ \frac{}{\Rightarrow 1} \qquad 0_r\ \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, 0} \qquad 1_l\ \frac{\Gamma \Rightarrow \Delta}{1, \Gamma \Rightarrow \Delta}$$

$$*_l\ \frac{\varphi, \psi, \Gamma \Rightarrow \Delta}{\varphi * \psi, \Gamma \Rightarrow \Delta} \qquad (\to_l)^*\ \frac{\Gamma \Rightarrow \Delta, \varphi \quad \psi, \Gamma' \Rightarrow \Delta'}{\varphi \to \psi, \Gamma, \Gamma' \Rightarrow \Delta, \Delta'} \qquad \vee_l\ \frac{\varphi, \Gamma \Rightarrow \Delta \quad \psi, \Gamma \Rightarrow \Delta}{\varphi \vee \psi, \Gamma \Rightarrow \Delta} \qquad \wedge_l^i\ \frac{\varphi_i, \Gamma \Rightarrow \Delta}{\varphi_0 \wedge \varphi_1, \Gamma \Rightarrow \Delta}\ (i{=}0,1)$$

$$*_r\ \frac{\Gamma \Rightarrow \Delta, \varphi \quad \Gamma' \Rightarrow \Delta', \psi}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta', \varphi * \psi} \qquad \to_r\ \frac{\varphi, \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \to \psi} \qquad \vee_r^i\ \frac{\Gamma \Rightarrow \Delta, \varphi_i}{\Gamma \Rightarrow \Delta, \varphi_0 \vee \varphi_1}\ (i{=}0,1) \qquad \wedge_r\ \frac{\Gamma \Rightarrow \Delta, \varphi \quad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi}$$

Figure 12 Sequent calculus $\mathbf{CFL_e}$ for substructural logic $\mathsf{CFL_e}$. Rules $(\vee_l)$, $(\wedge_l^i)$, $(\to_r)$, $(\vee_r^i)$, $(\wedge_r)$, and (cut) are the same as in Figure 4.

$$!_r\ \frac{!\Gamma \Rightarrow \varphi}{!\Gamma \Rightarrow !\varphi} \qquad !_l\ \frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma, !\varphi \Rightarrow \Delta} \qquad \mathsf{w}!\ \frac{\Gamma \Rightarrow \Delta}{\Gamma, !\varphi \Rightarrow \Delta} \qquad \mathsf{c}!\ \frac{\Gamma, !\varphi, !\varphi \Rightarrow \Delta}{\Gamma, !\varphi \Rightarrow \Delta} \qquad K!\ \frac{\Gamma \Rightarrow \varphi}{!\Gamma \Rightarrow !\varphi} \qquad D!\ \frac{\Gamma \Rightarrow}{!\Gamma \Rightarrow}$$

Figure 13 Sequent rules for '!' for linear logic.

the language by $\Box$, the rules in Figure 13 resemble standard modal rules. That is why we, e.g., write $\mathbf{FL_eK}$ to mean $\mathbf{FL_e} + \{K!\}$, write $\mathbf{FL_{ew}D}$ to mean $\mathbf{FL_{ew}} + \{K!, D!\}$, or write $\mathbf{FL_eT}$ to mean $\mathbf{FL_e} + \{!_l\}$, and likewise, we write $\mathsf{FL_eK}$, $\mathsf{FL_{ew}D}$, and $\mathsf{FL_eT}$ for their corresponding logics. The following calculi are proved to enjoy cut admissibility in [20, 30, 32, 40, 41, 48, 49, 73–75, 85], as well as [21, Proposition 4]:

▶ **Theorem 51.** *The following sequent calculi enjoy cut admissibility:* $\mathbf{FL_e}$, $\mathbf{FL_{ew}}$, $\mathbf{FL_{ec}}$, $\mathbf{CFL_e}$, $\mathbf{CFL_{ew}}$, $\mathbf{CFL_{ec}}$, **LJ**, **LK**, **IMALL**, **AIMALL**, **RIMALL**, **MALL**, **AMALL**, **RMALL**, $\mathbf{X} + \{(!_r), (!_l)\}$ *where* $\mathbf{X} \in \{\mathbf{CFL_e}, \mathbf{CFL_{ew}}, \mathbf{CFL_{ec}}, \mathbf{LK}\}$, $\mathbf{CFL_e} + \{(!_r), (!_l), (\mathsf{c}!)\}$, $\mathbf{CFL_{ew}} + \{(!_r), (!_l), (\mathsf{c}!)\}$, **ILL**, **CLL**, **ALL**, **RLL**, **ELL**.

Then, the analog of Theorem 48 for substructural and linear logics will become the following (the results can be generalized to multimodal substructural logics [7], which we do not cover due to space limitations):

▶ **Theorem 52** ([7]). *Let* L *be a logic over* $\mathcal{L} \in \{\mathcal{L}^b, \mathcal{L}^u, \mathcal{L}^b_!, \mathcal{L}^u_!\}$ *and* $\mathbf{G}$ *be a single-conclusion (resp. multi-conclusion) semi-analytic calculus for* L*. Then,*

- *if* $\mathcal{L} = \mathcal{L}^b$ *or* $\mathcal{L} = \mathcal{L}^b_!$ *and* $\mathsf{IMALL} \subseteq \mathsf{L}$ *(resp.* $\mathsf{MALL} \subseteq \mathsf{L}$*), then* L *has the CIP, and*
- *if* $\mathcal{L} = \mathcal{L}^u$ *or* $\mathcal{L} = \mathcal{L}^u_!$ *and* $\mathsf{FL_e} \subseteq \mathsf{L}$ *(resp.* $\mathsf{CFL_e} \subseteq \mathsf{L}$*), then* L *has the CIP.*

▶ **Corollary 53** (Positive results). *Concrete examples of the logics that enjoy the CIP are the logics of the sequent calculi in Theorem 51.*

Calculi over $\mathcal{L}^u$:

$$\begin{aligned}
\mathbf{CFL_e} &:= \text{Figure 12}\\
\mathbf{CFL_{ew}} &:= \mathbf{CFL_e} + \{(\mathsf{w}_l), (\mathsf{w}_r)\}\\
\mathbf{CFL_{ec}} &:= \mathbf{CFL_e} + \{(\mathsf{c}_l), (\mathsf{c}_r)\}
\end{aligned}$$

Calculi over $\mathcal{L}^b$:

$$\begin{aligned}
\mathbf{MALL} &:= \mathbf{CFL_e} + \{(\bot_l)^*, (\top_r)^*\}\\
\mathbf{AMALL} &:= \mathbf{MALL} + \{(\mathsf{w}_l), (\mathsf{w}_r)\}\\
\mathbf{RMALL} &:= \mathbf{MALL} + \{(\mathsf{c}_l), (\mathsf{c}_r)\}
\end{aligned}$$

Calculi over $\mathcal{L}^b_!$:

$$\begin{aligned}
\mathbf{CLL} &:= \mathbf{MALL} + \{(!_r), (!_l), (\mathsf{w}!), (\mathsf{c}!)\}\\
\mathbf{ILL} &:= \mathbf{IMALL} + \{(!_r), (!_l), (\mathsf{w}!), (\mathsf{c}!)\}\\
\mathbf{ELL} &:= \mathbf{MALL} + \{(\mathsf{w}!), (\mathsf{c}!), (K!), (D!)\}\\
\mathbf{ALL} &:= \mathbf{CLL} + \{(\mathsf{w}_l), (\mathsf{w}_r)\}\\
\mathbf{RLL} &:= \mathbf{CLL} + \{(\mathsf{c}_l), (\mathsf{c}_r)\}
\end{aligned}$$

Figure 14 Sequent calculi for logics over different languages. In **ILL** we use the single-conclusion version of the rules. Rules (w_l), (w_r), (c_l), and (c_r) are displayed in Figure 4, and rules $(\bot_l)^*$ and $(\top_r)^*$ in Figure 11.

Proof. The presented sequent calculi for these logics consist of focused axioms and semi-analytic rules. Therefore, by Theorem 48, we get the CIP for their logics. ◀

We use a series of facts regarding the nonexistence of the CIP in a wide range of logics to obtain our negative results. Before that, we recall some of these logics. Figure 15 presents equational conditions to define varieties of pointed commutative residuated lattices and Figure 16 presents equational logics in the usual sense. Let $\mathbf{A}$ be a single pointed commutative residuated lattice and $\mathbf{Fm}$ be the formula algebra for the language of pointed commutative residuated lattices. Given any variety $\mathbb{V}$ of pointed commutative residuated lattices, a *logic* $\mathsf{L}^{\mathbb{V}}$ may be defined by fixing, for all $\Gamma \cup \{\varphi\} \subseteq \mathbf{Fm}$,

$$\begin{aligned}
\Gamma \vdash_{\mathsf{L}^{\mathbb{V}}} \varphi \Leftrightarrow \text{for any } \mathbf{A} \in \mathbb{V} \text{ and homomorphism } v\colon \mathbf{Fm} \to \mathbf{A}:\\
\text{if for all } \psi \in \Gamma, \quad 1 \leqslant v(\psi), \quad \text{then} \quad 1 \leqslant v(\varphi)
\end{aligned}$$

By $\mathcal{V}(\mathbb{K})$, we mean the variety generated by a class of pointed commutative residuated lattices $\mathbb{K}$. Let us define some fundamental fuzzy logics. For any $n > 1$, let $L_n = \{0, \frac{1}{n-1}, \ldots, \frac{n-2}{n-1}, 1\}$ and

$$\mathbb{L}_n = \langle L_n, min, max, *_{\text{Ł}}, \to_{\text{Ł}}, 1, 0\rangle \quad , \quad \mathbb{G}_n = \langle L_n, min, max, min, \to_G, 1, 0\rangle$$

where $x *_{\text{Ł}} y = max\{0, x + y - 1\}$, $x \to_{\text{Ł}} y = min\{1, 1 - x + y\}$, and $x \to_G y$ is y if $x > y$, otherwise 1. Define $\mathsf{Ł}_n$ and G_n for $n > 1$ as the logics of $\mathcal{V}(\mathbb{L}_n)$ and $\mathcal{V}(\mathbb{G}_n)$, respectively (see [67]). Separately, define the relevance logic R as the logic of the variety of distributive pointed commutative residuated lattices satisfying $x \leq x * x$ and $\neg\neg x = x$ for any x.

(prl)	$1 \leq (x \to y) \vee (y \to x)$	(fp)	$0 = 1$
(dis)	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	(div)	$x * (x \to y) = y * (y \to x)$
(inv)	$\neg\neg x = x$	(can)	$x \to (x * y) = y$
(int)	$x \leq 1$	(rcan)	$1 = \neg x \vee ((x \to (x * y)) \to y)$
(bd)	$0 \leq x$	(nc)	$x \wedge \neg x \leq 0$
(id)	$x = x * x$		

Figure 15 Some equational conditions.

UL^-	$:= \mathsf{FL_e} + (\mathsf{prl}) + (\mathsf{dis})$	Ł	$:= \mathsf{MTL} + (\mathsf{div}) + (\mathsf{inv})$
IUL^-	$:= \mathsf{UL}^- + (\mathsf{inv})$	P	$:= \mathsf{MTL} + (\mathsf{div}) + (\mathsf{rcan})$
MTL	$:= \mathsf{UL}^- + (\mathsf{int}) + (\mathsf{bd})$	CHL	$:= \mathsf{UL}^- + (\mathsf{int}) + (\mathsf{fp}) + (\mathsf{div}) + (\mathsf{can})$
SMTL	$:= \mathsf{MTL} + (\mathsf{nc})$	UML^-	$:= \mathsf{UL}^- + (\mathsf{id})$
IMTL	$:= \mathsf{MTL} + (\mathsf{inv})$	RM^e	$:= \mathsf{UML}^- + (\mathsf{inv})$
BL	$:= \mathsf{MTL} + (\mathsf{div})$	IUML^-	$:= \mathsf{UML}^- + (\mathsf{inv}) + (\mathsf{fp})$
G	$:= \mathsf{MTL} + (\mathsf{id})$	A	$:= \mathsf{IUL}^- + (\mathsf{fp}) + (\mathsf{can})$

Figure 16 Some substructural logics. In each logic, axioms (prl) and (dis) are present.

▶ **Corollary 54** (Negative results, CIP)**.** *The following two families of logics lack the CIP and hence cannot have a single-conclusion semi-analytic sequent calculus. The first family:*

- ([10, 53, 88][67, Corollary 3.2])**.** UL^-, MTL, SMTL, BL, P, CHL, G_n *(for* $n \geq 4$*);*
- ([65])**.** *All consistent extensions of* IPC*, except for possibly* IPC*,* G*,* KC*,* Bd_2*,* Sm*,* GSc*, and* CPC*;*
- ([10,70][67, Theorem 3.3])**.** *All consistent* BL*-extensions, except for possibly* G*,* G3 *and* CPC*;*

And the second family:

- ([10, 53, 88][67, Corollary 3.2])**.** R, A, IUL^-, IMTL, Ł, $\mathsf{Ł}_n$ *(for* $n \geq 3$*);*
- ([67, Corollary 3.5][51])**.** *All consistent* IMTL*-extensions, and consequently, all consistent extensions of* Ł*, except for* CPC*;*
- ([67, Theorem 4.9])**.** *All consistent extensions of* RM^e*, except for possibly* RM^e*,* IUML^-*,* CPC*,* RM^e_3*,* RM^e_4*,* $\mathsf{CPC} \cap \mathsf{IUML}^-$*,* $\mathsf{RM}^e_4 \cap \mathsf{IUML}^-$*, and* $\mathsf{CPC} \cap \mathsf{RM}^e_3$*. This includes:* RM^e_n *for* $n \geq 5$*,* $\mathsf{RM}^e_{2m} \cap \mathsf{RM}^e_{2n+1}$ *for* $n \geq m \geq 1$ *with* $n \geq 2$*, and* $\mathsf{RM}^e_{2m} \cap \mathsf{IUML}^-$ *for* $m \geq 3$*;*
- ([28, Corollary 3.14])**.** *Continuum-many extensions of* MALL *and* CLL*.*

The logics in the second family do not have a multi-conclusion semi-analytic calculus either.

Regarding the UIP, we have the following result. Our theorem generalizes the results of [8] and [12] to also cover the modal cases:

Modal axioms:

(M) $\Box(\varphi \wedge \psi) \to \Box\varphi \wedge \Box\psi$ (C) $\Box\varphi \wedge \Box\psi \to \Box(\varphi \wedge \psi)$ (N) $\Box\top$

Conditional axioms:

(CM) $(\varphi \rhd \psi \wedge \theta) \to (\varphi \rhd \psi) \wedge (\varphi \rhd \theta)$ (CN) $\varphi \rhd \top$ (ID) $\varphi \rhd \varphi$

(CC) $(\varphi \rhd \psi) \wedge (\varphi \rhd \theta) \to (\varphi \rhd \psi \wedge \theta)$ (CEM) $(\varphi \rhd \psi) \vee (\varphi \rhd \neg\psi)$

Modal and conditional rules:

$$\text{(E)}\ \frac{\varphi \leftrightarrow \psi}{\Box\varphi \leftrightarrow \Box\psi} \qquad \text{(CE)}\ \frac{\varphi_0 \leftrightarrow \varphi_1 \quad \psi_0 \leftrightarrow \psi_1}{\varphi_0 \rhd \psi_0 \to \varphi_1 \rhd \psi_1}$$

Figure 17 Modal and conditional axioms and rules.

▶ **Corollary 55** (Positive results, UIP). *Logics* $\mathsf{CFL_e}$, $\mathsf{CFL_{ew}}$, *and* CPC, *their* K *and* D *modal versions, as well as logics* $\mathsf{FL_e}$, $\mathsf{FL_{ew}}$, $\mathsf{FL_eK}$, $\mathsf{FL_{ew}K}$, $\mathsf{FL_eD}$, $\mathsf{FL_{ew}D}$, $\mathsf{FL_eT}$, *and* $\mathsf{FL_{ew}T}$ *have the UIP.*

Proof. The rules of the usual calculi of these logics are semi-analytic, and their axioms are focused, and in the absence of the contraction rule, the calculi are clearly fully terminating. ◀

4.2.3 Non-normal Modal and Conditional Logics

It is possible to extend the methodology to cover (non-normal) modal logics, conditional logics, and intuitionistic modal logics. As a witness to the generality of the methodology, we yet again apply it to a wide range of logics and this time we prove the ULIP. First, we need to introduce these logics.

Set $\mathcal{L}_{\rhd} := \{\wedge, \vee, \to, \bot, \rhd\}$ as the language of conditional logics. For non-normal modal and conditional logics, the property U(L)IP is less explored than in the setting of normal logics. Recall that the basic non-normal modal logic E (resp. the conditional logic CE) is the smallest set of $\mathcal{L}_{\Box}$-formulas (resp. $\mathcal{L}_{\rhd}$-formulas) containing classical tautologies, closed under (MP) and (E) (resp. (CE)). Define the other non-normal modal and conditional logics as in Figure 18, by adding the axioms in Figure 17. Also, recall the well-known intuitionistic non-normal modal logic, namely the intuitionistic monotone modal logic iM, which is axiomatized over IPC by the rule (E) and the axiom $\Box(\varphi \wedge \psi) \to \Box\varphi \wedge \Box\psi$.

To prove the ULIP for logics, we need to extend this concept to sequent calculi, similar to the case of the UIP. If a sequent calculus has this form of the ULIP, then its corresponding logic has the ULIP as well. The summary of the results is provided in the following theorem.

▶ **Theorem 56** (Positive results, [3,4]). *The following have the ULIP (hence the UIP and LIP):*

- *Non-normal modal logics:* E, M, EN, MN, MC, *and* K.
- *Conditional logics:* CE, CM, CEN, CMN, CMC, CK, *and* CKID.
- *The intuitionistic monotone modal logic* iM.

Non-normal modal logics:

$\mathsf{EN} := \mathsf{E} + (\mathsf{N})$ $\quad \mathsf{M} := \mathsf{E} + (\mathsf{M})$ $\quad \mathsf{MN} := \mathsf{M} + (\mathsf{N})$ $\quad \mathsf{MC} := \mathsf{M} + (\mathsf{C})$

$\mathsf{K} := \mathsf{MC} + (\mathsf{N})$ $\quad \mathsf{EC} := \mathsf{E} + (\mathsf{C})$ $\quad \mathsf{ECN} := \mathsf{EC} + (\mathsf{N})$

Conditional logics:

$\mathsf{CEN} := \mathsf{CE} + (\mathsf{CN})$ $\quad \mathsf{CM} := \mathsf{CE} + (\mathsf{CM})$ $\quad \mathsf{CMN} := \mathsf{CM} + (\mathsf{CN})$ $\quad \mathsf{CMC} := \mathsf{CM} + (\mathsf{CC})$

$\mathsf{CK} := \mathsf{CMC} + (\mathsf{CN})$ $\quad \mathsf{CEC} := \mathsf{CE} + (\mathsf{CC})$ $\quad \mathsf{CECN} := \mathsf{CEC} + (\mathsf{CN})$ $\quad \mathsf{CKID} := \mathsf{CK} + (\mathsf{ID})$

$\mathsf{CKCEM} := \mathsf{CK} + (\mathsf{CEM})$ $\quad \mathsf{CKCEMID} := \mathsf{CKCEM} + (\mathsf{ID})$

Figure 18 Non-normal modal and conditional logics.

▶ **Theorem 57** (Negative results, [3])**.**

- *The logics* CKCEM *and* CKCEMID *do* not *have the ULIP (but they have the UIP). Hence, they cannot have a fully terminating semi-analytic calculus.*
- *Non-normal modal logics* EC *and* ECN *and conditional logics* CEC *and* CECN *do* not *have the CIP (and hence no U(L)IP). Therefore, they cannot have a semi-analytic calculus.*

5 Interpolation Using Generalizations of Sequent Calculi

For many logics, no cut-free sequent calculus is known. While certain types of restricted cuts do not impede the Maehara method (see Section 3.2), it is often easier to find a cut-free *hypersequent*, *nested sequent*, or *labelled sequent* calculus for a logic than a (semi-)analytic sequent calculus. In addition, the former are bettter suited to demonstrate the LIP. In this section we show how to adapt the Maehara method to these generalizations of sequents.

We aim to provide a general manual on how to construct interpolants to further explore interpolation properties for other logics of the reader's interest besides the logics treated in this section. We do this by presenting a reverse engineering strategy. What we mean by this is that we will analyze the form of the rules, which will guide us in coming up with the correct constructions of interpolants.

Since hypersequents and nested sequents can usually be translated into the labelled sequent format and there are powerful general methods of constructing cut-free labelled sequent calculi based on Kripke semantics of the logic [71, 72], we first focus on labelled sequents in Sections 5.1 and 5.2 for modal logics over CPC. In Section 5.3, we then briefly explain how to adapt this method to hypersequents and nested sequents.

The method described in this section is sometimes criticized for the explicit use of semantic elements. In this respect, we would like to cite the alternative formulation of the method in [63] that avoids such explicit use.

5.1 Labelled Sequents

A *labelled sequent* (over a language $\mathcal{L}$) is a sequent (over $\mathcal{L}$) enriched with labels: every formula φ in the sequent is supplied with some label i, resulting in a labelled formula $i{:}\varphi$. Labels i can be taken from any set. W.l.o.g., we will use integer labels from $\mathbb{N}$.

The presented method of proving interpolation was originally developed by Fitting and Kuznets for nested sequents [27] and later adapted to labelled sequents by Kuznets in [57]. Here we largely follow [58], where additional details can be found. As in Section 3, we will present general conditions guaranteeing that interpolation holds, but based on Kripke frame conditions rather than the sequent rule shape.

Like labelled sequents themselves, this method relies on the Kripke semantics. In particular, labels i are interpreted as Kripke worlds. Whereas standard sequents $\Gamma \Rightarrow \Delta$ consist of two multisets of formulas (antecedent Γ and succedent Δ), labelled sequents $\mathcal{R}, \Gamma \Rightarrow \Delta$ are comprised of two multisets Γ and Δ of *labelled formulas* plus relational part $\mathcal{R}$. This $\mathcal{R}$ consists of relational atoms of the form iRj, where i and j are labels and R is the Kripke accessibility relation.[1] $\mathtt{L}(S)$ denotes the set of all labels occurring in S. For instance, $\mathtt{L}(1R2,\ 2{:}\varphi,\ 4{:}p \ \Rightarrow\ 4{:}\psi) = \{1, 2, 4\}$. We write $i \in S$ to mean $i \in \mathtt{L}(S)$.

▶ **Definition 58** (Labelled semantics). *Let $\mathcal{M} = (W, R, V)$ be a Kripke model. For a labelled sequent $S = \mathcal{R}, \Gamma \Rightarrow \Delta$, a function $[\![\cdot]\!] \colon X \to W$ is called an* interpretation of S into $\mathcal{M}$ *iff $X \supseteq \mathtt{L}(S)$, i.e., $[\![\cdot]\!]$ interprets all labels of S, and $[\![i]\!] R [\![j]\!]$ in $\mathcal{M}$ whenever $iRj \in \mathcal{R}$. For such an interpretation, $\mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Gamma \Rightarrow \Delta$ iff*

$$\textit{either} \quad \mathcal{M}, [\![i]\!] \not\models \varphi \textit{ for some } i{:}\varphi \in \Gamma \quad \textit{or} \quad \mathcal{M}, [\![j]\!] \models \psi \textit{ for some } j{:}\psi \in \Delta,$$

in which case we say that $\mathcal{R}, \Gamma \Rightarrow \Delta$ is true for $[\![\cdot]\!]$. Otherwise, it is false. S is valid in a class $\mathcal{C}$ of models, written $\mathcal{C} \models S$, iff S is true for all interpretations $[\![\cdot]\!]$ of S into models $\mathcal{M} \in \mathcal{C}$.

The notion of validity does not change if interpretations are defined exactly on the labels $\mathtt{L}(S)$ of a sequent S rather than on any superset $X \supseteq \mathtt{L}(S)$ thereof.

▶ Remark 59. An ordinary sequent $\Gamma \Rightarrow \Delta$ can be viewed as a single-label labelled sequent $1{:}\Gamma \Rightarrow 1{:}\Delta$ with $\mathcal{R} = \varnothing$, where $i{:}\Pi := \{i{:}\varphi \mid \varphi \in \Pi\}$. This representation is faithful: any mapping $[\![\cdot]\!]$ defined on 1 is an interpretation of $1{:}\Gamma \Rightarrow 1{:}\Delta$ and $\mathcal{M}[\![\cdot]\!] \models 1{:}\Gamma \Rightarrow 1{:}\Delta$ iff $\mathcal{M}, [\![1]\!] \models \bigwedge \Gamma \to \bigvee \Delta$. ⌟

Inference rules and proofs for ordinary sequents are defined in Section 2.2, so we will use the same notation. Examples of labelled sequent rules can be found in Figures 19–21 which will be explained in more detail in the next paragraphs. We will typically omit the word *labelled*, while talking about sequents, rules, etc. to mean the labelled versions and using *unlabelled* or *ordinary* otherwise. Names of labelled calculi are prefixed with **L** and names of labelled rules are prefixed with L.

[1] Multiple relations and/or n-ary relations may be used according to the Kripke semantics.

$$\mathsf{L}\mathsf{id}^* \frac{}{\mathcal{R}, i{:}\varphi, \Gamma \Rightarrow \Delta, i{:}\varphi} \qquad \mathsf{L}\wedge_l \frac{\mathcal{R}, i{:}\varphi, i{:}\psi, \Gamma \Rightarrow \Delta}{\mathcal{R}, i{:}\varphi \wedge \psi, \Gamma \Rightarrow \Delta} \qquad \mathsf{L}\to_l \frac{\mathcal{R}, \Gamma \Rightarrow \Delta, i{:}\varphi \quad \mathcal{R}, i{:}\psi, \Gamma \Rightarrow \Delta}{\mathcal{R}, i{:}\varphi \to \psi, \Gamma \Rightarrow \Delta}$$

Figure 19 Labelled versions of some propositional sequent rules from Figure 11.

$$\mathsf{L}\Box_l \frac{iRj, \mathcal{R}, j{:}\varphi, i{:}\Box\varphi, \Gamma \Rightarrow \Delta}{iRj, \mathcal{R}, i{:}\Box\varphi, \Gamma \Rightarrow \Delta} \qquad \mathsf{L}\Box_r \frac{iRj, \mathcal{R}, \Gamma \Rightarrow \Delta, j{:}\varphi}{\mathcal{R}, \Gamma \Rightarrow \Delta, i{:}\Box\varphi}\ (j \notin \mathcal{R}, \Gamma \Rightarrow \Delta)$$

Figure 20 Labelled sequent rules for the only modality $\Box$ of language $\mathcal{L}_\Box$.

$$\mathsf{L}B \frac{jRi, iRj, \mathcal{R}, \Gamma \Rightarrow \Delta}{iRj, \mathcal{R}, \Gamma \Rightarrow \Delta} \qquad \mathsf{L}4 \frac{iRk, iRj, jRk, \mathcal{R}, \Gamma \Rightarrow \Delta}{iRj, jRk, \mathcal{R}, \Gamma \Rightarrow \Delta} \qquad \mathsf{L}D \frac{iRj, \mathcal{R}, \Gamma \Rightarrow \Delta}{\mathcal{R}, \Gamma \Rightarrow \Delta}\ (j \notin \mathcal{R}, \Gamma \Rightarrow \Delta)$$

Figure 21 Labelled sequent rules for frame conditions of symmetry, transitivity, and seriality.

A labelled sequent calculus typically consists of three groups of rules: propositional rules, modal rules, and special rules. Propositional rules can be obtained by applying the following modification to each of the unlabelled rules from Figure 11 (or any other suitable calculus), other than cut.[2] For an unlabelled rule, add the same relational part $\mathcal{R}$ to the conclusion and all premises, treat multisets Γ, Δ, etc. as multisets of labelled formulas, and assign the same label to all principal and active formulas. The result for three of the rules from Figure 11 is shown in Figure 19. The modal rules require two rules (per modality) and reflect the Kripke semantics for the respective modality. The two rules for language $\mathcal{L}_\Box$ are shown in Figure 20. Special rules are determined on a logic-by-logic basis and can often be generated algorithmically based on the Kripke frame conditions for the logic [72]. For instance, the rules added if the frames are symmetric, transitive, and serial respectively are shown in Figure 21.

▶ **Definition 60.** *Let modal logic* L *be complete w.r.t. a class* $\mathcal{C}$ *of Kripke models. A sequent calculus* **LG** *is a* calculus for L (w.r.t. $\mathcal{C}$) *iff* $\mathbf{LG} \vdash S \iff \mathcal{C} \models S$ *for every sequent* S.

Labelled versions of the propositional rules from Figure 11, i.e., all rules but $(K)^*$, together with the rules from Figure 20, form the (cut-free) calculus **LG3K** (over $\mathcal{L}_\Box$) for modal logic K. Adding to **LG3K** any combination of ($\mathsf{L}B$), ($\mathsf{L}4$), and ($\mathsf{L}D$) from Figure 21 yields calculi for the logics complete w.r.t. the frames of the respective type. Note that some of these logics, e.g, B, possess no known cut-free *unlabelled* sequent calculus. Cf. [72] for general methods of constructing cut-free labelled sequent calculi.

[2] We omit the treatment of cut, as labelled sequent calculi are typically cut-free.

▶ **Proposition 61.** *If* **LG** *is a calculus for* L*, then* $\mathsf{L} \vdash \varphi \to \psi$ *iff* $\mathbf{LG} \vdash i{:}\varphi \Rightarrow i{:}\psi$ *for any* i*.*

5.2 Labelled Interpolation

As in the unlabelled case, an interpolant of a theorem $\varphi \to \psi$ is found based on a proof of a split sequent $i{:}\varphi;\ \Rightarrow\ ; i{:}\psi$ (for any label i). Splits are defined as in Definition 11 with relational atoms remaining outside of the split. The left (right) side of the split sequent $\mathcal{R}, \Gamma; \Gamma' \Rightarrow \Delta; \Delta'$ is $\mathcal{R}, \Gamma \Rightarrow \Delta$ (resp. $\mathcal{R}, \Gamma' \Rightarrow \Delta'$). Split rules are obtained as in the unlabelled case. For the modal rules from Figure 20, Γ and Δ are split the same way in the premise and conclusion, whereas both $i{:}\Box\varphi$ and $j{:}\varphi$ in the premise have to be on the same side of the split as $i{:}\Box\varphi$ in the conclusion.

Since labelled formulas are interpreted at different Kripke worlds according to a given interpretation $[\![\cdot]\!]$, the same should hold for interpolants, which cannot be mere formulas, but rather boolean combinations[3] of labelled formulas:

▶ **Definition 62** (Multiformula). *Grammar* $\mathfrak{V} ::= i{:}\varphi \mid (\mathfrak{V} \otimes \mathfrak{V}) \mid (\mathfrak{V} \oplus \mathfrak{V})$ *where* $\varphi \in \mathsf{Form}_{\mathcal{L}}$ *and* $i \in \mathbb{N}$ *defines* multiformulas *(over a language* $\mathcal{L}$*).* $i \in \mathfrak{V}$ *denotes that label* i *occurs in* $\mathfrak{V}$*.*

▶ **Definition 63** (Multiformula semantics). *Let* $\mathfrak{V}$ *be a multiformula and let* $[\![\cdot]\!] \colon X \to W$ *be an interpretation into* $\mathcal{M} = (W, R, V)$ *with* $X \supseteq \mathtt{L}(\mathfrak{V})$*. Then* $\mathcal{M}[\![\cdot]\!] \models i{:}\varphi$ *iff* $\mathcal{M}, [\![i]\!] \models \varphi$*, while* multiformula conjunction $\otimes$ *and* disjunction $\oplus$ *behave classically, i.e.,* $\mathcal{M}[\![\cdot]\!] \models \mathfrak{V}_1 \otimes \mathfrak{V}_2$ *(*$\mathcal{M}[\![\cdot]\!] \models \mathfrak{V}_1 \oplus \mathfrak{V}_2$*) iff* $\mathcal{M}[\![\cdot]\!] \models \mathfrak{V}_j$ *for both* $j = 1, 2$ *(resp. for at least one of* $j = 1, 2$*). We say that two multiformulas* $\mathfrak{V}$ *and* $\mathfrak{V}'$ *are* logically equivalent*, iff for any model* $\mathcal{M}$ *and any interpretation* $[\![\cdot]\!]$ *of* $\mathfrak{V}$ *and* $\mathfrak{V}'$ *into* $\mathcal{M}$*, we have* $\mathcal{M}[\![\cdot]\!] \models \mathfrak{V}$ *iff* $\mathcal{M}[\![\cdot]\!] \models \mathfrak{V}'$*.*

$\mathsf{Var}(\mathcal{R}, \Gamma \Rightarrow \Delta)$ and $\mathsf{Var}(\mathfrak{V})$ are defined by ignoring $\mathcal{R}$, as well as all the labels.

▶ **Definition 64** (Labelled Craig interpolation). *A calculus* **LG** *over a language* $\mathcal{L}$ *has the* Labelled Craig Interpolation Property (LCIP) *w.r.t. class* $\mathcal{C}$ *of Kripke models iff, whenever* $\vdash_{\mathbf{LG}} S$ *for a sequent* $S = \mathcal{R}, \Gamma, \Gamma' \Rightarrow \Delta, \Delta'$*, there exists a multiformula* $\mathfrak{V}$ *over* $\mathcal{L}$ *such that*

1. $\mathsf{Var}(\mathfrak{V}) \subseteq \mathsf{Var}(\Gamma, \Delta) \cap \mathsf{Var}(\Gamma', \Delta')$*, i.e., all atoms in* $\mathfrak{V}$ *occur in both sides of the split;*
2. *for any interpretation* $[\![\cdot]\!]$ *of* S *into a Kripke model* $\mathcal{M} \in \mathcal{C}$*:*
   - **a** *if* $\mathcal{M}[\![\cdot]\!] \not\models \mathfrak{V}$*, then* $\mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Gamma \Rightarrow \Delta$ *and*
   - **b** *if* $\mathcal{M}[\![\cdot]\!] \models \mathfrak{V}$*, then* $\mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Gamma' \Rightarrow \Delta'$*;*
3. $\mathtt{L}(\mathfrak{V}) \subseteq \mathtt{L}(\mathcal{R}, \Gamma, \Gamma' \Rightarrow \Delta, \Delta')$*, i.e., all labels in* $\mathfrak{V}$ *occur in the sequent.*

In this case, we write $\vdash_{\mathbf{LG}} \mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{V}}{\Longrightarrow} \Delta; \Delta'$ *and call* $\mathfrak{V}$ *the* (labelled Craig) interpolant *for the split sequent* $\mathcal{R}, \Gamma; \Gamma' \Rightarrow \Delta; \Delta'$ *in* **LG***.*

[3] Conjunctions and disjunctions are often sufficient with additional connectives added if need be.

Here conditions 1–2 are labelled analogs of the two conditions from Definition 12, with condition 2a being contrapositive relative to the first part of Definition 12(2). Condition 3 is labelled-specific (also used for nested and hypersequents). For the LIP, as in Definition 13:

▶ **Definition 65** (Labelled Lyndon interpolation). *The* Labelled Lyndon Interpolation Property (LLIP) *is obtained by replacing condition 1 in Definition 64 with:*
1′. $\mathsf{Var}^+(\mathcal{U}) \subseteq \mathsf{Var}^-(\Gamma \Rightarrow \Delta) \cap \mathsf{Var}^+(\Gamma' \Rightarrow \Delta')$ *and* $\mathsf{Var}^-(\mathcal{U}) \subseteq \mathsf{Var}^+(\Gamma \Rightarrow \Delta) \cap \mathsf{Var}^-(\Gamma' \Rightarrow \Delta')$.

▶ **Theorem 66.** *Let* $\mathbf{LG}$ *be a calculus for* L *(both over* $\mathcal{L}_\Box$*) w.r.t. a class* $\mathcal{C}$ *of Kripke models. If* $\mathbf{LG}$ *has the LCIP/LLIP, then* L *has the CIP/LIP.*

Proof sketch. If $\mathcal{U}$ is a labelled Craig/Lyndon interpolant for $1{:}\varphi;\ \Rightarrow\ ;1{:}\psi$ in $\mathbf{LG}$, then one can prove that formula $\mathsf{form}(\mathcal{U})$ is a Craig/Lyndon interpolant for $\varphi \to \psi$ in L, where $\mathsf{form}(\mathcal{U})$ is defined by $\mathsf{form}(i{:}\varphi) := \varphi$, $\mathsf{form}(\mathcal{U}_1 \circledcirc \mathcal{U}_2) := \mathsf{form}(\mathcal{U}_1) \circ \mathsf{form}(\mathcal{U}_2)$ for $\circ \in \{\wedge, \vee\}$. ◀

Now we turn to a reverse engineering strategy to see how to define interpolant transformations for various labelled rules. The form of the rules will guide us on how to come up with a construction of appropriate interpolant transformations.

For this new presentation we will divide labelled rules into different categories depending on the shape of their split rules. Different categories will require different interpolant transformations. When going through these explanations, the reader might notice many similarities with the interpolant constructions from Figure 7 for unlabelled rules. Indeed, many of the (implicit) principles in Figure 7 transfer to the labelled case with minor modifications. A fundamental difference is that interpolants in the labelled case are multiformulas (contain labels) instead of just formulas, which means that here we need the multiformula variants $\oslash$ and $\otimes$ of $\wedge$ and $\vee$ respectively. The reverse engineering strategy leads to the interpolant transformations for propositional labelled rules presented in Figure 22, which show close similarities with Figure 7 for unlabelled rules.

We first discuss three types of interpolant transformations that can be categorized via so-called *local*, *conjunctive*, and *disjunctive* split rules. These categories are defined precisely in [58], but are in essence defined by the three conditions mentioned in the following three lemmas. In particular, a *local split rule* is a single-premise rule that preserves the local logical consequence (if the premise is true for an interpretation, the conclusion must be true for the same interpretation), which leads to the interpolant of the conclusion being the same as the interpolant of the premise. For a conjunctive (resp. disjunctive) binary split rule the interpolant of the conclusion is the conjunction (resp. disjunction) of the interpolants of the premises.

▶ **Lemma 67** (Local split rules, [58]). *Let* $\mathcal{C}$ *be a class of Kripke models. Assume the following:*
(i) $\mathtt{L}(\mathcal{R}^*, \Gamma; \Gamma' \Rightarrow \Delta; \Delta') \subseteq \mathtt{L}(\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma')$*;*
(ii) $\mathsf{Var}(\Gamma, \Delta) \subseteq \mathsf{Var}(\Pi, \Sigma)$ *and* $\mathsf{Var}(\Gamma', \Delta') \subseteq \mathsf{Var}(\Pi', \Sigma')$*;*
(iii) *For all* $\mathcal{M} \in \mathcal{C}$ *and interpretations* $[\![\cdot]\!]$ *of* $\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma'$ *into* $\mathcal{M}$*:*

$$\mathcal{M}[\![\cdot]\!] \models \mathcal{R}^*, \Gamma \Rightarrow \Delta \quad \Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi \Rightarrow \Sigma, \text{ and}$$
$$\mathcal{M}[\![\cdot]\!] \models \mathcal{R}^*, \Gamma' \Rightarrow \Delta' \quad \Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi' \Rightarrow \Sigma'.$$

Interpolants for the four splits of labelled axiom (Lid^*)

$$LR(\mathsf{Lid}^*)\ \frac{}{\mathcal{R}, i{:}\varphi, \Gamma; \Gamma' \overset{i:\varphi}{\Longrightarrow} \Delta; \Delta', i{:}\varphi} \qquad RR(\mathsf{Lid}^*)\ \frac{}{\mathcal{R}, \Gamma; \Gamma', i{:}\varphi \overset{i:\top}{\Longrightarrow} \Delta; \Delta', i{:}\varphi}$$

$$LL(\mathsf{Lid}^*)\ \frac{}{\mathcal{R}, i{:}\varphi, \Gamma; \Gamma' \overset{i:\bot}{\Longrightarrow} i{:}\varphi, \Delta; \Delta'} \qquad RL(\mathsf{Lid}^*)\ \frac{}{\mathcal{R}, \Gamma; \Gamma', i{:}\varphi \overset{i:\neg\varphi}{\Longrightarrow} \Delta, i{:}\varphi; \Delta'}$$

Interpolant transformations for the two splits of unary labelled rule ($\mathsf{L}\!\wedge_l$)

$$L(\mathsf{L}\!\wedge_l)\ \frac{\mathcal{R}, i{:}\varphi, i{:}\psi, \Gamma; \Gamma' \overset{\mathfrak{O}}{\Longrightarrow} \Delta; \Delta'}{\mathcal{R}, i{:}\varphi \wedge \psi, \Gamma; \Gamma' \overset{\mathfrak{O}}{\Longrightarrow} \Delta; \Delta'} \qquad R(\mathsf{L}\!\wedge_l)\ \frac{\mathcal{R}, \Gamma; \Gamma', i{:}\varphi, i{:}\psi \overset{\mathfrak{O}}{\Longrightarrow} \Delta; \Delta'}{\mathcal{R}, \Gamma; \Gamma', i{:}\varphi \wedge \psi \overset{\mathfrak{O}}{\Longrightarrow} \Delta; \Delta'}$$

Interpolant transformations for the two splits of binary labelled rule ($\mathsf{L}\!\rightarrow_l$)

$$L(\mathsf{L}\!\rightarrow_l)\ \frac{\mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{O}_1}{\Longrightarrow} i{:}\varphi, \Delta; \Delta' \quad \mathcal{R}, i{:}\psi, \Gamma; \Gamma' \overset{\mathfrak{O}_2}{\Longrightarrow} \Delta; \Delta'}{\mathcal{R}, i{:}\varphi \to \psi, \Gamma; \Gamma' \overset{\mathfrak{O}_1 \mathbin{@} \mathfrak{O}_2}{\Longrightarrow} \Delta; \Delta'} \qquad R(\mathsf{L}\!\rightarrow_l)\ \frac{\mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{O}_1}{\Longrightarrow} \Delta; \Delta', i{:}\varphi \quad \mathcal{R}, \Gamma; \Gamma', i{:}\psi \overset{\mathfrak{O}_2}{\Longrightarrow} \Delta; \Delta'}{\mathcal{R}, \Gamma; \Gamma', i{:}\varphi \to \psi \overset{\mathfrak{O}_1 \mathbin{@} \mathfrak{O}_2}{\Longrightarrow} \Delta; \Delta'}$$

Figure 22 Craig/Lyndon interpolant construction for sample split rules obtained from Figure 11. Capital L (R) denotes that the principal formula is on the left (resp. right) side of the split.

Then the following interpolant transformation works for the LCIP:

$$\frac{\mathcal{R}^*, \Gamma; \Gamma' \overset{\mathfrak{O}}{\Longrightarrow} \Delta; \Delta'}{\mathcal{R}, \Pi; \Pi' \overset{\mathfrak{O}}{\Longrightarrow} \Sigma; \Sigma'}\ ,$$

i.e., if $\mathfrak{O}$ is a labelled Craig interpolant for $\mathcal{R}^*, \Gamma; \Gamma' \Rightarrow \Delta; \Delta'$, then $\mathfrak{O}$ is also a labelled Craig interpolant for $\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma'$.

▶ **Lemma 68** (Conjunctive split rules, [58]). *Let $\mathcal{C}$ be a class of Kripke models. Assume:*

(i) $\mathsf{L}(\mathcal{R}_a, \Gamma_a; \Gamma'_a \Rightarrow \Delta_a; \Delta'_a) \subseteq \mathsf{L}(\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma')$ *for both* $a \in \{1, 2\}$*;*

(ii) $\mathsf{Var}(\Gamma_a, \Delta_a) \subseteq \mathsf{Var}(\Pi, \Sigma)$ *and* $\mathsf{Var}(\Gamma'_a, \Delta'_a) \subseteq \mathsf{Var}(\Pi', \Sigma')$ *for both* $a \in \{1, 2\}$*;*

(iii) *For all $\mathcal{M} \in \mathcal{C}$ and interpretations $[\![\cdot]\!]$ of $\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma'$ into $\mathcal{M}$:*

$$\begin{aligned} \mathcal{M}[\![\cdot]\!] \models \mathcal{R}_1, \Gamma_1 \Rightarrow \Delta_1 \quad &\Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi \Rightarrow \Sigma, \\ \mathcal{M}[\![\cdot]\!] \models \mathcal{R}_2, \Gamma_2 \Rightarrow \Delta_2 \quad &\Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi \Rightarrow \Sigma, \ \textit{and} \\ \mathcal{M}[\![\cdot]\!] \models \mathcal{R}_1, \Gamma'_1 \Rightarrow \Delta'_1\ ,\ \mathcal{M}[\![\cdot]\!] \models \mathcal{R}_2, \Gamma'_2 \Rightarrow \Delta'_2 \quad &\Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi' \Rightarrow \Sigma'. \end{aligned}$$

Then the following interpolant transformation works for the LCIP:

$$\frac{\mathcal{R}_1, \Gamma_1; \Gamma'_1 \overset{\mathfrak{O}_1}{\Longrightarrow} \Delta_1; \Delta'_1 \quad \mathcal{R}_2, \Gamma_2; \Gamma'_2 \overset{\mathfrak{O}_2}{\Longrightarrow} \Delta_2; \Delta'_2}{\mathcal{R}, \Pi; \Pi' \overset{\mathfrak{O}_1 \mathbin{@} \mathfrak{O}_2}{\Longrightarrow} \Sigma; \Sigma'}\ .$$

▶ **Lemma 69** (Disjunctive split rules, [58]). *Let $\mathcal{C}$ be a class of Kripke models. Assume* (i) *and* (ii) *from Lemma 68 hold and assume:*
(iii) *For all $\mathcal{M} \in \mathcal{C}$ and interpretations $[\![\cdot]\!]$ of $\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma'$ into $\mathcal{M}$:*

$$\begin{aligned} \mathcal{M}[\![\cdot]\!] \models \mathcal{R}_1, \Gamma_1 \Rightarrow \Delta_1 \ , \ \mathcal{M}[\![\cdot]\!] \models \mathcal{R}_2, \Gamma_2 \Rightarrow \Delta_2 \quad &\Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi \Rightarrow \Sigma, \\ \mathcal{M}[\![\cdot]\!] \models \mathcal{R}_1, \Gamma_1' \Rightarrow \Delta_1' \quad &\Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi' \Rightarrow \Sigma', \ \textit{and} \\ \mathcal{M}[\![\cdot]\!] \models \mathcal{R}_2, \Gamma_2' \Rightarrow \Delta_2' \quad &\Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi' \Rightarrow \Sigma'. \end{aligned}$$

Then the following interpolant transformation works for the LCIP:

$$\frac{\mathcal{R}_1, \Gamma_1; \Gamma_1' \overset{\mho_1}{\Longrightarrow} \Delta_1; \Delta_1' \quad \mathcal{R}_2, \Gamma_2; \Gamma_2' \overset{\mho_2}{\Longrightarrow} \Delta_2; \Delta_2'}{\mathcal{R}, \Pi; \Pi' \overset{\mho_1 \otimes \mho_2}{\Longrightarrow} \Sigma; \Sigma'} \ .$$

The transformations in Lemmas 67–69 also work for the LLIP if assumptions (ii) are strengthened to satisfy Definition 65(1′), taking polarity into account (for details see [58]).

Before we provide examples of labelled local, conjunctive, and disjunctive split rules, we note that these concepts can already be observed in the unlabelled case.

▶ **Example 70** (The unlabelled case). Recall from Remark 59 that an ordinary sequent can be viewed as a labelled sequent with a single label. Applying the conditions of Lemmas 67–69 to rules in Figure 7, we see that $R(\wedge_l^i)$, $R(\rightarrow_r)$, and $L(\mathsf{c}_r)$ are local split rules, $L(\vee_l)$, $L(\wedge_r)$, and $L(\rightarrow_l)$ are disjunctive split rules, while $R(\vee_l)$, $R(\wedge_r)$, and $R(\rightarrow_l)$ are conjunctive split rules. Note the symmetry that often occurs for the Left/Right splits of a rule vs. disjunctive/conjunctive behavior of the interpolant construction. A similar pattern can be seen in the interpolant transformation for the analytic cut rule in the proof of Theorem 23.

For modal unlabelled rules, consider Figure 5. Any split of rule (T) is local w.r.t. any class $\mathcal{C}$ of reflexive models: if φ is false in a reflexive world, then so is $\Box\varphi$. This explains the interpolant preservation for $L(T)$ in Figure 8. By contrast, the other rules in Figure 5 are not local, hence, require non-trivial transformations in Figure 8. ⌟

Let us now consider the labelled analog of Example 70.

▶ **Example 71** (The labelled case). For the propositional rules in Figure 22, we can identify local, conjunctive, and disjunctive split rules according to essentially the same argument as in Example 70. For modal labelled rules, the argument is different from the unlabelled case. The standard split version of $(\mathsf{L}\Box_l)$ from Figure 20 is local (w.r.t. any class) because $\mathcal{M}, [\![j]\!] \not\models \varphi$ and $[\![i]\!]R[\![j]\!]$ imply $\mathcal{M}, [\![i]\!] \not\models \Box\varphi$. Consequently, the following interpolant transformations work for the LCIP/LLIP w.r.t. any class of models:

$$L(\mathsf{L}\Box_l)\ \frac{iRj, \mathcal{R}, j{:}\varphi, i{:}\Box\varphi, \Gamma; \Gamma' \overset{\mho}{\Longrightarrow} \Delta; \Delta'}{iRj, \mathcal{R}, i{:}\Box\varphi, \Gamma; \Gamma' \overset{\mho}{\Longrightarrow} \Delta; \Delta'} \qquad R(\mathsf{L}\Box_l)\ \frac{iRj, \mathcal{R}, \Gamma; \Gamma', i{:}\Box\varphi, j{:}\varphi \overset{\mho}{\Longrightarrow} \Delta; \Delta'}{iRj, \mathcal{R}, \Gamma; \Gamma', i{:}\Box\varphi \overset{\mho}{\Longrightarrow} \Delta; \Delta'}$$

However, the left and right split versions of ($\mathsf{L}\Box_r$) cannot be identified as local, conjunctive, or disjunctive and need different treatments as discussed in Example 80. ⌟

In addition, many of the unary special labelled rules are local, hence, require no change to the interpolant. A large class of special rules is obtained by so-called Horn clauses. Restricting a class of frames by a Horn clause of the form $i_1Rj_1 \wedge \cdots \wedge i_mRj_m \to kRl$ corresponds to adding to the labelled calculus the rule

$$\frac{kRl, i_1Rj_1, \ldots, i_mRj_m, \mathcal{R}, \Gamma \Rightarrow \Delta}{i_1Rj_1, \ldots, i_mRj_m, \mathcal{R}, \Gamma \Rightarrow \Delta}\,(k, l \in i_1Rj_1, \ldots, i_mRj_m, \mathcal{R}, \Gamma \Rightarrow \Delta) \tag{2}$$

along with the rules obtained from it by the *closure condition*, i.e., by contracting identical relational atoms i_aRj_a and i_bRj_b from the conclusion in both premise and conclusion for those instances of the rule that contain such identical atoms [72, Th. 11.27, Cor. 11.29]. Note that in the absence of a principal formula, such rules only have one corresponding split rule (instead of a left and right split version).

▶ **Example 72** (Horn frame conditions, [57])**.** For any split of (2) or its contractions, the following interpolant transformation works for the LCIP/LLIP w.r.t. any class of models satisfying the condition $i_1Rj_1 \wedge \cdots \wedge i_mRj_m \to kRl$:

$$\frac{kRl, i_1Rj_1, \ldots, i_mRj_m, \mathcal{R}, \Gamma; \Gamma' \overset{\mho}{\Longrightarrow} \Delta; \Delta'}{i_1Rj_1, \ldots, i_mRj_m, \mathcal{R}, \Gamma; \Gamma' \overset{\mho}{\Longrightarrow} \Delta; \Delta'}$$

Consequently, the split version of rule ($\mathsf{L}B$) for any class of symmetric frames and of rule ($\mathsf{L}4$) for any class of transitive frames from Figure 21 are local and, therefore, the above interpolant transformation works for the LCIP/LLIP. ⌟

Two rules from Figures 20 and 21 do not fit any of the patterns described so far: the premise of both ($\mathsf{L}\Box_r$) and ($\mathsf{L}D$) has a fresh label not occurring in the conclusion. Such rules require a more complex transformation to remove the extra label from premise interpolants. For that, we introduce additional notation for modifying interpretations.

▶ **Definition 73.** *Let $\mathcal{M} = (W, R, V)$ be a Kripke model and $[\![\cdot]\!] \colon X \to W$ map labels from $X \subseteq \mathbb{N}$ to worlds of $\mathcal{M}$. For $v \in W$ and $j \in \mathbb{N}$, we define $[\![\cdot]\!]_j^v := \big([\![\cdot]\!] \setminus (\{j\} \times W)\big) \sqcup \{(j, v)\}$ where $\sqcup$ is used to emphasize that this is the union of disjoint sets.*

The domain of $[\![\cdot]\!]_j^v$ is $X \cup \{j\}$. Note that the only possible difference between $[\![\cdot]\!]_j^v$ and $[\![\cdot]\!]$ is that the former maps j to v. Note also that if $j \notin \mathsf{L}(S)$ for a labelled sequent S, then $[\![\cdot]\!]$ is an interpretation for S iff $[\![\cdot]\!]_j^v$ is and, if both are, $\mathcal{M}[\![\cdot]\!] \models S$ iff $\mathcal{M}[\![\cdot]\!]_j^v \models S$ for any $v \in W$. The same applies to multiformulas.

▶ **Definition 74.** *Let $\mathcal{M} = (W, R, V)$ be a Kripke model and $[\![\cdot]\!]\colon X \to W$ be an interpretation of a sequent S into $\mathcal{M}$. For $i \in X$ and $j \neq i$, we write*

$$\mathcal{M}[\![\cdot]\!]_j^{iR\forall} \models S \qquad \text{whenever} \qquad \forall v \in W \text{ such that } [\![i]\!]Rv, \text{ we have } \mathcal{M}[\![\cdot]\!]_j^{v} \models S$$
$$\mathcal{M}[\![\cdot]\!]_j^{iR\exists} \models S \qquad \text{whenever} \qquad \exists v \in W \text{ such that } [\![i]\!]Rv \text{ and } \mathcal{M}[\![\cdot]\!]_j^{v} \models S.$$

▶ **Definition 75.** *A label j is* $\parr\otimes$-separated *(resp.* $\otimes\parr$-separated*) in a multiformula $\mathfrak{U}$ iff for some formulas $B_1, \dots, B_m$ and multiformulas $\mathfrak{U}_1, \dots, \mathfrak{U}_m$ with $j \notin \mathtt{L}(\mathfrak{U}_k)$ for any $1 \leq k \leq m$, we have $\mathfrak{U} = \bigparr_{k=1}^{m}(j{:}B_k \otimes \mathfrak{U}_k)$ (resp. $\mathfrak{U} = \bigotimes_{k=1}^{m}(j{:}B_k \parr \mathfrak{U}_k)$).*

▶ **Proposition 76.** *Let $j \in \mathbb{N}$. For any multiformula $\mathfrak{U}$, there is a logically equivalent multiformula $\mathfrak{U}'$ such that j is $\parr\otimes$-separated (resp. $\otimes\parr$-separated) in $\mathfrak{U}'$.*

Proof. $\mathfrak{U}'$ can be constructed by rewritings similar to the ones used for CNF (resp. DNF), in view of logical equivalences of $j{:}\varphi \parr j{:}\psi$ to $j{:}\varphi \wedge \psi$ and of $j{:}\varphi \otimes j{:}\psi$ to $j{:}\varphi \vee \psi$. If $j{:}B_k$ is absent for some k, either $j{:}\top\parr$ or $j{:}\bot\otimes$ is added, whichever is logically equivalent. ◀

Proposition 76 does not guarantee the uniqueness of $\mathfrak{U}'$, but given $\mathfrak{U}$ we know there is at least one logically equivalent multiformula $\mathfrak{U}'$ with j $\parr\otimes$-separated (resp. $\otimes\parr$-separated) in it. From now on, given a multiformula $\mathfrak{U}$, we allow ourselves to pick one such $\mathfrak{U}'$ and denote it by $\mathfrak{U}_{\parr j\otimes}$ (resp. $\mathfrak{U}_{\otimes j\parr}$). For a multiformula $\mathfrak{U}$, we further write $\mathfrak{U}^{j\mapsto i\Box}$ (resp. $\mathfrak{U}^{j\mapsto i\Diamond}$) to mean the result of replacing each occurrence of $j{:}$ in $\mathfrak{U}$ with $i{:}\Box$ (resp. with $i{:}\Diamond$).

▶ **Example 77.** Suppose $\mathfrak{U}_{\parr j\otimes} = \bigparr_{k=1}^{m}(j{:}B_k \otimes \mathfrak{U}_k)$ (resp. $\mathfrak{U}_{\otimes j\parr} = \bigotimes_{k=1}^{m}(j{:}B_k \parr \mathfrak{U}_k)$) with $j \notin \mathtt{L}(\mathfrak{U}_k)$ for any $1 \leq k \leq m$. Then,

$$\mathfrak{U}^{j\mapsto i\Box}_{\parr j\otimes} = \bigparr_{k=1}^{m}(i{:}\Box B_k \otimes \mathfrak{U}_k) \qquad \big(\text{respectively} \qquad \mathfrak{U}^{j\mapsto i\Diamond}_{\otimes j\parr} = \bigotimes_{k=1}^{m}(i{:}\Diamond B_k \parr \mathfrak{U}_k)\big).$$

Clearly, $j \notin \mathtt{L}(\mathfrak{U}^{j\mapsto i\Box}_{\parr j\otimes})$ (resp. $j \notin \mathtt{L}(\mathfrak{U}^{j\mapsto i\Diamond}_{\otimes j\parr})$). ⌟

The general transformation applicable to rules ($\mathtt{L}\Box_r$) and ($\mathtt{L}D$) additionally exploits the fact that the fresh label j is only connected to one existing label i by one relational atom iRj.

▶ **Lemma 78** (□-like rules, [58])**.** *Let $\mathcal{C}$ be a class of Kripke models. Assume:*

(i) $\mathsf{Var}(\Gamma, \Delta) \subseteq \mathsf{Var}(\Pi, \Sigma)$ *and* $\mathsf{Var}(\Gamma', \Delta') \subseteq \mathsf{Var}(\Pi', \Sigma')$,
(ii) $j \notin \mathtt{L}(\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma')$ *and* $\{i\} \cup \mathtt{L}(\mathcal{R}^*) \subseteq \mathtt{L}(\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma')$,
(iii) $\mathtt{L}(\Gamma; \Gamma' \Rightarrow \Delta; \Delta') \subseteq \{j\} \sqcup \mathtt{L}(\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma')$,
(iv) *For all $\mathcal{M} \in \mathcal{C}$ and interpretations $[\![\cdot]\!]$ of $\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma'$ into $\mathcal{M}$:*

$$\mathcal{M}[\![\cdot]\!]_j^{iR\exists} \models iRj, \mathcal{R}^*, \Gamma \Rightarrow \Delta \quad \Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi \Rightarrow \Sigma, \text{ and}$$
$$\mathcal{M}[\![\cdot]\!]_j^{iR\forall} \models iRj, \mathcal{R}^*, \Gamma' \Rightarrow \Delta' \quad \Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi' \Rightarrow \Sigma'.$$

Then the following interpolant transformation works for the LCIP:

$$\frac{iRj, \mathcal{R}^*, \Gamma; \Gamma' \overset{\mathfrak{O}}{\Longrightarrow} \Delta; \Delta'}{\mathcal{R}, \Pi; \Pi' \overset{\mathfrak{O}^{j\mapsto \Box i}_{\mathbb{O}j\mathbb{O}}}{\Longrightarrow} \Sigma; \Sigma'}.$$

▶ **Lemma 79** (◇-like rules, [58]). *Let $\mathcal{C}$ be a class of Kripke models. Assume* (i),(ii), *and* (iii) *from Lemma 78 hold. Moreover, assume:*

(iv) *For all $\mathcal{M} \in \mathcal{C}$ and interpretations $[\![\cdot]\!]$ of $\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma'$ into $\mathcal{M}$:*

$$\mathcal{M}[\![\cdot]\!]^{iR\forall}_j \models iRj, \mathcal{R}^*, \Gamma \Rightarrow \Delta \quad \Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi \Rightarrow \Sigma, \text{ and}$$
$$\mathcal{M}[\![\cdot]\!]^{iR\exists}_j \models iRj, \mathcal{R}^*, \Gamma' \Rightarrow \Delta' \quad \Longrightarrow \quad \mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi' \Rightarrow \Sigma'.$$

Then the following interpolant transformation works for the LCIP:

$$\frac{iRj, \mathcal{R}^*, \Gamma; \Gamma' \overset{\mathfrak{O}}{\Longrightarrow} \Delta; \Delta'}{\mathcal{R}, \Pi; \Pi' \overset{\mathfrak{O}^{j\mapsto \Diamond i}_{\mathbb{O}j\mathbb{O}}}{\Longrightarrow} \Sigma; \Sigma'}.$$

The transformations in Lemmas 78 and 79 also work for the LLIP if assumption (i) is strengthened to satisfy Definition 65(1′), taking polarity into account (for details see [58]).

Proof. We treat Lemma 79 (Lemma 78 is dual). Here we only prove property 2 from Definition 64. Let $[\![\cdot]\!]$ be an interpretation for $\mathcal{R}, \Pi; \Pi' \Rightarrow \Sigma; \Sigma'$. Suppose that $\mathcal{M}[\![\cdot]\!] \not\models \mathfrak{O}^{j\mapsto \Diamond i}_{\mathbb{O}j\mathbb{O}}$, where multiformulas $\mathfrak{O}_{\mathbb{O}j\mathbb{O}}$ and $\mathfrak{O}^{j\mapsto \Diamond i}_{\mathbb{O}j\mathbb{O}}$ are as in Example 77. Then, for each $1 \leq k \leq m$, either $\mathcal{M}, [\![i]\!] \not\models \Diamond B_k$ or $\mathcal{M}[\![\cdot]\!] \not\models \mathfrak{O}_k$. Thus, for all v such that $[\![i]\!]Rv$ either $\mathcal{M}, v \not\models B_k$ or $\mathcal{M}[\![\cdot]\!]^v_j \not\models \mathfrak{O}_k$ (since $j \notin \mathtt{L}(\mathfrak{O}_k)$). Consequently, $\mathcal{M}[\![\cdot]\!]^v_j \not\models \mathfrak{O}_{\mathbb{O}j\mathbb{O}}$, i.e., $\mathcal{M}[\![\cdot]\!]^v_j \not\models \mathfrak{O}$ for all v such that $[\![i]\!]Rv$. Given that $\mathfrak{O}$ is an interpolant of the premise, $\mathcal{M}[\![\cdot]\!]^{iR\forall}_j \models iRj, \mathcal{R}^*, \Gamma \Rightarrow \Delta$, which yields $\mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi \Rightarrow \Sigma$ by the first assumed validity.

Suppose now that that $\mathcal{M}[\![\cdot]\!] \models \mathfrak{O}^{j\mapsto \Diamond i}_{\mathbb{O}j\mathbb{O}}$. Then, there is some $1 \leq k \leq m$ such that $\mathcal{M}, [\![i]\!] \models \Diamond B_k$ and $\mathcal{M}[\![\cdot]\!] \models \mathfrak{O}_k$. From the former, we find a v such that $[\![i]\!]Rv$, and $\mathcal{M}, v \models B_k$, and $\mathcal{M}[\![\cdot]\!]^v_j \models \mathfrak{O}_k$. Consequently, $\mathcal{M}[\![\cdot]\!]^v_j \models \mathfrak{O}_{\mathbb{O}j\mathbb{O}}$, i.e., $\mathcal{M}[\![\cdot]\!]^v_j \models \mathfrak{O}$ for this v. Given that $\mathfrak{O}$ is an interpolant of the premise, we get $\mathcal{M}[\![\cdot]\!]^{iR\exists}_j \models iRj, \mathcal{R}^*, \Gamma' \Rightarrow \Delta'$, which yields the requisite $\mathcal{M}[\![\cdot]\!] \models \mathcal{R}, \Pi' \Rightarrow \Sigma'$ by the second assumed validity. ◀

▶ **Example 80** (□- and ◇-like rules). The split version of $(\mathsf{L}\Box_r)$ is □-like (resp. ◇-like) w.r.t. any class of models if $\Box\varphi$ is on the right (resp. left) side of the split. Consequently, the following interpolant transformations work for the LCIP/LLIP w.r.t. any class of models:

$$L(\mathsf{L}\Box_r)\ \frac{iRj, \mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{O}}{\Longrightarrow} j{:}\varphi, \Delta; \Delta'}{\mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{O}^{j\mapsto \Diamond i}_{\mathbb{O}j\mathbb{O}}}{\Longrightarrow} i{:}\Box\varphi, \Delta; \Delta'} \qquad R(\mathsf{L}\Box_r)\ \frac{iRj, \mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{O}}{\Longrightarrow} \Delta; \Delta', j{:}\varphi}{\mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{O}^{j\mapsto \Box i}_{\mathbb{O}j\mathbb{O}}}{\Longrightarrow} \Delta; \Delta', i{:}\Box\varphi}$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{3R2, 2R3, 1R2, 2{:}q, 3{:}p, 3{:}\Box q, 2{:}\Box(p \wedge \Box q), 1{:}\Box\Box(p \wedge \Box q);\ \overset{2:q}{\Longrightarrow}\ ; 2{:}q, 2{:}r}\,LR(\mathsf{Lid}^*)}{3R2, 2R3, 1R2, 2{:}q, 3{:}p, 3{:}\Box q, 2{:}\Box(p \wedge \Box q), 1{:}\Box\Box(p \wedge \Box q);\ \overset{2:q}{\Longrightarrow}\ ; 2{:}(q \vee r)}\,R(\mathsf{L}\vee_r)}{3R2, 2R3, 1R2, 2{:}q, 3{:}p, 3{:}\Box q, 2{:}\Box(p \wedge \Box q), 1{:}\Box\Box(p \wedge \Box q);\ \overset{2:q}{\Longrightarrow}\ ; 2{:}(q \vee r)}\,L(\mathsf{L}\Box_l)}{3R2, 2R3, 1R2, 3{:}p, 3{:}\Box q, 2{:}\Box(p \wedge \Box q), 1{:}\Box\Box(p \wedge \Box q);\ \overset{2:q}{\Longrightarrow}\ ; 2{:}(q \vee r)}\,\mathsf{L}B}{2R3, 1R2, 3{:}p, 3{:}\Box q, 2{:}\Box(p \wedge \Box q), 1{:}\Box\Box(p \wedge \Box q);\ \overset{2:q}{\Longrightarrow}\ ; 2{:}(q \vee r)}\,L(\mathsf{L}\wedge_l)}{2R3, 1R2, 3{:}(p \wedge \Box q), 2{:}\Box(p \wedge \Box q), 1{:}\Box\Box(p \wedge \Box q);\ \overset{2:q}{\Longrightarrow}\ ; 2{:}(q \vee r)}\,L(\mathsf{L}\Box_l)}{2R3, 1R2, 2{:}\Box(p \wedge \Box q), 1{:}\Box\Box(p \wedge \Box q);\ \overset{2:q \equiv 3:\top \otimes 2:q}{\Longrightarrow}\ ; 2{:}(q \vee r)}\,\mathsf{L}D}{1R2, 2{:}\Box(p \wedge \Box q), 1{:}\Box\Box(p \wedge \Box q);\ \overset{2:\Diamond\top \otimes 2:q}{\Longrightarrow}\ ; 2{:}(q \vee r)}\,L(\mathsf{L}\Box_l)}{1R2, 1{:}\Box\Box(p \wedge \Box q);\ \overset{2:\Diamond\top \otimes 2:q \equiv 2:q \otimes 1:\bot}{\Longrightarrow}\ ; 2{:}(q \vee r)}\,R(\mathsf{L}\Box_r)}{1{:}\Box\Box(p \wedge \Box q);\ \overset{1:\Box q \otimes 1:\bot}{\Longrightarrow}\ ; 1{:}\Box(q \vee r)}
$$

Figure 23 Labelled interpolant computation in Example 81.

where i does and j does not occur in the conclusion. Rule ($\mathsf{L}D$) is interesting because there is no separation into the right and left split variants. Its only split version is both $\Box$-like and $\Diamond$-like as long as the frame is serial. (The seriality is needed to avoid the vacuous fulfillment of the $\mathcal{M}[\![\cdot]\!]_j^{iR\forall} \models$ parts in Lemmas 78 and 79.) Thus, for i occurring and j not occurring in the conclusion, w.r.t. any class $\mathcal{C}$ of serial frames, both transformations are valid:

$$
(\mathsf{L}D)\ \dfrac{iRj, \mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{V}}{\Longrightarrow} \Delta; \Delta'}{\mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{V}^{j \mapsto \Diamond i}_{\otimes j \otimes}}{\Longrightarrow} \Delta; \Delta'} \qquad (\mathsf{L}D)\ \dfrac{iRj, \mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{V}}{\Longrightarrow} \Delta; \Delta'}{\mathcal{R}, \Gamma; \Gamma' \overset{\mathfrak{V}^{j \mapsto \Box i}_{\otimes j \otimes}}{\Longrightarrow} \Delta; \Delta'}\ .
$$

Note that the $\Box\theta$ and $\Diamond\theta$ transformations in Figure 8 are the unlabelled manifestations of $\Box$- and $\Diamond$-like rules. ⌟

▶ **Example 81.** Let $\varphi = \Box\Box(p \wedge \Box q)$ and $\psi = \Box(q \vee r)$. In Figure 23, we find an interpolant of $\varphi \to \psi$ using the split labelled calculus for logic DB (the logic of symmetric serial frames), which has no cut-free sequent calculus. Thus, $\mathsf{form}(1{:}\Box q \otimes 1{:}\bot) = \Box q \vee \bot$, or equivalently $\Box q$, is a Lyndon interpolant of $\varphi \to \psi = \Box\Box(p \wedge \Box q) \to \Box(q \vee r)$ in logic DB. Here rule ($\mathsf{L}D$) was treated as a $\Diamond$-like, with label 3 being $\otimes\oslash$-separated in $3{:}\top \oslash 2{:}q \equiv 2{:}q$. Similarly, label 2 is $\oslash\otimes$-separated in $2{:}q \otimes 1{:}\bot \equiv 2{:}\Diamond\top \oslash 2{:}q$. (For this logic, $\Diamond\top$ may be dropped due to seriality. In, e.g., K, one would use $2{:}(\Diamond\top \wedge q) \otimes 1{:}\bot$ instead.) ⌟

In the next section we see how these techniques lead to powerful interpolation results.

5.3 Interpolation Results for Labelled, Hyper-, and Nested Sequents

The general approach explained in the previous section of defining interpolants by looking at the shape of labelled split rules has two advantages. First, it leads to interpolation results (including Lyndon interpolation) for a range of logics. Second, the techniques can be adjusted to other types of sequents such as nested sequents and hypersequents. In this section we discuss results in both directions and point to relevant literature for more details.

Composing the algorithm for generating cut-free labelled sequent calculi rules [72] with the algorithm discussed in Section 5.2 for constructing interpolant transformations can yield general theorems as follows:

▶ **Theorem 82** ([58]). *The modal logic* $\mathsf{L}_{\mathcal{C}}$ *of any class* $\mathcal{C}$ *of frames defined by a set of frame conditions in the Horn-clause form* $i_1Rj_1 \wedge \cdots \wedge i_mRj_m \to kRl$ *has the LIP.*

▶ **Corollary 83.** *Of the frame conditions listed in [31, Sect. 8], any combination of seriality, reflexivity, transitivity, symmetry, euclideanity, and shift reflexivity produces a logic with the LIP. This includes all 15 modal logics of the so-called modal cube [31, Sect. 8], among them* K*,* T*,* D*,* K4*,* S4*, and* S5 *from Figure 3.*

The range of Kripke-frame classes guaranteeing the LIP can be expanded further, in particular to deal with the convergence restriction [31, Sect. 8] and *hijk*-convergence [31, Sect. 9], yielding logics often named after Scott–Lemon or Geach. For details we refer to [58].

With Corollary 83 we can illustrate that labelled sequent calculi can be more suitable to obtain Lyndon interpolation than ordinary sequent calculi with some form of cut. For example, in Theorem 23 we used Maehara's method to prove Craig interpolation for logic S5 by using the analytic cut rule. However, analytic cuts do not preserve polarity, resulting in the inability to obtain Lyndon interpolation. Corollary 83, by contrast, establishes the LIP for S5 using the techniques for labelled sequents.

Now we turn to interpolation results via hypersequents and nested sequents. Nested sequents and hypersequents are closely related to labelled sequents. A labelled sequent $S = \mathcal{R}, \Gamma \Rightarrow \Delta$ can be viewed as consisting of ordinary, unlabelled sequents $\{\Gamma_i \Rightarrow \Delta_i\}_{i \in \mathsf{L}(S)}$ where multisets $\Gamma_i := \{\varphi \mid i{:}\varphi \in \Gamma\}$ and $\Delta_i := \{\varphi \mid i{:}\varphi \in \Delta\}$ contain (unlabelled) formulas. In this view, each sequent $\Gamma_i \Rightarrow \Delta_i$ is a *sequent component of* S and relational atoms from $\mathcal{R}$ describe the relationships among these sequent components. Relational atoms are expressive enough to represent any (Kripke-like) relationship. If a particular relationship is regular enough, it can be encoded by structural connectives, bypassing the use of explicit labels.

Hypersequents A *hypersequent* is defined as a finite multiset (alternatively, a set or a list) of sequent components, commonly written as $\Gamma_1 \Rightarrow \Delta_1 \mid \cdots \mid \Gamma_n \Rightarrow \Delta_n$. For example, modal logic S5 admits a hypersequent calculus where hypersequents can be seen as labelled sequents in which $\mathcal{R}$ describes a total relation (e.g., $iRj \in \mathcal{R}$ for all $1 \leq i, j \leq n$). Although hypersequents predate labelled sequents, having been introduced in [9, 69, 78], the labelled

view on hypersequents means that any interpolant transformation from Section 5.2 can be used whenever its conditions are satisfied. To use the interpolation algorithm, labels have to be assigned to each sequent component of each hypersequent in a proof. One may need to be careful when assigning labels, depending on whether a rule modifies a sequent component or replaces it with a different component. The latter case calls for using a different label, e.g., the proper labelling of Avron's ($\Rightarrow \Box$) rule from [9] can be understood as follows:

$$\Rightarrow\!\Box\,\frac{G \mid \Box\Gamma \Rightarrow \varphi}{G \mid \Box\Gamma \Rightarrow \Box\varphi} \quad\rightsquigarrow\quad \frac{G \mid \overbrace{}^{i} \mid \overbrace{\Box\Gamma \Rightarrow \varphi}^{j}}{G \mid \underbrace{\Box\Gamma \Rightarrow \Box\varphi}_{i}} \quad\rightsquigarrow\quad \mathsf{L}\!\Rightarrow\!\Box\,\frac{\Pi, j{:}\Box\Gamma \Rightarrow \Sigma, j{:}\varphi}{\Pi, i{:}\Box\Gamma \Rightarrow \Sigma, i{:}\Box\varphi}\, j \notin X \sqcup \{i\}$$

where $X = \mathtt{L}(G)$ is the set of labels assigned to the side components from G. Consequently, j is a label to be removed from the premise interpolant, and the two split versions of this rule are $\Box$-like and $\Diamond$-like, rather than local. See [55] and [58, Sect. 3] for more details and interpolant transformations for other rules. The proof-theoretic methods to prove the LIP directly via hypersequents have been applied to S5 and modal logic S4.2 in [55, 58].

Nested Sequents, or Tree-Hypersequents Another commonly used configuration of sequent components is a tree of sequent components [16, 61, 77], usually under the names of *nested sequents* or *tree-hypersequents*. Tree structures can be conveniently represented by labels consisting of sequences of integers, so it is no surprise that prefixed tableaux relate to nested sequents the same way as ordinary tableaux do to ordinary sequents [26]. In this case, the implicit relational atoms are of the form $\sigma R \sigma.n$, where σ is a sequence of integers and $\sigma.n$ is obtained by appending it with integer n. All the details can be found in [27] and [58, Sect. 5]. Here, we only show an instance of a nested rule whose splits are $\Box$- and $\Diamond$-like rules respectively:

$$\frac{\Pi\{\Gamma \Rightarrow \Delta, [\varphi]\}}{\Pi\{\Gamma \Rightarrow \Delta, \Box\varphi\}} \quad\rightsquigarrow\quad \frac{\sigma R \sigma.j, \mathcal{R}, \Pi^{\ell}, \sigma{:}\Gamma \Rightarrow \sigma{:}\Delta, \sigma.j{:}\varphi}{\mathcal{R}, \Pi^{\ell}, \sigma{:}\Gamma \Rightarrow \sigma{:}\Delta, \sigma{:}\Box\varphi}\, \sigma.j \notin \mathcal{R}, \Pi^{\ell}, \sigma{:}\Gamma \Rightarrow \sigma{:}\Delta, \sigma{:}\Box\varphi$$

where Π^{ℓ} is the labelled translation of the nested context $\Pi\{\ \}$, sequent $\Gamma \Rightarrow \Delta, \Box\varphi$ is an ordinary sequent labelled with an integer sequence σ, and $\mathcal{R}$ describes the tree structure of the nested tree of $\Pi\{\ \}$. Here, label $\sigma.j$ is the one to be removed from the premise interpolant. The proof-theoretic methods to prove the LIP directly via nested sequents have been applied [27, 58] to all 15 logics of the modal cube from [31, Sect. 8].

Intermediate Logics. All formalisms so far in Section 5 dealt with CPC-based modal logics. The method can also be applied to intermediate logics, with intuitionistic implication and negation playing the role of modality in modal logics. In most cases, this requires specialized, restricted types of nested sequents and/or hypersequents (see [59, 60] for details). All seven interpolable intermediate logics can be dealt with using one calculus or another. Moreover,

the LIP for the Gödel–Dummett logic, the logic of linear intuitionistic Kripke frames, was first demonstrated using this method [59].

Uniform Interpolation. Hypersequents and nested sequents can also be used to demonstrate the UIP/ULIP. This has been done to reprove the UIP for logics K, D, and T, and to provide the first proof-theoretic proof of the ULIP for logic $\mathsf{K5}$ [38,39]. Similar to the labelled sequent definition of Craig interpolation (Definition 64), a labelled version of uniform interpolation is incorporating semantic notions, based on bisimulation quantifiers.

6 Conclusion

This chapter provides methods for proving interpolation results that simultaneously provide algorithms for constructing interpolants based on various (generalized) sequent calculi. For each algorithm, standard interpolant transformations are presented for common rule types, which cover a wide range of logics. As a result, the chapter can be used as a manual: To prove a particular interpolation property for a logic represented by a given calculus, the rules of the calculus should be checked against the common rule types. If each of the calculus rules is covered by one of these types, the interpolation property directly follows. Otherwise, to prove interpolation, it is sufficient to develop interpolant transformations only for the rules not covered by the common types. The proof-theoretic techniques for sequent calculi are further embedded within the framework of universal proof theory, which systematically relates interpolation properties to the existence of well-behaved proof systems.

It is a common misconception that the existence of a cut-free (sequent) calculus for a logic necessarily implies that the logic enjoys interpolation. It may be impossible to provide an interpolant transformation for a particular rule, even for an analytic one. An example of such non-interpolable hypersequent rule for the Gödel–Dummett logic can be found in the proof of [59, Theorem 4.6]. Note that this logic does enjoy interpolation, both Craig and Lyndon. Thus, more complex calculi may be necessary despite the existence of a cut-free sequent one.

Pointers to the Literature. For related reading on the discussed material in this chapter we provide the reader with some pointers to the literature:

- *Maehara's method:* Maehara's original work [64]. Textbooks on proof theory with Craig interpolation proofs via sequents for classical/intuitionistic propositional logic and classical/intuitionistic first-order logic [81, 84, 86]. See also the lecture notes in [37] on which Section 3 is based.
- *Tableau methods:* Interpolation proofs via tableaux for modal and intuitionistic logics [25].
- *Pitts' method:* Pitts' original paper on the proof of uniform interpolation for intuitionistic propositional logic [76]. Proof-theoretic methods for uniform interpolation for modal logics [11].

- *Universal proof theory:* Semi-analytic rules and the existence of well-behaved sequent calculi via the examination of interpolation [2–5, 7, 44, 45]. See also the published lecture notes in [47] on which Section 4 is based.
- *Interpolation via hypersequents:* Craig/Lyndon interpolation for modal logics [55, 58] and for intermediate logics [59]. Uniform modal interpolation [39].
- *Interpolation via nested sequents:* Craig/Lyndon interpolation for modal logics [27, 58] and for an intermediate logic [59] using linear nested sequents. Purely syntactic representation of the method [63]. Uniform interpolation for modal logics [13, 39].
- *Interpolation via labelled sequents:* Craig/Lyndon interpolation for modal logics [57, 58].
- *Interpolation via custom-made generalizations of sequents:* Hypersequents with an additional designated root component [38]. Nested sequents with bounds on depth and/or branching [59, 60].

Related Topics. Here we would like to point to some related research directions on proof-theoretic methods of interpolation that fall beyond the scope of this chapter:

- *Interpolation as cut-introduction:* In [80], a *proof-relevant* view on Maehara's method in linear logic is provided in the sense that given a cut-free sequent proof π of $\varphi \Rightarrow \psi$, there is a formula θ in the common signature of φ and ψ, and there are sequent proofs π_1, π_2 for $\varphi \Rightarrow \theta$ and $\theta \Rightarrow \psi$, respectively, such that π_1 composed with π_2 cut-reduces to π. This leads to the introduction of cuts by synthesizing the interpolant.
- *Interpolation via cyclic and non-wellfounded proofs:* Chapter 11 of this book discusses Maehara-type methods to construct interpolants for fixpoint logics via sequent proofs that may contain infinite branches.
- *Interpolation and proof complexity:* Interpolation has also been extensively studied in proof systems such as resolution and cutting planes [53, 80], where it plays a central role in proof complexity. In particular, *feasible interpolation* investigates when interpolants can be efficiently extracted from proofs, thereby linking short proofs to small interpolating circuits. This connection provides a powerful bridge between proof complexity and circuit complexity, allowing lower bounds on circuit size to be transferred to lower bounds on proof length and enabling the classification of proof systems and the establishment of exponential separations. These themes are discussed in detail in Chapter 6 of this book.

Acknowledgments

The authors sincerely thank Patrick Koopmann, Rosalie Iemhoff, and Sam van Gool for careful reading of the earlier versions of this manuscript and for their insightful comments, as well as Marianna Girlando for helpful discussions. Iris van der Giessen acknowledges the support by the Dutch Research Council (NWO) under the project *Finding Interpolants: Proofs in Action* with file number VI.Veni.232.369 of the research program Veni. Roman Kuznets was supported by Czech Science Foundation Grant No. 22-06414L.

References

1 Bahareh Afshari, Graham E. Leigh, and Guillermo Menéndez Turata. Uniform interpolation from cyclic proofs: The case of modal mu-calculus. In A. Das and S. Negri, editors, *TABLEAUX 2021*, volume 12842 of *LNCS*, pages 335–353. Springer, 2021. doi:10.1007/978-3-030-86059-2_20.

2 Amirhossein Akbar Tabatabai, Rosalie Iemhoff, and Raheleh Jalali. Uniform Lyndon interpolation for basic non-normal modal logics. In A. Silva et al., editors, *WoLLIC 2021*, volume 13038 of *LNCS*, pages 287–301. Springer, 2021. doi:10.1007/978-3-030-88853-4_18.

3 Amirhossein Akbar Tabatabai, Rosalie Iemhoff, and Raheleh Jalali. Uniform Lyndon interpolation for basic non-normal modal and conditional logics. Eprint 2208.05202, arXiv, 2022. doi:10.48550/arXiv.2208.05202.

4 Amirhossein Akbar Tabatabai, Rosalie Iemhoff, and Raheleh Jalali. Uniform Lyndon interpolation for intuitionistic monotone modal logic. Eprint 2208.04607, arXiv, 2022. doi:10.48550/arXiv.2208.04607.

5 Amirhossein Akbar Tabatabai and Raheleh Jalali. Universal proof theory: Semi-analytic rules and uniform interpolation. Eprint 1808.06258, arXiv, 2018. doi:10.48550/arXiv.1808.06258.

6 Amirhossein Akbar Tabatabai and Raheleh Jalali. Universal proof theory: Feasible admissibility in intuitionistic modal logics. *Annals of Pure and Applied Logic*, 176:103526, 2025. doi:10.1016/j.apal.2024.103526.

7 Amirhossein Akbar Tabatabai and Raheleh Jalali. Universal proof theory: Semi-analytic rules and Craig interpolation. *Annals of Pure and Applied Logic*, 176:103509, 2025. doi:10.1016/j.apal.2024.103509.

8 Majid Alizadeh, Farzaneh Derakhshan, and Hiroakira Ono. Uniform interpolation in substructural logics. *Review of Symbolic Logic*, 7:455–483, 2014. doi:10.1017/S175502031400015X.

9 Arnon Avron. The method of hypersequents in the proof theory of propositional non-classical logics. In W. Hodges et al., editors, *Logic: from Foundations to Applications*, pages 1–32. Clarendon Press, 1996. doi:10.1093/oso/9780198538622.003.0001.

10 Matthias Baaz and Helmut Veith. Interpolation in fuzzy logic. *Archive for Mathematical logic*, 38:461–489, 1999. doi:10.1007/s001530050151.

11 Marta Bílková. *Interpolation in Modal Logics*. PhD thesis, Charles University, Prague, Czech Republic, 2006. URL: http://hdl.handle.net/20.500.11956/15732.

12 Marta Bílková. Uniform interpolation and propositional quantifiers in modal logics. *Studia Logica*, 85:1–31, 2007. doi:10.1007/s11225-007-9021-5.

13 Marta Bílková. A note on uniform interpolation proofs in modal deep inference calculi. In N. Bezhanishvili et al., editors, *TbiLLC 2009*, volume 6618 of *LNCS*, pages 30–45. Springer, 2011. doi:10.1007/978-3-642-22303-7_3.

14 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001. doi:10.1017/CBO9781107050884.

15 Branislav R. Boričić. A cut-free Gentzen-type system for the logic of the weak law of excluded middle. *Studia Logica*, 45:39–53, 1986. doi:10.1007/BF01881548.

16 Kai Brünnler. Deep sequent systems for modal logic. *Archive for Mathematical Logic*, 48:551–577, 2009. doi:10.1007/s00153-009-0137-3.

17 A. Carbone. Interpolants, cut elimination and flow graphs for the propositional calculus. *Annals of Pure and Applied Logic*, 83:249–299, 1997. doi:10.1016/S0168-0072(96)00019-X.

18 Alexander Chagrov and Michael Zakharyaschev. *Modal Logic*. Clarendon Press, 1997. doi:10.1093/oso/9780198537793.001.0001.

19 William Craig. Linear reasoning. A new form of the Herbrand–Gentzen Theorem. *Journal of Symbolic Logic*, 22:250–268, 1957. doi:10.2307/2963593.

20 Kosta Došen. Sequent-systems and groupoid models. I. *Studia Logica*, 47:353–385, 1988. doi:10.1007/BF00671566.

21 Kosta Došen. Modal translations in substructural logics. *Journal of Philosophical Logic*, 21:283–336, 1992. doi:10.1007/BF00260931.

22 Roy Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 57:795–807, 1992. doi:10.2307/2275431.

23 Hugo Férée, Iris van der Giessen, Sam van Gool, and Ian Shillito. Mechanised uniform interpolation for modal logics K, GL, and iSL. In C. Benzmüller et al., editors, *IJCAR 2024*, volume 14740 of *LNCS*, pages 43–60. Springer, 2024. doi:10.1007/978-3-031-63501-4_3.

24 Hugo Férée and Sam van Gool. Formalizing and computing propositional quantifiers. In R. Krebbers et al., editors, *CPP 2023*, pages 148–158. ACM, 2023. doi:10.1145/3573105.3575668.

25 Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel Publishing Company, 1983. doi:10.1007/978-94-017-2794-5.

26 Melvin Fitting. Prefixed tableaus and nested sequents. *Annals of Pure and Applied Logic*, 163:291–313, 2012. doi:10.1016/j.apal.2011.09.004.

27 Melvin Fitting and Roman Kuznets. Modal interpolation via nested sequents. *Annals of Pure and Applied Logic*, 166:274–305, 2015. doi:10.1016/j.apal.2014.11.002.

28 Wesley Fussner and Simon Santschi. Interpolation in linear logic and related systems. *ACM Transactions on Computational Logic*, 25:20, 2024. doi:10.1145/3680284.

29 Dov M. Gabbay and Larisa Maksimova. *Interpolation and Definability: Modal and Intuitionistic Logics*. Clarendon Press, 2005. doi:10.1093/acprof:oso/9780198511748.001.0001.

30 N. Galatos, P. Jipsen, T. Kowalski, and H. Ono. *Residuated Lattices: An algebraic Glimpse at Substructural Logics*, volume 151 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2007.

31 James Garson. Modal logic. In E. N. Zalta and U. Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Stanford University, 2024. URL: https://plato.stanford.edu/archives/spr2024/entries/logic-modal/.

32 Gerhard Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39:176–210, 1935. doi:10.1007/BF01201353.

33 Gerhard Gentzen. Untersuchungen über das logische Schließen. II. *Mathematische Zeitschrift*, 39:405–431, 1935. doi:10.1007/BF01201363.

34 Silvio Ghilardi and Marek Zawadowski. Undefinability of propositional quantifiers in the modal system S4. *Studia Logica*, 55:259–271, 1995. doi:10.1007/BF01061237.

35 Silvio Ghilardi and Marek Zawadowski. *Sheaves, Games, and Model Completions: A Categorical Approach to Nonclassical Propositional Logics*. Springer, 2002. doi:10.1007/978-94-015-9936-8.

36 Iris van der Giessen. *Uniform Interpolation and Admissible Rules: Proof-theoretic investigations into (intuitionistic) modal logics*. PhD thesis, Utrecht University, Utrecht, Netherlands, 2022. doi:10.33540/1486.

37 Iris van der Giessen. Interpolation through the lens of proof theory: Lecture notes for Logic Summer School @ANU 2024, Canberra, Australia. Eprint X-2025-05, ILLC Preprints and Publications, 2025. URL: https://eprints.illc.uva.nl/id/eprint/2403.

38 Iris van der Giessen, Raheleh Jalali, and Roman Kuznets. Extensions of K5: Proof theory and uniform Lyndon interpolation. In R. Ramanayake and J. Urban, editors, *TABLEAUX 2023*, volume 14278 of *LNCS*, pages 263–282. Springer, 2023. doi:10.1007/978-3-031-43513-3_15.

39 Iris van der Giessen, Raheleh Jalali, and Roman Kuznets. Uniform interpolation via nested sequents and hypersequents. *Journal of Logic and Computation*, 35(6), 2025. doi:10.1093/logcom/exae053.

40 J.-Y. Girard. Linear logic: its syntax and semantics. In J.-Y. Girard et al., editors, *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995. doi:10.1017/CBO9780511629150.002.

41 Jean-Yves Girard. Light linear logic. In D. Leivant, editor, *LCC 1994*, volume 960 of *LNCS*, pages 145–176. Springer, 1995. doi:10.1007/3-540-60178-3_83.

42 Stefan Hetzl and Raheleh Jalali. On the completeness of interpolation algorithms. In *LICS 2024*. ACM, 2024. doi:10.1145/3661814.3662112.

43 Jörg Hudelmaier. An $O(n \log n)$-space decision procedure for intuitionistic propositional logic. *Journal of Logic and Computation*, 3:63–75, 1993. doi:10.1093/logcom/3.1.63.

44 Rosalie Iemhoff. Uniform interpolation and sequent calculi in modal logic. *Archive for Mathematical Logic*, 58:155–181, 2019. doi:10.1007/s00153-018-0629-0.

45 Rosalie Iemhoff. Uniform interpolation and the existence of sequent calculi. *Annals of Pure and Applied Logic*, 170:102711, 2019. doi:10.1016/j.apal.2019.05.008.

46 Rosalie Iemhoff. The G4i analogue of a G3i sequent calculus. *Studia Logica*, 110:1493–1506, 2022. doi:10.1007/s11225-022-10008-3.

47 Rosalie Iemhoff and Raheleh Jalali. Universal proof theory. In M. M. Clementino et al., editors, *Topology, Algebra and Categories in Logic: Lecture Notes of the Coimbra TACL Summer School, 14–18 June 2022*, volume 5 of *Coimbra Mathematical Texts*. Springer, 2026. In Press.

48 Max Kanovich, Stepan Kuznetsov, Vivek Nigam, and Andre Scedrov. Subexponentials in non-commutative linear logic. *Mathematical Structures in Computer Science*, 29:1217–1249, 2019. doi:10.1017/S0960129518000117.

49 Eiji Kiriyama and Hiroakira Ono. The contraction rule and decision problems for logics without structural rules. *Studia Logica*, 50:299–319, 1991. doi:10.1007/BF00370189.

50 Zoltan A. Kocsis. Proof-theoretic methods in quantifier-free definability. *Annals of Pure and Applied Logic*, 176:103555, 2025. doi:10.1016/j.apal.2025.103555.

51 Yuichi Komori. Super-Łukasiewicz propositional logics. *Nagoya Mathematical Journal*, 84:119–133, 1981. doi:10.1017/S0027763000019577.

52 Tomasz Kowalski and Hiroakira Ono. Analytic cut and interpolation for bi-intuitionistic logic. *Review of Symbolic Logic*, 10:259–283, 2017. doi:10.1017/S175502031600040X.

53 P.S. Krzystek and S. Zachorowski. Łukasiewicz logics have not the interpolation property. *Reports on Mathematical Logic*, 9:39–40, 1977.

54 Taishi Kurahashi. Uniform Lyndon interpolation property in propositional modal logics. *Archive for Mathematical Logic*, 59(5–6):659–678, August 2020. doi:10.1007/s00153-020-00713-y.

55 Roman Kuznets. Craig interpolation via hypersequents. In D. Probst and P. Schuster, editors, *Concepts of Proof in Mathematics, Philosophy, and Computer Science*, pages 193–214. De Gruyter, 2016. doi:10.1515/9781501502620-012.

56 Roman Kuznets. Interpolation method for multicomponent sequent calculi. In S. Artemov and A. Nerode, editors, *LFCS 2016*, volume 9537 of *LNCS*, pages 202–218. Springer, 2016. doi:10.1007/978-3-319-27683-0_15.

57 Roman Kuznets. Proving Craig and Lyndon interpolation using labelled sequent calculi. In L. Michael and A. Kakas, editors, *JELIA 2016*, volume 10021 of *LNCS*, pages 320–335. Springer, 2016. doi:10.1007/978-3-319-48758-8_21.

58 Roman Kuznets. Multicomponent proof-theoretic method for proving interpolation properties. *Annals of Pure and Applied Logic*, 169:1369–1418, 2018. doi:10.1016/j.apal.2018.08.007.

59 Roman Kuznets and Björn Lellmann. Interpolation for intermediate logics via hyper- and linear nested sequents. In G. Bezhanishvili et al., editors, *Advances in Modal Logic*, volume 12, pages 473–492. College Publications, 2018.

60 Roman Kuznets and Björn Lellmann. Interpolation for intermediate logics via injective nested sequents. *Journal of Logic and Computation*, 31:797–831, 2021. doi:10.1093/logcom/exab015.

61 Björn Lellmann and Francesca Poggiolesi. Nested sequents or tree-hypersequents—a survey. In Y. Weiss and R. Birman, editors, *Saul Kripke on Modal Logic*, pages 243–301. Springer, 2024. doi:10.1007/978-3-031-57635-5_11.

62 Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, 410:4747–4768, 2009. doi:10.1016/j.tcs.2009.07.041.

63 Tim Lyon, Alwen Tiu, Rajeev Goré, and Ranald Clouston. Syntactic interpolation for tense logics and bi-intuitionistic logic via nested sequents. In M. Fernández and A. Muscholl, editors, *CSL 2020*, volume 152 of *LIPIcs*, pages 28:1–28:16. Dagstuhl–LZI, 2020. doi:10.4230/LIPIcs.CSL.2020.28.

64 Shôji Maehara. On the interpolation theorem of Craig. *Sugaku*, 12:235–237, 1961. In Japanese. doi:10.11429/sugaku1947.12.235.

65 L. L. Maksimova. Craig's theorem in superintuitionistic logics and amalgamable varieties of pseudo-Boolean algebras. *Algebra and Logic*, 16:427–455, 1977. doi:10.1007/BF01670006.

66 Larisa Maksimova. Amalgamation and interpolation in normal modal logics. *Studia Logica*, 50:457–471, 1991. doi:10.1007/BF00370682.

67 Enrico Marchioni and George Metcalfe. Craig interpolation for semilinear substructural logics. *Mathematical Logic Quarterly*, 58:468–481, 2012. doi:10.1002/malq.201200004.

68 Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied logic*, 51:125–157, 1991. doi:10.1016/0168-0072(91)90068-W.

69 G. E. Minc. On some calculi of modal logic. In V. P. Orevkov, editor, *The Calculi of Symbolic Logic. I*, number 98 (1968) in Proceedings of the Steklov Institute of Mathematics, pages 97–124. AMS, 1971. URL: https://www.mathnet.ru/eng/tm/v98/p88.

70 Franco Montagna. Interpolation and Beth's property in propositional many-valued logics: A semantic investigation. *Annals of Pure and Applied Logic*, 141:148–179, 2006. doi:10.1016/j.apal.2005.11.001.

71 Sara Negri. Proof analysis in modal logic. *Journal of Philosophical Logic*, 34:507–544, 2005. doi:10.1007/s10992-005-2267-3.

72 Sara Negri and Jan von Plato. *Proof Analysis: A Contribution to Hilbert's Last Problem*. Cambridge University Press, 2011. doi:10.1017/CBO9781139003513.

73 Hiroakira Ono. Structural rules and a logical hierarchy. In P. P. Petkov, editor, *Mathematical Logic*, pages 95–104. Plenum Press, 1990. doi:10.1007/978-1-4613-0609-2_8.

74 Hiroakira Ono. Proof-theoretic methods in nonclassical logic — an introduction. In M. Takahashi et al., editors, *Theories of Types and Proofs*, pages 207–254. Mathematical Society of Japan, 1998. doi:10.2969/msjmemoirs/00201C060.

75 Hiroakira Ono and Yuichi Komori. Logics without the contraction rule. *Journal of Symbolic Logic*, 50:169–201, 1985. doi:10.2307/2273798.

76 Andrew M. Pitts. On an interpretation of second order quantification in first order intuitionistic propositional logic. *Journal of Symbolic Logic*, 57:33–52, 1992. doi:10.2307/2275175.

77 F. Poggiolesi. The method of tree-hypersequents for modal propositional logic. In D. Makinson et al., editors, *Towards Mathematical Philosophy*, pages 31–51. Springer, 2009. doi:10.1007/978-1-4020-9084-4_3.

78 Garrel Pottinger. Uniform, cut-free formulations of T, S_4, and S_5. *Journal of Symbolic Logic*, 48:900, 1983. Abstract. doi:10.2307/2273495.

79 Dirk Roorda. Interpolation in fragments of classical linear logic. *Journal of Symbolic Logic*, 59:419–444, 1994. doi:10.2307/2275398.

80 Alexis Saurin. Interpolation as cut-introduction: On the computational content of Craig–Lyndon interpolation. In M. Fernández, editor, *FSCD 2025*, volume 337 of *LIPIcs*, pages 32:1–32:21. Dagstuhl–LZI, 2025. doi:10.4230/LIPIcs.FSCD.2025.32.

81 K. Schütte. *Proof Theory*. Springer, 1977. doi:10.1007/978-3-642-66473-1.

82 D. S. Shamkanov. Circular proofs for the Gödel–Löb provability logic. *Mathematical Notes*, 96:575–585, 2014. doi:10.1134/S0001434614090326.

83 Daniyar S. Shamkanov. Interpolation properties for provability logics GL and GLP. *Proceedings of the Steklov Institute of Mathematics*, 274:303–316, 2011. doi:10.1134/S0081543811060198.

84 Gaisi Takeuti. *Proof Theory*, volume 81 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1975.

85 A. S. Troelstra. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. CSLI Publications, 1992.

86 A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, 2nd edition, 2000. doi:10.1017/CBO9781139168717.

87 A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics: An Introduction, Volume I*, volume 121 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1988.

88 Alasdair Urquhart. Failure of interpolation in relevant logics. *Journal of Philosophical Logic*, 22:449–479, 1993. doi:10.1007/BF01349560.

89 N. N. Vorob'ev. The derivability problem in the constructive propositional calculus with strong negation. *Doklady Akademii Nauk SSSR*, 85:689–692, 1952. In Russian.

90 N. N. Vorob'ev. A new algorithm for derivability in the constructive propositional calculus. *American Mathematical Society Translations, Series 2*, 94:37–71, 1970. doi:10.1090/trans2/094.

Chapter 6

Proof Complexity and Feasible Interpolation

Amirhossein Akbar Tabatabai
Bernoulli Institute, University of Groningen, The Netherlands

Abstract

Propositional proof complexity is a branch of proof theory that studies the complexity of proofs, primarily their length, across various proof systems for propositional and modal logics. In this chapter, we provide a gentle introduction to propositional proof complexity, focusing especially on the method of *feasible interpolation* as a tool for establishing the existence of hard theorems, that is, short formulas requiring exponentially long proofs. We begin by introducing some basic concepts from descriptive complexity, circuit complexity, and cryptography. Next, we present an abstract definition of a proof system along with several illustrative examples. We then explore various forms of the feasible interpolation property and apply them to demonstrate the existence of hard theorems in proof systems for classical logic, such as resolution and cut-free sequent calculus, as well as in systems for intuitionistic and modal logics, including their standard Hilbert-style and sequent calculi. Finally, we show that this method fails for strong proof systems such as the sequent calculus **LK**, under widely believed cryptographic hardness assumptions including the security of RSA and the Diffie-Hellman key exchange protocol against polynomial-size circuits. Using this failure, we prove that these systems are not *automatable* in a precise formal sense.

Contents

1 Introduction 198
1.1 Propositional Incompleteness 198
1.2 Feasible Interpolation 199
1.3 Outline of the Chapter 200
2 Preliminaries 201
3 The Propositional World 204
3.1 Descriptive Power 204
3.2 Computational Power 208
4 Proof Systems 216
4.1 Simulations and Relative Efficiency 220
4.2 Absolute Efficiency 221



© Amirhossein Akbar Tabatabai;
licensed under Creative Commons License CC BY 4.0
In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 6, pages 197–243. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.f

5 Interpolants and their Computation 223
5.1 Computing Craig Interpolants in Classical Logic 223
5.2 Non-classical Alternatives to Craig Interpolation 225
6 Feasible Interpolation (Classical) 229
6.1 Applications 233
7 Feasible Interpolation (Non-classical) 236
7.1 Super-intuitionistic Logics 237
7.2 Modal Logics 240
Acknowledgments 241
References 241

1 Introduction

At first glance, propositional language might seem too simple to capture the interest of a working mathematician, as its expressive power, compared to first-order languages, is quite limited, mainly due to the lack of quantifiers. However, this initial impression is misleading. As we will see later in this chapter, the propositional language is fully capable of describing any *finite* structure (even up to isomorphism), much like first-order languages are used to describe *arbitrary* structures (only up to elementary equivalence).

With this hidden expressive power in mind, propositional proofs should be regarded as more significant than they are often perceived, since they can be used to reason indirectly about any finite structure. But what do we mean by a propositional proof? In its most abstract form, a propositional proof is defined by the proof system in which it resides. For a given propositional (or sometimes modal) logic L, a *proof system for* L is a polynomial-time predicate $P(\pi, \phi)$ such that $\phi \in L$ if and only if there exists π satisfying $P(\pi, \phi)$, which we read as "π is a P-*proof* of ϕ". P is assumed to be polynomial-time computable to ensure that verifying whether a given object is a valid proof of a proposition is *computationally easy*, while the equivalence condition encodes the standard requirements of *soundness* and *completeness* of P with respect to L.

1.1 Propositional Incompleteness

Proof systems, in the sense defined above, cannot be incomplete, since they are sound and complete by definition. However, by shifting the focus from the "non-existence of a proof" to the "non-existence of a *short* proof," where short means polynomial in the size of the formula we want to prove, a new form of incompleteness emerges. Formally, given a logic L, we call a proof system for L *polynomially bounded* (or *p-bounded*) if there exists a polynomial p such that any $\phi \in L$ has a P-proof π whose size is bounded by $p(|\phi|)$. In this way, a p-bounded proof system can be interpreted as the analogue of a *complete first-order theory*, where the

condition of the "existence of a proof for any true statement" is replaced by the "existence of a short proof for any formula in L."

Similar to the quest for first-order completeness, we may ask whether a p-bounded proof system exists for a given logic L. It is easy to see that a logic L has a p-bounded proof system if and only if $L \in \mathbf{NP}$. Therefore, classical (resp. intuitionistic) propositional logic, CPC (resp. IPC), has a p-bounded proof system if and only if $\mathbf{CoNP} = \mathbf{NP}$ (resp. $\mathbf{PSPACE} = \mathbf{NP}$). Since these equalities are widely believed to be false, it is similarly believed that no p-bounded proof system exists for CPC or IPC. One may view this conjectural non-existence as the propositional analogue of *Gödel's incompleteness theorem*, which asserts that any sufficiently strong and consistent theory is necessarily incomplete.

As these forms of *propositional incompleteness* are equivalent to long-standing open problems in computational complexity, their full proofs remain beyond our current reach. Nevertheless, this does not prevent us from demonstrating that some specific proof systems, such as the sequent calculi **LK** for CPC and **LJ** for IPC, are not p-bounded. Fortunately, a general technique exists for proving such results, known as *feasible interpolation*. This chapter is dedicated to presenting this method and its applications.

1.2 Feasible Interpolation

Roughly speaking, a proof system P for CPC has feasible interpolation if we can *easily* compute a Craig interpolant for an implicational tautology from any of its P-proofs. Now, subject to certain conjectures in computational complexity, we claim that if P has feasible interpolation, then it cannot be p-bounded. To prove that, first, by invoking some conjectures (or occasionally results) from computational complexity, we construct an implicational tautology with hard-to-compute Craig interpolants. Next, since P has feasible interpolation, if we have access to a P-proof of the implication, obtaining the hard-to-compute interpolant becomes easy. Therefore, the P-proofs must be too long, implying that P is not p-bounded. Note that feasible interpolation effectively reduces a question in proof complexity to one in computational complexity, where known hardness assumptions or results on *computation* can be leveraged to establish the difficulty of *proving* certain theorems.

For classical logic, the method of feasible interpolation was first introduced by Krajíček [22, 23]. Since then, it has been successfully applied to a variety of *weak* proof systems for CPC, including the cut-free sequent calculus, the resolution refutation system [23], cutting planes [31], and the Nullstellensatz system [32], among others. Of all known techniques for showing that a proof system is not p-bounded, each applicable only to weak systems, feasible interpolation stands out for its broad applicability.

For sufficiently *strong* systems, however, short proofs may exist for implications whose interpolants are conjectured (often on cryptographic grounds) to be computationally hard. In such cases, the direction of the argument can be reversed to conclude that the system does not admit feasible interpolation. For instance, **LK** fails to have feasible interpolation unless the Diffie-Hellman key exchange protocol is insecure against poly-size circuits [5, 6, 24, 25]. Since

feasible interpolation is the main technique we rely on, this limitation might initially seem like a setback. However, the failure of feasible interpolation leads to an interesting consequence: under a natural technical condition, it implies that the proof system is *non-automatable*, meaning that it is impossible to find a proof of a tautology in time polynomially bounded by the sum of the size of its shortest proof and the size of the formula itself. As a result, we can show that **LK** is not automatable, unless the Diffie-Hellman key exchange protocol is insecure against poly-size circuits. For a comprehensive discussion on feasible interpolation and its connections to computational complexity and cryptography, see [21].

For non-classical logics, one typically works with strong proof systems, and, as in the classical case, the feasible interpolation technique is generally unavailable. However, the power of this method can be recovered by replacing Craig interpolants with a parametrized version of the disjunction property, which we refer to as *disjunctive interpolants*. This notion was first developed by Buss and Pudlák [8], following the feasible version of the disjunction property proved by Buss and Mints [7]. The concept of feasible disjunctive interpolation was later generalized by Hrubeš [11–13], who used it to show that **LJ** and any Hilbert-style proof system for intuitionistic logic, or for modal logics below $\mathsf{S4}$ or GL, are not p-bounded. These results were subsequently extended to superintuitionistic logics and modal logics of infinite branching by Jeřábek [16] and by Jalali [15] to any logic between Full Lambek logic FL and a superintuitionistic logic of infinite branching. For additional results on non-classical proof complexity, see [17–19], and for an insightful survey, refer to [4].

1.3 Outline of the Chapter

The structure of this chapter is as follows. After introducing the preliminaries in Section 2, Section 3 presents tools from descriptive complexity that enable the translation of first-order formulas over finite structures into propositional formulas. We also utilize the propositional language to describe computations over finite domains through circuits and discuss relevant topics in cryptography and disjoint **NP** pairs. In Section 4, we introduce the fundamentals of proof systems, provide examples, and compare them in terms of their relative strength. Section 5 uses hard disjoint **NP** pairs to construct implicational classical tautologies with Craig interpolants that are hard to compute. It also defines disjunctive interpolants and provides hard-to-compute instances for them. Readers interested in the computational complexity of these interpolants, but not necessarily in the proof complexity aspects, will find relevant material in this section. Section 6 introduces the concept of feasible interpolation in the classical setting, including its monotone version and automatability. We show that the cut-free sequent calculus and the resolution system have (monotone) feasible interpolation and are therefore not p-bounded. Additionally, assuming certain cryptographic hardness conjectures, we demonstrate that some strong proof systems lack feasible interpolation and are thus not automatable. Finally, in Section 7, we extend these ideas to superintuitionistic and modal sequent-style and Hilbert-style proof systems.

A Word to the Reader: The literature on proof complexity often assumes fluency in computational complexity and bounded theories of arithmetic, which can be challenging for newcomers. To make this material more accessible, we present it with the assumption of only a basic familiarity with propositional, modal, and first-order logic, as well as a fundamental understanding of key concepts in computational complexity, such as the definitions of the classes **NP** and **PSPACE**. Any additional concepts will be introduced and explained as needed.

2 Preliminaries

By $\mathcal{L}_b$ we mean the propositional language $\{\top, \bot, \wedge, \vee, \neg\}$. The language $\mathcal{L}_b^u = \{\top, \bot, \bigwedge, \bigvee, \neg\}$ is defined similarly to $\mathcal{L}_b$, with the difference that it allows conjunctions and disjunctions of arbitrary finite arity rather than just binary ones. The language $\mathcal{L}_p$ is defined as $\{\top, \bot, \wedge, \vee, \rightarrow\}$, and $\mathcal{L}_\Box = \mathcal{L}_p \cup \{\Box\}$. For any $\mathfrak{L} \in \{\mathcal{L}_b, \mathcal{L}_b^u, \mathcal{L}_p, \mathcal{L}_\Box\}$, we define $\mathfrak{L}$-*formulas*, or simply *formulas in* $\mathfrak{L}$, in the usual way. When the context is clear, we may omit specifying the language and simply refer to them as formulas. In the languages $\mathcal{L}_p$ and $\mathcal{L}_\Box$, by $\neg\phi$ and $\boxdot\phi$ we mean $\phi \rightarrow \bot$ and $(\Box\phi \wedge \phi)$, respectively. For any formula ϕ, we denote by $V(\phi)$ the set of atomic formulas occurring in ϕ. Over $\mathcal{L}_b$ and $\mathcal{L}_b^u$, a *literal* is either a variable p or its negation $\neg p$, and a *clause* is a finite sequence of literals, interpreted as their disjunction. More generally, a formula in $\mathcal{L}_b$ is said to be in *negation normal form* if the only negations used in the formula appear on atomic formulas, $\top$, or $\bot$. It is clear that any formula in the languages $\mathcal{L}_b$, $\mathcal{L}_b^u$, and $\mathcal{L}_p$ is classically equivalent to one in negation normal form, using De Morgan's laws. For a set P of atomic formulas, a formula is called *monotone* in P if $\neg p$ does not occur in its negation normal form, for any $p \in P$. A formula in any of the four languages is called *monotone* if it contains no negation or implication. For any $\mathcal{L}_b^u$-formula ϕ, define the depth $dp(\phi)$ as follows: $dp(p) = dp(\top) = dp(\bot) = 0$, $dp(\neg\phi) = dp(\phi)$, $dp(\bigwedge_i \phi_i) = 1 + \max_i dp(\phi_i)$ and $dp(\bigvee_i \phi_i) = 1 + \max_i dp(\phi_i)$.

Let $\mathfrak{L} \in \{\mathcal{L}_b, \mathcal{L}_b^u, \mathcal{L}_p\}$ and let $P = \{p_1, \ldots, p_m\}$ be a set of atomic formulas. By a *Boolean assignment* for P, we mean a tuple $\bar{a} \in \{0,1\}^m$ that assigns a_i to p_i for each $1 \leq i \leq m$. When m is implicit, we simply write $\bar{a} \in \{0,1\}$. An $\mathfrak{L}$-formula is called *valid* if it evaluates to 1 under all Boolean assignments to its atomic formulas. A formula (resp. a set of clauses) is called *satisfiable* if there exists a Boolean assignment for all involved atomic formulas such that the formula (resp. the disjunction of any of the clauses) evaluates to 1. We define CPC as the set of all valid $\mathcal{L}_p$-formulas. By a slight abuse of notation, we also use CPC to refer to the set of valid $\mathfrak{L}$-formulas for $\mathfrak{L} \in \{\mathcal{L}_b, \mathcal{L}_b^u\}$. The logic CPC has the *Craig Interpolation Property*, meaning that for any valid implication $\phi \rightarrow \psi$, there exists a formula I such that $V(I) \subseteq V(\phi) \cap V(\psi)$ and both $(\phi \rightarrow I)$ and $(I \rightarrow \psi)$ are valid. Any such I is called a *Craig interpolant* for $\phi \rightarrow \psi$. Sometimes, the Craig interpolation property is stated in terms of unsatisfiable clauses: that is, for any unsatisfiable set $\{C_i\}_i \cup \{D_j\}_j$ of clauses, there exists a formula I using only the variables common to $\{C_i\}_i$ and $\{D_j\}_j$ such that both $\{C_i\}_i \cup \{I\}$

$$\frac{}{\phi \Rightarrow \phi} \qquad \frac{}{\bot \Rightarrow} \qquad \frac{}{\Rightarrow \top}$$

$$\frac{\Gamma, \phi, \psi, \Pi \Rightarrow \Delta}{\Gamma, \psi, \phi, \Pi \Rightarrow \Delta}\, Le \qquad \frac{\Gamma \Rightarrow \Delta, \phi, \psi, \Lambda}{\Gamma \Rightarrow \Delta, \psi, \phi, \Lambda}\, Re \qquad \frac{\Gamma \Rightarrow \phi, \Delta \qquad \Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\, cut$$

$$\frac{\Gamma \Rightarrow \Delta}{\Gamma, \phi \Rightarrow \Delta}\, Lw \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \phi, \Delta}\, Rw \qquad \frac{\Gamma, \phi, \phi \Rightarrow \Delta}{\Gamma, \phi \Rightarrow \Delta}\, Lc \qquad \frac{\Gamma \Rightarrow, \phi, \phi, \Delta}{\Gamma \Rightarrow \phi, \Delta}\, Rc$$

$$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}\, L\wedge_1 \qquad \frac{\Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}\, L\wedge_2 \qquad \frac{\Gamma \Rightarrow \phi, \Delta \qquad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}\, R\wedge$$

$$\frac{\Gamma, \phi \Rightarrow \Delta \qquad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}\, L\vee \qquad \frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}\, R\vee_1 \qquad \frac{\Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}\, R\vee_2$$

$$\frac{\Gamma \Rightarrow \phi, \Delta \qquad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \to \psi \Rightarrow \Delta}\, L\to \qquad \frac{\Gamma, \phi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \to \psi, \Delta}\, R\to$$

Figure 1 The sequent calculus **LK**

and $\{D_j\}_j \cup \{\neg I\}$ are unsatisfiable. These I's are also called Craig interpolants, this time for the set $\{C_i\}_i \cup \{D_j\}_j$.

For any $\mathfrak{L} \in \{\mathcal{L}_b, \mathcal{L}_b^u, \mathcal{L}_p, \mathcal{L}_\Box\}$, a *sequent* over $\mathfrak{L}$ is an expression of the form $\Gamma \Rightarrow \Delta$, where Γ and Δ are finite sequences of $\mathfrak{L}$-formulas. A sequent is called *single-conclusion* if Δ contains at most one $\mathfrak{L}$-formula. We say that a sequent $\Gamma \Rightarrow \Delta$ over $\mathfrak{L} \in \{\mathcal{L}_b, \mathcal{L}_b^u, \mathcal{L}_p\}$ is *valid* if the formula $\bigwedge \Gamma \to \bigvee \Delta$ is valid. The sequent calculus **LK** over $\mathcal{L}_p$ is defined as in Figure 1. Define **LJ** over $\mathcal{L}_p$ similarly to **LK**, restricting it to operate only on single-conclusion sequents. Over $\mathcal{L}_\Box$, consider the following list of modal rules:

$$\frac{\Gamma \Rightarrow \phi}{\Box\Gamma \Rightarrow \Box\phi}\, K \qquad \frac{\Gamma \Rightarrow \Delta}{\Box\Gamma \Rightarrow \Box\Delta}\, D \qquad \frac{\Box\Gamma, \Gamma \Rightarrow \phi}{\Box\Gamma \Rightarrow \Box\phi}\, K4 \qquad \frac{\Box\Gamma, \Gamma \Rightarrow \Delta}{\Box\Gamma \Rightarrow \Box\Delta}\, KD4$$

$$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma, \Box\phi \Rightarrow \Delta}\, LS4 \qquad \frac{\Box\Gamma \Rightarrow \phi}{\Box\Gamma \Rightarrow \Box\phi}\, RS4 \qquad \frac{\Box\Gamma, \Gamma, \Box\phi \Rightarrow \phi}{\Box\Gamma \Rightarrow \Box\phi}\, GL$$

where in rules (D) and $(KD4)$, the sequence Δ contains at most one formula. Now, consider the sequent calculi listed in Table 1. In each of the sequent calculi introduced above, the cut rule can be eliminated without affecting the set of provable sequents. We denote the cut-free version of a calculus by appending a superscript minus sign to its name. For example, the system **LK** without the cut rule is denoted by $\mathbf{LK}^-$. For any of the sequent calculi G defined above, by its *logic* we mean the set of all formulas ϕ such that $G \vdash \Rightarrow \phi$. We usually denote sequent calculi with boldface letters and their corresponding logics with the same letters in sans-serif font. For instance, the logic $\mathsf{S4}$ is the logic of the calculus **S4**. The only exceptions are the logics IPC and CPC, which are the logics of **LJ** and **LK**, respectively.

Table 1 Some sequent calculi and their logics.

Calculus Name	Definition	Logic Name
K	**LK** + K	K
D	**LK** + D	D
KT	**LK** + K + LS4	KT
K4	**LK** + K4	K4
KD4	**LK** + KD4	KD4
S4	**LK** + LS4 + RS4	S4
GL	**LK** + GL	GL

There is a connection between IPC and CPC that we will use in this chapter, called *Glivenko's theorem*. It states that for any sequence $\Gamma \cup \{\phi\}$ of formulas, if $\Gamma \vdash_{\mathsf{CPC}} \phi$, then $\neg\neg\Gamma \vdash_{\mathsf{IPC}} \neg\neg\phi$.

A *superintuitionistic* or *si logic* is a set L of $\mathcal{L}_p$-formulas that is closed under substitution and modus ponens, such that $\mathsf{IPC} \subseteq L \subseteq \mathsf{CPC}$. For any set Φ of $\mathcal{L}_p$-formulas, the logic $\mathsf{IPC}+\Phi$ refers to the smallest si logic containing Φ. For any si logic L and any set $\Gamma\cup\{\phi\}$ of $\mathcal{L}_p$-formulas, we write $\Gamma \vdash_L \phi$ if $(\Rightarrow \phi)$ is provable in **LJ** from the sequents $\{\Rightarrow \gamma\}_{\gamma\in\Gamma} \cup \{\Rightarrow \psi \mid \psi \in L\}$. A *normal modal logic*, or simply a *modal logic*, is defined as a set $L \supseteq \mathsf{K}$ of $\mathcal{L}_\Box$-formulas closed under substitution, modus ponens, and the necessitation rule $\phi/\Box\phi$. It is clear that any logic defined in Table 1 is a modal logic. For any set Φ of $\mathcal{L}_\Box$-formulas, the logic $\mathsf{K}+\Phi$ refers to the smallest modal logic containing $L \cup \Phi$. For any modal logic L and any set $\Gamma \cup \{\phi\}$ of $\mathcal{L}_\Box$-formulas, we write $\Gamma \vdash_L \phi$ if $(\Rightarrow \phi)$ is provable in **K** from the sequents $\{\Rightarrow \gamma\}_{\gamma\in\Gamma} \cup \{\Rightarrow \psi \mid \psi \in L\}$. Consider the modal logics $\mathsf{K}+\{\Box p \leftrightarrow p\}$ and $\mathsf{K}+\{\Box p\}$. Both of these modal logics are *conservative* over classical logic; that is, for any propositional formula, if it is provable in either of these logics, then it is also provable in classical logic. Moreover, by Makinson's theorem [27], any consistent modal logic is a subset of one of these two logics. There are also two translation functions, f and c, from $\mathcal{L}_\Box$ to $\mathcal{L}_p$ that preserve every constant, atom, and propositional connectives and defined on the modality by: $(\Box\phi)^f = \phi^f$ and $(\Box\phi)^c = \top$. The function f *forgets* boxes, while c *collapses* them into $\top$, corresponding to the use of the axioms $\Box p \leftrightarrow p$ and $\Box p$, respectively.

Define T_k as the si logic $\mathsf{IPC} + \{\bigwedge_{i=0}^{k}\left((p_i \to \bigvee_{j\neq i} p_j) \to \bigvee_j p_j\right) \to \bigvee_i p_i\}$. For the reader familiar with Kripke frames, the logic T_k is the set of $\mathcal{L}_p$-formulas valid in all finite Kripke frames with branching at most k. Similarly, in the modal case, define $\mathsf{K4BB}_k$ as $\mathsf{K} + \{\Box p \to \Box\Box p, \Box\left(\bigvee_{i\leq k}\Box\left(\boxdot p_i \to \bigvee_{j\neq i} p_j\right) \to \bigvee_{i\leq k}\boxdot p_i\right) \to \bigvee_{i\leq k}\Box\bigvee_{j\neq i} p_j\}$. Again, $\mathsf{K4BB}_k$ is the set of $\mathcal{L}_\Box$-formulas that are valid in all finite transitive Kripke frames with branching at most k. We say that a si (resp. modal) logic L has *branching* k if $L \supseteq \mathsf{T}_k$ (resp. $L \supseteq \mathsf{K4BB}_k$). A logic is called of *infinite branching* if it does not have branching k

for any $k \in \mathbb{N}$. In the case of si logics, Jeřábek [16] provided an interesting characterization of the logics of infinite branching. First, recall that the $\mathcal{L}_p$-formulas BD_n are defined by: $\mathrm{BD}_0 := \bot$ and $\mathrm{BD}_{n+1} := p_n \vee (p_n \to \mathrm{BD}_n)$. Then, consider the logics $\mathsf{IPC} + \{\mathrm{BD}_2\}$ and $\mathsf{IPC} + \{\mathrm{BD}_3, \neg p \vee \neg\neg p\}$ and denote them by $\mathsf{BD_2}$ and $\mathsf{BD_3} + \mathsf{KC}$, respectively. Jeřábek's characterization states that a si logic L is of infinite branching if and only if $\mathsf{L} \subseteq \mathsf{BD_2}$ or $\mathsf{L} \subseteq \mathsf{BD_3} + \mathsf{KC}$. For more on non-classical logics and their proof systems, see Chapter 3 and Chapter 5, respectively.

Finally, some points about binary strings and computation. By $\{0,1\}^*$, we mean the set of all binary strings. For any $w \in \{0,1\}^*$, let $|w|$ denote the length of w, and for $i \leq |w|$, let w_i represent the i-th bit of w. By a *sequence of short binary strings* $\{w_n\}_{n=n_0}^{\infty}$, we mean a sequence of binary strings such that there exists a polynomial p with $|w_n| \leq p(n)$ for any $n \geq n_0$. We can encode any type of finitary object, such as numbers, formulas, and proofs, by binary strings. Therefore, we can apply similar definitions to these objects via their representations. By a *language* L, we mean any subset of $\{0,1\}^*$, and L_n is defined as the set $\{w \in L \mid |w| = n\}$. The class **FP** consists of all functions $f : \{0,1\}^* \to \{0,1\}^*$ that can be computed by a deterministic Turing machine in polynomial time. We define the classes **P** and **PSPACE** as the set of languages decidable by a deterministic Turing machine in polynomial time and space, respectively. Similarly, **NP** refers to the set of languages decidable by a non-deterministic Turing machine in polynomial time. The class **CoNP** is defined as the set of languages whose complement belongs to **NP**. It is known that a language L is in **NP** if and only if there exists a polynomial time decidable predicate $P \subseteq \{0,1\}^* \times \{0,1\}^*$ and a polynomial p such that $w \in L$ if and only if there exists $u \in \{0,1\}^*$ with $|u| \leq p(|w|)$ and $P(u,w)$. It is widely believed that **NP** $\neq$ **CoNP**, and hence **NP** $\neq$ **P**. Moreover, it is believed that **NP** $\subsetneq$ **PSPACE**.

3 The Propositional World

As promised in the introduction, in this section we show that the propositional language is sufficiently powerful to address various matters related to *finite* structures. We will accomplish this in two subsections. First, we explore the *expressive* power of the propositional language. Then, in the second subsection, we use this language to define circuits and examine the *computational* power of the language in computing functions between finite sets.

3.1 Descriptive Power

Fix a multi-sorted first-order language $\mathcal{L}$ with equality and without any function symbols and choose an interpretation I that assigns to each sort σ a finite, non-empty set $I(\sigma)$. We then expand the language $\mathcal{L}$ to a new language $\mathcal{L}(I)$ by adding a constant symbol for each element in these sets. Note that we leave all relational symbols (except for equality) unspecified. Given an $\mathcal{L}(I)$-*sentence* ϕ, we define a propositional formula $[\![\phi]\!]$ that expresses the same content as ϕ. The key idea is this: for each relational symbol (except equality) of the form $R(x_1^{\sigma_1}, \ldots, x_k^{\sigma_k})$,

we introduce a propositional variable $p_{a_1a_2\ldots a_k}$ to represent the truth value of $R(a_1,\ldots,a_k)$ for every $a_i \in I(\sigma_i)$. Equality statements $a = b$ are interpreted by evaluating whether a and b denote the same element. Propositional connectives are mapped to themselves, and quantifiers are translated into finite conjunctions and disjunctions. More formally, the translation $[\![-]\!]$ from $\mathcal{L}(I)$-sentences to propositional formulas is defined recursively as follows:

- $[\![c]\!] = c$ for any $c \in \{\top, \bot\}$;
- $[\![a = b]\!] = \top$ if $a = b$, and $[\![a = b]\!] = \bot$ otherwise;
- $[\![R(a_1,\ldots,a_k)]\!] = p_{a_1a_2\ldots a_k}$ for any $a_i \in I(\sigma_i)$;
- $[\![\phi \circ \psi]\!] = [\![\phi]\!] \circ [\![\psi]\!]$ for any logical connective $\circ \in \{\wedge, \vee, \to\}$;
- $[\![\forall x^\sigma\, \phi(x)]\!] = \bigwedge_{a\in I(\sigma)} [\![\phi(a)]\!]$ and $[\![\exists x^\sigma\, \phi(x)]\!] = \bigvee_{a\in I(\sigma)} [\![\phi(a)]\!]$.

For any extension of I to a full interpretation I' (one that also interprets the relational symbols), we can define a Boolean assignment that maps each propositional variable $p_{a_1a_2\ldots a_k}$ to true if and only if $R(a_1,\ldots,a_k)$ holds under I', for all $a_i \in I(\sigma_i)$. Conversely, given any Boolean assignment to the atoms $p_{a_1a_2\ldots a_k}$, we can construct a corresponding extension of I to a full interpretation. It is straightforward to verify that under this correspondence, the $\mathcal{L}(I)$-sentence ϕ holds under the interpretation I' if and only if the propositional formula $[\![\phi]\!]$ evaluates to true under the associated Boolean assignment. In this sense, ϕ and $[\![\phi]\!]$ express the same content.

Now, we use the propositional translation $[\![-]\!]$ to introduce the propositional counterparts of certain first-order sentences that will play a role later in the chapter. These examples will also help illustrate how the translation operates in concrete cases.

▶ **Example 1.** Let $\mathcal{L}$ be a two-sorted language with sorts σ and τ, containing no function symbols and a single binary relation symbol $R(x^\sigma, y^\tau)$. Define $I(\sigma) = \{1,\ldots,m\}$ and $I(\tau) = \{1,\ldots,n\}$. The formula $\phi = \forall x^\sigma \forall y_1^\tau y_2^\tau\, (R(x^\sigma, y_1^\tau) \wedge R(x^\sigma, y_2^\tau) \to y_1^\tau = y_2^\tau)$ expresses that the relation R is functional. To transform this sentence into a propositional formula, we introduce propositional atoms p_{ij} to represent the truth of $R(i,j)$, for all $i \le m$ and $j \le n$. Then, we have $[\![\phi]\!] = \bigwedge_{i\le m} \bigwedge_{j_1\le n} \bigwedge_{j_2\le n} ((p_{ij_1} \wedge p_{ij_2}) \to [\![j_1 = j_2]\!])$. When $j_1 = j_2$, the formula $[\![\phi]\!]$ is trivially true as $[\![j_1 = j_2]\!] = \top$. Hence, it is enough to restrict to $j_1 \neq j_2$. As $[\![j_1 = j_2]\!] = \bot$ in this case, $[\![\phi]\!]$ is equivalent to $\bigwedge_{i\le m} \bigwedge_{j_1\neq j_2\le n} (\neg p_{ij_1} \vee \neg p_{ij_2})$.

For the second example, consider $\psi = \forall x_1^\sigma x_2^\sigma \forall y^\tau\, (R(x_1^\sigma, y^\tau) \wedge R(x_2^\sigma, y^\tau) \to x_1^\sigma = x_2^\sigma)$ stating that R is an injective relation. Using a similar argument as above, we have: $[\![\psi]\!] = \bigwedge_{i_1\neq i_2\le m} \bigwedge_{j\le n} (\neg p_{i_1j} \vee \neg p_{i_2j})$. As the final example, consider the sentence $\theta = \forall x^\sigma \exists y^\tau R(x^\sigma, y^\tau)$ stating that R is a total relation. Therefore, we have $[\![\theta]\!] = \bigwedge_{i\le m} \bigvee_{j\le n} p_{ij}$. ⌟

▶ **Example 2.** Let $\mathcal{L}$ be the first-order language with one sort σ, no function symbols, and the binary relation $E(x^\sigma, y^\sigma)$. Define $I(\sigma) = \{1,\ldots,n\}$, and interpret any element of $I(\sigma)$ as a vertex of a graph with n vertices. Now, if we interpret $E(x^\sigma, y^\sigma)$ as "there is an edge from x to y," then any extension of I to a full interpretation corresponds to a directed graph on n vertices. First, consider the sentence $\phi = \forall x^\sigma \neg E(x^\sigma, x^\sigma) \wedge \forall x_1^\sigma x_2^\sigma (E(x^\sigma, y^\sigma) \to E(y^\sigma, x^\sigma))$, which states that the graph is symmetric and has no loops; i.e., it is essentially a simple graph.

To translate ϕ into a propositional formula, we introduce propositional atoms s_{ij} to encode whether $E(i,j)$ holds, for all $i,j \leq n$. Then, $[\![\phi]\!] = (\bigwedge_{i\leq n} \neg s_{ii}) \wedge \bigwedge_{i_1,i_2\leq n}(s_{i_1i_2} \to s_{i_2i_1})$. To simplify the propositional translation, we use the following trick: Instead of s_{ij}'s, use the atoms p_{ij} for any pair $\{i,j\} \subseteq \{1,\ldots,n\}$ that contains exactly two elements. It is clear that the assignments for these new atoms are in one-to-one correspondence with the assignments that make $[\![\phi]\!]$ true. Using this trick, we can indirectly refer to simple graphs without writing out any propositional formula.

Now, let us extend $\mathcal{L}$ by adding a new sort τ and a new relational symbol $R(z^\tau, x^\sigma)$. Define $I(\tau) = \{1,\ldots,k\}$. From Example 1, we know how to write a sentence ψ that states that $R(z^\tau, x^\sigma)$ is an injective function from $\{1,\ldots,k\}$ into $\{1,\ldots,n\}$. Then, consider the sentence $\theta = \psi \wedge \forall z_1^\tau z_2^\tau \forall x_1^\sigma x_2^\sigma\, [(R(z_1^\tau, x_1^\sigma) \wedge R(z_2^\tau, x_2^\sigma) \wedge z_1^\tau \neq z_2^\tau) \to E(x_1^\sigma, x_2^\sigma)]$, which states that the function R maps different elements to connected vertices. This simply says that R describes a k-clique inside the graph. Using q_{ui} to encode whether $R(u,i)$ holds, for $u \leq k$ and $i \leq n$, the propositional formula $[\![\theta]\!]$ is the conjunction of the following:

- $\bigvee_{i\leq n} q_{ui}$, for all $u \leq k$,
- $\neg q_{ui_1} \vee \neg q_{ui_2}$, for all $u \leq k$ and any $i_1 \neq i_2 \leq n$,
- $\neg q_{u_1 i} \vee \neg q_{u_2 i}$, for all $u_1 \neq u_2 \leq k$ and any $i \leq n$,
- $\neg q_{u_1 i_1} \vee \neg q_{u_2 i_2} \vee p_{i_1 i_2}$, for all $u_1 \neq u_2 \leq k$ and any set $\{i_1, i_2\} \subseteq \{1,\ldots,n\}$ with two elements.

Note that the formula is monotone in $\bar{p}$. This reflects the fact that if additional edges are added to the graph, a k-clique selected by $\bar{q}$ remains a k-clique. ⌟

▶ **Example 3.** We continue with a similar language as in Example 2, consisting of two sorts, σ and τ, and binary relational symbols $E(x^\sigma, y^\sigma)$ and $R(x^\sigma, z^\tau)$. Define $I(\sigma) = \{1,\ldots,n\}$ and $I(\tau) = \{1,\ldots,l\}$. From Example 1, we know how to write a sentence ϕ that states that $R(x^\sigma, z^\tau)$ is a function from $\{1,\ldots,n\}$ into $\{1,\ldots,l\}$. Then, consider the sentence $\psi = \phi \wedge \forall z_1^\tau z_2^\tau \forall x_1^\sigma x_2^\sigma\, [(R(x_1^\sigma, z_1^\tau) \wedge R(x_2^\sigma, z_2^\tau) \wedge E(x_1^\sigma, x_2^\sigma)) \to z_1^\tau \neq z_2^\tau]$, which states that R maps connected vertices into different elements. This simply says that R describes an l-coloring of the graph. Using p_{ij}'s as in Example 2 to encode the simple graph and r_{ia} to encode $R(i,a)$, for any $i \leq n$ and $a \leq l$, the formula $[\![\psi]\!]$ is the conjunction of the following:

- $\bigvee_{a\leq l} r_{ia}$, for all $i \leq n$,
- $\neg r_{ia_1} \vee \neg r_{ia_2}$, for all $a_1 \neq a_2 \leq l$ and any $i \leq n$,
- $\neg r_{i_1 a} \vee \neg r_{i_2 a} \vee \neg p_{i_1 i_2}$, for all $a \leq l$ and any set $\{i_1, i_2\} \subseteq \{1,\ldots,n\}$ with two elements. ⌟

So far, we have seen that any first-order formula with uninterpreted relational symbols, provided it refers exclusively to a single finite domain, can be translated into an equivalent propositional formula. By systematically ranging over all finite possibilities, one can express the finite content of a first-order formula as a sequence of propositional formulas. More precisely, let $\mathcal{L}$ be a multi-sorted first-order language with equality and without any function

symbols.[1] For each sort σ in $\mathcal{L}$, fix a polynomial P_σ in one variable, and for each natural number $n \in \mathbb{N}$, define the interpretation I_n by setting $I_n(\sigma) = \{1, \ldots, P_\sigma(n)\}$. Let $[\![\phi]\!]_n$ denote the propositional translation of the first-order $\mathcal{L}$-sentence ϕ with respect to I_n. The resulting sequence $\{[\![\phi]\!]_n\}_n$ can then be viewed as a propositional encoding of ϕ across all finite domains, parametrized polynomially by n. It is also customary to restrict to $n \geq n_0$, for some given $n_0 \in \mathbb{N}$ to encode ϕ evaluated over all *large enough* finite domains. Since the size of each domain $I_n(\sigma)$ grows polynomially with n, it follows that the size of $[\![\phi]\!]_n$ is also polynomially bounded in n. Consequently, this translation produces a sequence of short propositional formulas (see Section 2 for the definition). In cases where all extensions of I_n to a full interpretation make ϕ true, the translation gives us a sequence of short classical *tautologies*. A canonical example of such a situation is:

▶ **Example 4** (Propositional Pigeonhole Principle)**.** Let $n_0 \in \mathbb{N}$ be a number and $m(n)$ be a polynomially bounded function satisfying $m(n) > n$, for any $n \geq n_0$. By the pigeonhole principle, there is no injective function from the set $\{1, \ldots, m(n)\}$ to the set $\{1, \ldots, n\}$, if $n \geq n_0$. We want to express this fact propositionally. As illustrated in Example 1, to propositionally express that a binary relation $R \subseteq \{1, \ldots, m(n)\} \times \{1, \ldots, n\}$ is an injective function, it suffices to use atoms p_{ij} to indicate that $R(i, j)$, for any $i \leq m(n)$ and $j \leq n$. Then, the conjunction of the following clauses is the propositional translation we needed:

- $\neg p_{ij_1} \vee \neg p_{ij_2}$, for all $i \leq m(n)$ and $j_1, j_2 \leq n$ with $j_1 \neq j_2$,
- $\neg p_{i_1 j} \vee \neg p_{i_2 j}$, for all $i_1, i_2 \leq m(n)$ and $j \leq n$ with $i_1 \neq i_2$,
- $\bigvee_{j \leq n} p_{ij}$, for all $i \leq m(n)$.

Since every assignment for the above clauses corresponds one-to-one with an injective function $R \subseteq \{1, \ldots, m(n)\} \times \{1, \ldots, n\}$, and there is no such R, we conclude that the above conjunction is unsatisfiable. We define the *propositional pigeonhole principle*, denoted by $\mathrm{PHP}_n^{m(n)}$, as the negation of this conjunction. Therefore, $\{\mathrm{PHP}_n^{m(n)}\}_{n \geq n_0}$ is a sequence of short tautologies. Notable choices of $m(n)$ discussed in the literature include $m(n) = n + 1$, $m(n) = 2n$, and $m(n) = n^2$ [21]. ⌟

Using the explained translation, we transformed first-order formulas into sequences of short propositional formulas, establishing a connection between two types of *description*. We can also shift the first-order side of this connection to something computational to transform any language in **NP** into a sequence of short propositional formulas. The key is to find an appropriate way to *describe* the languages in **NP**. To this goal, interpret any binary string $w \in \{0, 1\}^*$ of length n as a unary relation R_w on the set $\{1, \ldots, n\}$, where $i \in R_w$ if and only if $w_i = 1$, for any $i \leq n$. Then, one direction of the celebrated Fagin's Theorem states:

[1] The machinery we are about to introduce does not interact smoothly with function symbols. However, function symbols can always be replaced by relational symbols, accompanied by axioms asserting their functionality. In this sense, nothing is lost.

▶ **Theorem 5** (Fagin [14]). *Let $L \in \mathbf{NP}$ be a language. Then, there exists a one-sorted first-order language $\mathcal{L}$ with no function symbols and a first-order formula $\phi(\bar{S}, R)$, where R is a unary predicate such that $(\{1,\dots,n\}, R_w) \vDash \exists \bar{S}\, \phi(\bar{S}, R)$ iff $w \in L$, for every $w \in \{0,1\}^*$.*

Using this descriptive characterization of **NP** and the translation process we explained, we obtain the following:

▶ **Theorem 6.** *For any language $L \in \mathbf{NP}$ (resp. $L \in \mathbf{CoNP}$), there exists a sequence $\{\phi_n(p_1,\dots,p_n,\bar{q})\}_n$ of short propositional formulas such that for every $n \in \mathbb{N}$ and every $w \in \{0,1\}^n$, the formula $\phi_n(w_1,\dots,w_n,\bar{q})$ is satisfiable (resp. a tautology) iff $w \in L$.*

Proof. We will prove the case for **NP** only; the other case follows immediately as a consequence. By Theorem 5, there exists a one-sorted first-order language $\mathcal{L}$ and a first-order formula $\phi(R,\bar{S})$ in $\mathcal{L}$ such that for any string $w \in \{0,1\}^n$, we have $w \in L$ iff $(\{1,\dots,n\}, R_w) \vDash \exists \bar{S}\, \phi(R,\bar{S})$. Fix $n \in \mathbb{N}$, use the interpretation I_n that maps the only sort to $\{1,\dots,n\}$, and translate the formula $\phi(R,\bar{S})$ into its propositional counterpart $\phi_n = [\![\phi]\!]_n$. Let $p_1,\dots,p_n$ and $\bar{q}$ denote the propositional variables in ϕ_n that encode the relations R and $\bar{S}$, respectively. Since there is a one-to-one correspondence between interpretations of the relations in $\bar{S}$ (resp. R) and Boolean assignments to $\bar{q}$ (resp. $\bar{p}$), it follows that $(\{1,\dots,n\}, R_w) \vDash \exists \bar{S}\, \phi(R,\bar{S})$ if and only if the Boolean assignment $w_1,\dots,w_n$ to the atoms $p_1,\dots,p_n$ makes $\phi_n(p_1,\dots,p_n,\bar{q})$ satisfiable. ◀

3.2 Computational Power

In the previous subsection, we showed that the propositional language is expressive enough to represent first-order formulas over finite domains. Moreover, by focusing on satisfiability, we saw that any language in **NP** can be encoded as a sequence of short propositional formulas. Similarly, the propositional language can be used to *compute* any function between finite domains. To formalize such computations, we introduce *circuits* as our model of computation and explain some related concepts. For a comprehensive introduction, see [20].

▶ **Definition 7** ($\mathfrak{L}$-Circuit). *Let $\mathfrak{L} \in \{\mathcal{L}_b, \mathcal{L}_p, \mathcal{L}_\square\}$. A* multi $\mathfrak{L}$-circuit *C with n inputs, m outputs, and size s is a directed acyclic graph $C = (V, E, t, O)$ where:*

- *V is the set of* gates *with s elements, and $E \subseteq V \times V$ is the set of directed* edges*,*
- *$t : V \to \{x_1,\dots,x_n\} \cup \mathfrak{L}$ assigns a type to each gate, where the arity of the type matches the gate's indegree (the arity of x_i's is zero),*
- *O is a sequence of m gates that serve as the* output *gates.*

A multi $\mathfrak{L}$-circuit is called monotone *if it contains no negation or implication gates. It is called an* $\mathfrak{L}$-circuit *if it has exactly one output.*

Observe that any $\mathfrak{L}$-formula can be viewed as an $\mathfrak{L}$-circuit in which every gate other than the output gate has outdegree one. Moreover, given any $\mathfrak{L}$-circuit, there is a canonical way to convert it into an $\mathfrak{L}$-formula by duplicating shared sub-circuits as needed. We denote this

formula by $[C]$. We call a (multi) $\mathcal{L}_b$-circuit simply a *(multi) circuit*. Any (multi) $\mathcal{L}_p$-circuit can be transformed into a (multi) circuit by replacing $\rightarrow$ gates with combinations of $\neg$ and $\vee$ gates. Conversely, any (multi) circuit can be transformed into a (multi) $\mathcal{L}_p$-circuit by rewriting $\neg$ using $\rightarrow$ and $\bot$. These transformations incur only a linear increase in (multi) circuit size. Therefore, we treat (multi) $\mathcal{L}_p$-circuits and (multi) circuits as essentially equivalent.

Given any (multi) $\mathcal{L}_\Box$-circuit C, we can erase all $\Box$ gates and continue the edge coming into the gate to all edges going out of it to obtain a (multi) $\mathcal{L}_p$-circuit, which we denote by C^f. Alternatively, we may eliminate all incoming edges to each $\Box$ gate and replace each such gate with a constant $\top$ gate, yielding another (multi) $\mathcal{L}_p$-circuit denoted by C^c. Note that the size of C^f and C^c are bounded by the size of C. These two operations correspond to the forgetful and collapse translations introduced in Section 2.

If C is a multi circuit with n inputs and m outputs and $w \in \{0,1\}^n$, we write $C(w)$ to denote the string output on the sequence of output gates when C runs on the input $x_i = w_i$, for any $1 \leq i \leq n$. A function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ is said to be *computed* by C if $C(w) = f(w)$ for every $w \in \{0,1\}^n$. It is easy to verify that any (monotone) function can be computed by a (monotone) multi circuit, although the size of the multi circuit may be exponential in n and m. Since any finite set can be embedded into $\{0,1\}^n$ for some $n \in \mathbb{N}$, we conclude, just as we did in the context of first-order descriptions, that the propositional language is expressive enough to compute any function between finite domains.

Similar to the previous subsection, we can use *sequences* of multi circuits to compute functions defined over all binary strings. The idea is to use a sequence $\{C_n\}_n$ of multi circuits, where each C_n has n input gates and computes the restriction of f to $\{0,1\}^n$. However, there is a technical issue with this approach: the length of $f(w)$ may vary across different inputs w of length n, while the number of output gates in C_n must be fixed for each n. To address this, we introduce a padding trick that ensures all outputs $f(w)$ for $w \in \{0,1\}^n$ are extended with additional bits so that they all have the same length.

To define the padding function in its general form, let $m, l \in \mathbb{N}$ be natural numbers and $m \geq l$. For any string $u \in \{0,1\}^l$, we define its padded version of length $2m$ by $\mathrm{pad}_m(u) = 1u_1\,1u_2 \cdots 1u_l\,00\cdots0$, where the number of trailing zeros is $2m - 2l$. Intuitively, we extend u by padding it with zeros on the right to reach the desired length. To mark the end of the original string, we insert a 1 before each bit of u. Then, by reading the last 1 in an even position, we can determine where the original u ends.

▶ **Definition 8.** *Let $\{C_n\}_n$ be a sequence of multi circuits such that each C_n has n inputs and an even number of outputs. Let $2o(n)$ denote the number of outputs of C_n. Then the sequence $\{C_n\}_n$ is said to compute a function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ if, for every $n \in \mathbb{N}$ and every $w \in \{0,1\}^n$, we have $|f(w)| \leq o(n)$ and $C_n(w) = \mathrm{pad}_{o(n)}(f(w))$. We say that the function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is* computable by poly-size circuits *if it is computed by such a sequence $\{C_n\}_n$ where the size of each C_n is bounded by a polynomial in n. Otherwise, we say that f is* hard for poly-size circuits.

▶ Remark 9. If the values $f(w)$ for $w \in \{0,1\}^n$ already have the same length $out_f(n)$, then without loss of generality we may define the computation without padding by requiring $C_n(w) = f(w)$ for every $n \in \mathbb{N}$ and $w \in \{0,1\}^n$. This is because when padding is used, the multi circuit C_n includes additional output gates to append constant zeros and ones to the actual output, but these additional outputs can be safely ignored. Conversely, since the number of constant-zero or constant-one output gates required for padding is bounded by $out_f(n)$, we can simply add them to the multi circuit to translate from the simplified formalism back to the padded one. Therefore, the identification between the two formalisms affects the size of the multi circuits only up to a constant factor. As the order of size is the primary parameter of interest for us, we treat the two formalisms as equivalent. ⌟

▶ **Definition 10** (Class $\mathbf{P/poly}$). *A language L is in $\mathbf{P/poly}$ if its characteristic function $\chi_L : \{0,1\}^* \to \{0,1\}$ is computable by poly-size circuits, i.e., there exists a sequence of circuits $\{C_n\}_n$ such that the size of each C_n is polynomially bounded in n, and for every $w \in \{0,1\}^n$, we have $w \in L$ if and only if $C_n(w) = 1$.*

Computation with poly-size circuits is the *non-uniform* analogue of polynomial-time computation, as it allows for an arbitrary choice of the multi circuit C_n for each input length n, as long as the size of these multi circuits remains polynomial in n. As expected, non-uniform computation must subsume uniform computation. In fact, if a function $f : \{0,1\}^* \to \{0,1\}^*$ is in $\mathbf{FP}$, it is computable by poly-size circuits, which implies $\mathbf{P} \subseteq \mathbf{P/poly}$ [3]. It is widely believed that $\mathbf{NP} \not\subseteq \mathbf{P/poly}$. Since this would imply $\mathbf{NP} \neq \mathbf{P}$, this belief motivates the combinatorial approach to attack the conjecture $\mathbf{NP} \neq \mathbf{P}$ by identifying languages in $\mathbf{NP}$ that require super-polynomial-size circuits. Despite decades of effort, this remains one of the most prominent open problems in theoretical computer science. However, for *monotone* circuits, such lower bounds have been successfully established. For instance, following Example 2, any binary string of length $\binom{n}{2}$ can be interpreted as the adjacency matrix of a simple graph on n vertices. Let Clique^n be the set of all such binary strings that, when read as graphs, contain a $\lfloor n^{1/4} \rfloor$-clique, and define $\text{Clique} = \bigcup_n \text{Clique}^n$. It is clear that $\text{Clique} \in \mathbf{NP}$. However:

▶ **Theorem 11** (Razborov [33], Alon-Boppana [2]). *For any sequence $\{C_n\}_n$ of monotone circuits computing* Clique*, the size of $C_{\binom{n}{2}}$ is at least $2^{\Omega(n^{1/8} \log n)}$.*

3.2.1 A Bit of Cryptography

In this subsubsection, we introduce some basic concepts and examples from cryptography, which interestingly employs the seemingly negative phenomenon of *computational hardness* to achieve the positive goal of *secure communication*. For our purposes, we simplify everything to the level needed in this chapter.

We begin by developing the intuition behind a notion we refer to as a *one-way protocol*, starting with its simplest setting. Let $P \subseteq \{0,1\}^*$ be a polynomial-time decidable predicate, and let $h : P \to \{0,1\}^*$ be a polynomial-time computable function. Intuitively, an element

$u \in P$ represents some *source data* that we wish to keep secret, yet must still communicate about through public channels. To overcome this seemingly contradictory situation, we apply h to u, producing $h(u)$, and transmit $h(u)$ publicly instead of u itself. Naturally, we prefer h to be injective, since otherwise the transformation may obscure or distort the original data u. Crucially, the utility of h lies in its assumed *one-wayness*, namely that it is computationally hard to invert, making it infeasible for a potential eavesdropper to recover u from $h(u)$.

We now generalize this setup. Retain P and h as defined, but relax the requirement that h be injective. Instead, introduce a polynomial-time computable function $k : P \to \{0,1\}^*$ such that $h(u_1) = h(u_2)$ implies $k(u_1) = k(u_2)$, for all $u_1, u_2 \in P$. We refer to this condition as *semi-injectivity*, a weakening of injectivity. In this scenario, the goal is not necessarily to communicate the source data u, but rather some *derived information* $k(u)$. This allows for certain distortions in u, provided that the derived output $k(u)$ remains intact. Semi-injectivity guarantees precisely this. Note that when k is the identity function, this notion collapses to ordinary injectivity of h. In this generalized framework, it is no longer sufficient to merely assume that h is hard to invert. Instead, we must assume that computing $k(u)$ from $h(u)$ is computationally hard. Since k is efficiently computable, this assumption automatically implies the one-wayness of h. We are now ready to formally define a one-way protocol:

▶ **Definition 12** (One-Way Protocol). *We call a triple (P, h, k) a* one-way protocol[2] *if P is a polynomial-time decidable predicate and $h, k : \{0,1\}^* \to \{0,1\}^*$ are polynomial-time computable functions satisfying:*

- semi-injectivity*, i.e., for all $u_1, u_2 \in P$, if $h(u_1) = h(u_2)$, then $k(u_1) = k(u_2)$,*
- boundedness*, i.e., there exists a polynomial-time computable function $l : \{0,1\}^* \to \{0,1\}^*$ and a polynomial p such that $|k(u)| = l(h(u))$ and $|u| \leq p(|h(u)|)$ for any $u \in \{0,1\}^*$,*
- security against $\mathbf{P/poly}$ adversaries*, i.e., computing $k(u)$ from $h(u)$ is hard for poly-size circuits. Formally, this means that there is no function $f : \{0,1\}^* \to \{0,1\}^*$ computable by poly-size circuits such that $k(u) = f(h(u))$ for all $u \in \{0,1\}^*$.*

The existence of a one-way protocol implies that $\mathbf{NP} \neq \mathbf{P}$. To prove that, assume $\mathbf{NP} = \mathbf{P}$ and let (P, h, k) be a one-way protocol. We reach a contradiction. Since h and k are polynomial-time computable, the language $\{(v, i) \mid \exists u(|u| \leq p(|v|) \wedge i \leq |k(u)| \wedge k(u)_i = 1)\}$ is in $\mathbf{NP} \subseteq \mathbf{P}$. Hence, we can effectively compute the i-th bit of $k(u)$ by reading $h(u)$ and $i \leq |k(u)|$. Given that $|k(u)| \leq |u|^{O(1)} \leq |h(u)|^{O(1)}$, we can compute all bits of $k(u)$ in time $|h(u)|^{O(1)}$. This implies the existence of a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f(h(u)) = k(u)$ for any $u \in \{0,1\}^*$. Since f is in $\mathbf{FP}$, it is

[2] This is a non-standard adaptation of *one-way permutations* tailored for the purposes of this chapter. It differs from the conventional definition in three key ways. First, we introduce an auxiliary function k to generalize the setting and accommodate examples such as the Diffie–Hellman protocol. Second, while standard cryptographic hardness is defined against probabilistic polynomial-time adversaries, we adopt a circuit-based notion of hardness more appropriate for our context. Third, in one-way permutations, we have $|h(u)| = |u|$. We have relaxed this condition to boundedness, which helps to present protocols like RSA and Diffie-Hellman as one-way protocols with only minor adjustments.

also computable by poly-size circuits which contradicts the security of the one-way protocol. Therefore, we conclude that $\mathbf{NP} \neq \mathbf{P}$.

As the existence of a one-way protocol implies $\mathbf{NP} \neq \mathbf{P}$, we currently do not know whether any one-way protocol exists. Nevertheless, there are concrete examples of triples (P, h, k) that are widely believed to form one-way protocols based on standard cryptographic assumptions. In the remainder of this subsubsection, we introduce two central examples from classical cryptography that we will refer to later: the RSA protocol and the Diffie–Hellman protocol.

▶ **Example 13** (RSA Protocol)**.** First, let us explain RSA in its original form as a way for a user to communicate a number to us securely. Let N be a number and $1 < e < N$ such that $(e, \phi(N)) = 1$, where $\phi(N)$ is Euler's function, which returns the number of integers less than N that are coprime to N. The pair (N, e) is called the *public key* and is known to everyone. However, the factorization of N is kept secret and is known only to us. Since $(e, \phi(N)) = 1$, there exists $d < N$ such that $ed \equiv 1 \pmod{\phi(N)}$. This d is called the *secret key*. It is widely believed that one cannot efficiently compute d from N and e alone. However, if the factorization of N is known, then $\phi(N)$ can be computed easily, and hence so can d. Therefore, we are able to compute d, while the public cannot. Now, let $u < N$ be a number such that $(u, N) = 1$. To communicate u to us securely, the user computes the remainder v of u^e modulo N and sends v through public channels. Since $(u, N) = 1$, Euler's theorem gives $u^{\phi(N)} \equiv 1 \pmod{N}$. Therefore, $v^d \equiv (u^e)^d = u^{ed} \equiv u \pmod{N}$. This allows us to recover the original message u from the ciphertext v.

Now, with a slight technical adjustment, one can view RSA as a one-way protocol. After suitable encoding of the tuples of binary strings as a binary string, define P as the set of tuples (N, M, e, u), where $(e, M) = 1$, $u < N$, $1 < e < N$, and $u^M \equiv 1 \pmod{N}$. Note that M plays the role of $\phi(N)$ to make the process simpler. As we have the polynomial-time Euclidean algorithm to compute the greatest common divisor and modular exponentiation is polynomial-time computable, it is clear that P is a polynomial-time predicate. Then, define the function $h : P \to \{0,1\}^*$ by $h(N, M, e, u) = (N, e, v)$, where v is the remainder of $u^e \bmod N$. Also, define $k(N, M, e, u) = \overline{0u}$, where $\overline{0u}$ is a sequence of zeros added to the binary expansion of u such that $|\overline{0u}| = |N|$. Padding u with zeros is a technicality to make $|k(u)|$ easily computable from $h(u)$. Both h and k are polynomial-time computable. For semi-injectivity, if $h(N_1, M_1, e_1, u_1) = h(N_2, M_2, e_2, u_2)$, then $N_1 = N_2 = N$ and $e_1 = e_2 = e$. Moreover, $u_1^e \equiv u_2^e \pmod{N}$. Since $(e, M) = 1$, there is $d < M$ such that $ed \equiv 1 \pmod{M}$. Therefore, as $u_1^M \equiv u_2^M \equiv 1 \pmod{N}$, we have $u_1 = u_1^1 \equiv u_1^{ed} \equiv u_2^{ed} \equiv u_2^1 = u_2 \pmod{N}$. As $u_1, u_2 < N$, we reach $u_1 = u_2$. Moreover, as $N_1 = N_2 = N$, the number of zeros needed to pad $u_1 = u_2$ into a string with the length $|N|$ are equal. Therefore, $k(N_1, M_1, e_1, u_1) = \overline{0u_1} = \overline{0u_2} = k(N_2, M_2, e_2, u_2)$. For boundedness, as $M, u \leq N$, we have $|(N, M, e, u)| \leq O(|(N, e, v)|) = O(|h(N, M, e, u)|)$. Moreover, note that $|k(N, M, e, u)| = |\overline{0u}| = |N|$ is computable from $h(N, M, e, u) = (N, e, v)$ in polynomial time, as we have a direct access to N. For the hardness, it is widely believed that RSA is secure against $\mathbf{P/poly}$ adversaries [21]. Under this assumption, RSA qualifies as a one-way protocol. ⌟

▶ **Example 14** (Diffie–Hellman Key Exchange Protocol)**.** First, let us explain the Diffie–Hellman key exchange protocol in its original form as a way to set a secret key between Alice and Bob over public channels. Let N and g be publicly known natural numbers. To establish a shared secret key, Alice chooses a secret number $a \leq N$ and sends the remainder X of g^a modulo N to Bob over public channels. Similarly, Bob chooses a secret number $b \leq N$ and sends the remainder Y of g^b modulo N to Alice over public channels. Note that the values X and Y are also publicly known, but a is known only to Alice and b only to Bob. Since Bob knows b, he can compute $X^b \equiv g^{ab} \pmod{N}$, and similarly, Alice can compute $Y^a \equiv g^{ab} \pmod{N}$. The resulting number g^{ab} is the shared secret key known to both parties. The central premise of the Diffie–Hellman protocol is that computing the remainder of g^{ab} modulo N from the public data N, g, X, and Y is computationally infeasible. In fact, when $N = pq$ for primes p and q, and $(g, N) = 1$, it is known that breaking the Diffie-Hellman protocol is harder than factoring N [6]. Again, this protocol can be seen as a one-way protocol. After encoding tuples of binary strings as a binary string, define P as the set of all tuples (N, g, a, b), where $a, b \leq N$, and note that P is a polynomial-time predicate. Then, define a function $h : P \to \{0,1\}^*$ by $h(N, g, a, b) = (N, g, X, Y)$, where X and Y are the remainders of g^a and g^b modulo N, respectively. Also define $k(N, g, a, b) = \bar{0}Z$, where Z is the binary expansion of the remainder of g^{ab} modulo N and $\bar{0}$ is a sequence of zeros such that $|\bar{0}Z| = |N|$. Both h and k are clearly computable in polynomial time as modular exponentiation is in polynomial time. For semi-injectivity, if $h(N_1, g_1, a_1, b_1) = h(N_2, g_2, a_2, b_2)$, then $N_1 = N_2 = N$, $g_1 = g_2 = g$, $g^{a_1} \equiv g^{a_2} \pmod{N}$, and $g^{b_1} \equiv g^{b_2} \pmod{N}$. Therefore, $g^{a_1 b_1} \equiv (g^{a_1})^{b_1} \equiv (g^{a_2})^{b_1} \equiv (g^{b_1})^{a_2} \equiv (g^{b_2})^{a_2} \equiv g^{a_2 b_2} \pmod{N}$, which implies that $k(N_1, g_1, a_1, b_1) = k(N_2, g_2, a_2, b_2)$. For boundedness, as $a, b \leq N$, we have $|(N, g, a, b)| \leq O(|(N, g, X, Y)|) = O(|h(N, g, a, b)|)$. Moreover, note that $|k(N, g, a, b)| = |\bar{0}Z| = |N|$ is computable from $h(N, g, a, b) = (N, g, X, Y)$ in polynomial time, as we have direct access to N. For the hardness condition, it is widely believed that the Diffie–Hellman protocol is secure against $\mathbf{P}/\mathbf{poly}$ adversaries. Under this assumption, the Diffie–Hellman key exchange protocol qualifies as a one-way protocol. ⌟

3.2.2 Disjoint NP Pairs

In this subsubsection, we introduce and explain the concept of a disjoint **NP** pair. Such pairs arise naturally in computational complexity and cryptography. Through the translation of languages in **NP** into propositional formulas, they are also closely connected to Craig interpolation for CPC, as will be explained in Section 5. We begin with the formal definition:

▶ **Definition 15** (Disjoint **NP** Pair)**.** *A* disjoint **NP** pair *is a pair of languages* (L, R) *such that* $L, R \in \mathbf{NP}$ *and* $L \cap R = \emptyset$*. A* separator *for such a pair is a language* S *satisfying* $L \subseteq S$ *and* $S \cap R = \emptyset$*. A disjoint* **NP** *pair is called* hard *if it has no separators in* $\mathbf{P}/\mathbf{poly}$*.*

A natural source of disjoint **NP** pairs comes from languages in the class $\mathbf{NP} \cap \mathbf{CoNP}$. Specifically, if $L \in \mathbf{NP} \cap \mathbf{CoNP}$, then the pair (L, L^c) forms a disjoint **NP** pair. Note that L is the only separator for this pair. Therefore, if $\mathbf{NP} \cap \mathbf{CoNP} \not\subseteq \mathbf{P}/\mathbf{poly}$, then

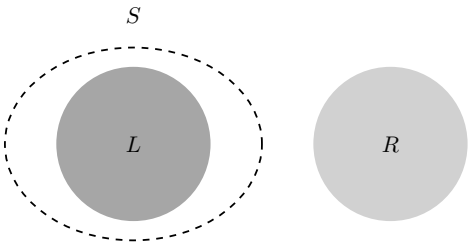


Figure 2 Disjoint **NP**-pair (L, R) with the separator S.

there exists a disjoint **NP** pair that has no separator in **P/poly**. Since it is believed that $\mathbf{NP} \cap \mathbf{CoNP} \not\subseteq \mathbf{P/poly}$, we conjecture that a hard disjoint **NP** pair must exist. However, the existence of such pairs remains unproven, as it would imply $\mathbf{NP} \neq \mathbf{P}$. Indeed, if $\mathbf{NP} = \mathbf{P}$, then for any disjoint **NP** pair (L, R), the language $L \in \mathbf{NP} = \mathbf{P} \subseteq \mathbf{P/poly}$ provides an obvious separator in **P/poly**.

There are some concrete settings that lead to the existence of hard disjoint **NP** pairs, where we rely on stronger complexity-theoretic conjectures. In return, we gain concrete instances of such pairs, which can be exploited later. A rich source of hard disjoint **NP** pairs arises in cryptography. Let (P, h, k) be a one-way protocol and let p be the polynomial such that $|u| \leq p(|h(u)|)$, for any $u \in \{0,1\}^*$. We want to construct a hard disjoint **NP** pair. Consider the following languages:

$$L = \{(v, i) \mid \exists u \in P\,[(|u| \leq p(|v|)) \wedge (i \leq |k(u)|) \wedge (h(u) = v) \wedge (k(u))_i = 1]\},$$
$$R = \{(v, i) \mid \exists u \in P\,[(|u| \leq p(|v|)) \wedge (i \leq |k(u)|) \wedge (h(u) = v) \wedge (k(u))_i = 0]\}.$$

Since h and k are polynomial-time computable, it is clear that $L, R \in \mathbf{NP}$. Moreover, $L \cap R = \varnothing$. To see this, suppose for contradiction that $(v, i) \in L \cap R$. Then there exist $u_1, u_2 \in P$ such that $h(u_1) = v = h(u_2)$. By semi-injectivity, we have $k(u_1) = k(u_2)$, which implies $k(u_1)_i = k(u_2)_i$. However, $k(u_1)_i = 1$ and $k(u_2)_i = 0$, yielding a contradiction. Thus, (L, R) is indeed a disjoint **NP** pair.

Now, using the assumed security of (P, h, k) against **P/poly** adversaries, we show that (L, R) has no separator in **P/poly**. Suppose, for contradiction, that $S \in \mathbf{P/poly}$ is a separator for (L, R). Then, there exists a sequence of poly-size circuits $\{C_n\}_n$ that computes S. First, we show that for any $u \in \{0,1\}^*$ with $|h(u)| = n$ and any $i \leq |k(u)|$, we have $C_{n+|i|}(h(u), i) = k(u)_i$. The reasoning is as follows: if $C_{n+|i|}(h(u), i) = 1$, then $(h(u), i) \in S$, hence $(h(u), i) \notin R$, as $S \cap R = \emptyset$. Given that $i \leq |k(u)|$ and $|u| \leq p(|h(u)|)$, it must be that $k(u)_i \neq 0$, so $k(u)_i = 1$. Similarly, if $C_{n+|i|}(h(u), i) = 0$, then $k(u)_i = 0$.

Now, as $|u| \leq p(|h(u)|)$ and $|k(u)| \leq |u|^{O(1)}$, there exists a polynomial $q(n)$ such that $|k(u)| \leq q(|h(u)|)$ for any $u \in \{0,1\}^*$. Moreover, let $l : \{0,1\}^* \to \{0,1\}^*$ be the polynomial-time computable function satisfying $|k(u)| = l(h(u))$, for any $u \in \{0,1\}^*$. As l is polynomial-time computable, the predicate $i \leq l(w)$ is decidable in polynomial time. Therefore, it is in

$\mathbf{P/poly}$. Let $\{D_n\}_n$ be a sequence of polynomial-size circuits that decide $i \leq l(w)$. For any $i \leq q(n)$, consider the circuit $D_{n+|i|}(x_1, \ldots, x_n, i)$. Now, define the circuit $E_n(x_1, \ldots, x_n)$ in the following way: First, make the disjoint union of the circuit $C_{n+|i|}(x_1, \ldots, x_n, i)$ with the output y_i and $D^i_n(x_1, \ldots, x_n)$ with the output z_i, for any $i \leq q(n)$. Then, add some $\wedge$ gates to compute $w_i = y_i \wedge z_i$, for any $i \leq q(n)$, and finally, for the output gates of E_n, pick the sequence $z_1 w_1 z_2 w_2 \ldots z_n w_n$. Since q is a polynomial, the size of E_n is polynomially bounded in n. Moreover, if $|h(u)| = n$, then for any $i \leq q(n)$, the output of $E_n(h(u))$ on z_i decides if $i \leq l(h(u)) = |k(u)|$, and its output on y_i is $k(u)_i$ when $i \leq |k(u)|$. Therefore, if $i \leq |k(u)|$, we get the bit $k(u)_i$ on w_i, and when $i > |k(u)|$, we get zero on w_i. Therefore, the output of $E_n(h(u))$ is $pad_{2q(n)}(k(u))$. This means that E_n computes $k(u)$ from $h(u)$ using poly-size multi circuits and considering the required padding. This contradicts the security of (P, h, k). Therefore, (L, R) has no separator in $\mathbf{P/poly}$.

▶ **Corollary 16.** *If one-way protocols exist, then hard disjoint* **NP** *pairs also exist.*

▶ **Example 17.** Apply the above construction to the RSA and Diffie–Hellman (believed to be) one-way protocols, as presented in Example 13 and Example 14. Denote the corresponding disjoint **NP** pairs by $(\mathrm{RSA}_1, \mathrm{RSA}_0)$ and $(\mathrm{DH}_1, \mathrm{DH}_0)$, respectively. If these protocols are secure against $\mathbf{P/poly}$-adversaries, then $(\mathrm{RSA}_1, \mathrm{RSA}_0)$ and $(\mathrm{DH}_1, \mathrm{DH}_0)$ must be hard. ⌟

We have seen that the existence of a hard disjoint **NP** pair is an open problem. However, if we restrict the computational devices to monotone circuits, some known hardness theorems exist. First, we introduce the most important disjoint **NP** pair in this paper:

▶ **Example 18.** Let $n_0 \in \mathbb{N}$, and let $K(n)$ and $L(n)$ be functions computable in time $n^{O(1)}$ such that $L(n) < K(n) \leq n$ for all $n \geq n_0$. Following Example 2, any binary string of length $\binom{n}{2}$ can be interpreted as the adjacency matrix of a simple graph on n vertices. For any $k, l \in \mathbb{N}$, define $\mathrm{Clique}_{k,n}$ as the set of all such binary strings that, when interpreted as graphs, contain a k-clique, and define $\mathrm{Clique}_K = \bigcup_{n \geq n_0} \mathrm{Clique}_{K(n),n}$. Similarly, define $\mathrm{Color}_{l,n}$ as the set of graphs that are l-colorable, and set $\mathrm{Color}_L = \bigcup_{n \geq n_0} \mathrm{Color}_{L(n),n}$. It is clear that both Clique_K and Color_L are in **NP**. Since $L(n) < K(n)$, it is impossible for a graph on n vertices to simultaneously contain a $K(n)$-clique and be $L(n)$-colorable. Hence, $\mathrm{Clique}_K \cap \mathrm{Color}_L = \emptyset$, and the pair $(\mathrm{Clique}_K, \mathrm{Color}_L)$ forms a disjoint **NP** pair. As a concrete special case, fix a real number $0 < \epsilon < 1/3$ and define $K(n) = \lfloor n^{2/3} \rfloor$ and $L(n) = \lfloor n^{2/3-2\epsilon} \rfloor$. These functions are computable in time $n^{O(1)}$ and satisfy $L(n) < K(n)$ for sufficiently large n. We denote Clique_K and Color_L in this setting by Clique_ϵ and Color_ϵ, respectively. ⌟

Here is a well-known exponential lower bound on the size of monotone circuits separating Clique-Color pairs:

▶ **Theorem 19** (Razborov [33], Alon-Boppana [2])**.** *Let $l, k, n \in \mathbb{N}$ be numbers such that $3 \leq l < k$ and $\sqrt{l}\, k \leq \frac{n}{8 \log n}$. Any monotone circuit C on $\binom{n}{2}$ variables computing a separator of $(\mathrm{Clique}_{k,n}, \mathrm{Color}_{l,n})$ has size at least $\frac{1}{8} 2^{\frac{\sqrt{l}+1}{2}}$.*

▶ **Corollary 20.** *Let $0 < \epsilon < 1/3$ be a real number, and let $\{C_n\}_n$ be a sequence of monotone circuits computing a separator for the disjoint* **NP** *pair* $(\mathrm{Clique}_\epsilon, \mathrm{Color}_\epsilon)$*. Then, the size of $C_{\binom{n}{2}}$ is at least $2^{\Omega(n^{\frac{1}{3}-\epsilon})}$.*

Proof. Recall that to define $(\mathrm{Clique}_\epsilon, \mathrm{Color}_\epsilon)$, we used $K(n) = \lfloor n^{2/3} \rfloor$ and $L(n) = \lfloor n^{2/3-2\epsilon} \rfloor$. As $\sqrt{L(n)}\, K(n) \leq n^{1/3-\epsilon} n^{2/3} = n^{1-\epsilon}$ and $0 < \epsilon < 1/3$, we eventually have $\sqrt{L(n)}\, K(n) \leq \frac{n}{8 \log n}$ and $\lfloor n^{2/3-2\epsilon} \rfloor \geq 3$. Therefore, we can apply Theorem 19. ◀

4 Proof Systems

In the previous section, we saw that the propositional language is sufficiently powerful to describe finite domains and to compute over them. This encompasses both the *descriptive* and *computational* aspects of the propositional language. We now turn to its third fundamental aspect: *proofs*.

We use the term "proof" in the broadest possible sense. To explain this usage, consider two types of *finitary objects*, denoted by **F** and **Pr** (for example, formulas, circuits, inequalities, polynomials, or sequences of such objects). Then, for any set $L \subseteq \mathbf{F}$, we interpret some of the elements of **Pr** as proofs or witnesses certifying that an object from **F** belongs to L. The key insight, due to Cook and Reckhow [9], is that a meaningful notion of proof requires only two essential properties: *soundness and completeness* with respect to L, and *efficient verifiability*. The latter means that one can efficiently check whether a given object in **Pr** constitutes a valid proof for a given object in **F**. Since any finitary object can be encoded as a binary string, this idea can be formalized as follows:

▶ **Definition 21.** *Let $L \subseteq \{0,1\}^*$. A* proof system for L *is a polynomial-time decidable relation $P \subseteq \{0,1\}^* \times \{0,1\}^*$ such that, for every $w \in \{0,1\}^*$, we have $w \in L$ if and only if there exists $u \in \{0,1\}^*$ with $P(u,w)$. We call u a P-proof of w.*

▶ Remark 22. Usually, L is the set of binary representations of formulas in a logic, such as CPC or IPC. However, in some interesting cases, L may simply be a set of finitary objects. For example, L could consist of a sequence of inconsistent sets of clauses or formulas of a specific form within a certain fragment of a language. In such cases, and only in these cases, we place emphasis on the nature of the finitary objects used. Additionally, when working with inconsistent sets of clauses, "proofs" are usually referred to as *refutations*. ⌟

▶ **Example 23.** The usual Hilbert-style systems, natural deduction systems, and sequent calculi (with or without the cut rule) for classical and intuitionistic logics, as well as the modal logics in Table 1, are all proof systems for their respective logics. Since these systems are designed to be sound and complete, it suffices to observe that verifying whether a given string u is a proof of another string w involves checking if the sequence adheres to the local rules for constructing proofs and ultimately concludes with w. This verification process can be completed in polynomial time relative to the lengths of u and w. ⌟

In the following, we will introduce some proof systems for si or modal logics, and many for classical logic CPC. In each case, it is easy to verify that they are indeed proof systems for the corresponding logics.

▶ **Example 24** (Frege Systems)**.** Let L be a si or modal logic. An *inference system* P for L is a finite set of inference rules, each of the form:

$$\frac{\phi_1 \qquad \phi_2 \qquad \dots \qquad \phi_k}{\phi}$$

An inference rule with no assumptions is called an *axiom*. A P-proof π of a formula ϕ from the assumptions in Γ is defined as a sequence $\pi = \theta_1, \theta_2, \dots, \theta_n$ of formulas, such that $\theta_n = \phi$ and for each $i \leq n$, either $\theta_i \in \Gamma$, or θ_i is the result of an instance of one of the rules in P applied to the formulas θ_j for $j < i$. We write $\Gamma \vdash_P \phi$, when there is a P-proof of ϕ from Γ. A *Frege system* for the logic L is an inference system that is both *sound* and *strongly complete*; that is, $\vdash_P \phi$ implies $\phi \in L$, and $\phi_1, \dots, \phi_m \vdash_L \phi$ implies $\phi_1, \dots, \phi_m \vdash_P \phi$. A Frege proof system is called *standard* if it is *strongly sound*, meaning that if $\phi_1, \dots, \phi_m \vdash_P \phi$, then $\phi_1, \dots, \phi_m \vdash_L \phi$. Frege systems are formalizations of what is typically referred to as a Hilbert-style system. ⌟

▶ **Example 25** (Substitution Frege Systems)**.** Let L be a si or modal logic. By a (standard) *substitution Frege* system P for L, we mean an inference system extending a (standard) Frege system with the following *substitution rule*:

$$\frac{\phi(p_1, \dots, p_n)}{\phi(\psi_1, \dots, \psi_n)}$$

Note that the substitution rule is inherently different from the rules in a Frege system. ⌟

▶ **Example 26** (Bounded depth **LK**)**.** Define $\mathbf{LK}^u$ as a sequent calculus over $\mathcal{L}_b^u$, similar to **LK** (see Section 2), but with the conjunction, disjunction, and implication rules replaced by the following:

$$\frac{\Gamma, \phi_a \Rightarrow \Delta}{\Gamma, \bigwedge_{i\in I} \phi_i \Rightarrow \Delta}\, L\bigwedge_a \qquad \frac{\{\Gamma \Rightarrow \phi_i, \Delta\}_{i\in I}}{\Gamma \Rightarrow \bigwedge_{i\in I} \phi_i, \Delta}\, R\bigwedge$$

$$\frac{\{\Gamma, \phi_i \Rightarrow \Delta\}_{i\in I}}{\Gamma, \bigvee_{i\in I} \phi_i \Rightarrow \Delta}\, L\bigvee \qquad \frac{\Gamma \Rightarrow \phi_a, \Delta}{\Gamma \Rightarrow \bigvee_{i\in I} \phi_i, \Delta}\, R\bigvee_a$$

$$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg\phi \Rightarrow \Delta}\, L\neg \qquad \frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg\phi, \Delta}\, R\neg$$

where I is any finite set and $a \in I$ is arbitrary. It is straightforward to observe that $\mathbf{LK}^u$ is a proof system for CPC when defined over $\mathcal{L}_b^u$. Now, let $d \in \mathbb{N}$. By an $\mathbf{LK}_d$-proof, we refer to an $\mathbf{LK}^u$-proof in which the depth of every formula occurring in the proof is at most d. It is well known that $\mathbf{LK}_d$ is sound and complete with respect to valid sequents in which every formula has depth at most d. Consequently, we can consider $\mathbf{LK}_d$ as a proof system for CPC_d, the set of all valid $\mathcal{L}_b^u$-formulas of depth bounded by d. ⌟

▶ **Example 27** (Negation Normal **LK**). Define $\mathbf{LK}_{nn}$ as the sequent calculus over $\mathcal{L}_b$ consisting of the following axioms:

$$\frac{}{p \Rightarrow p} \qquad \frac{}{p, \neg p \Rightarrow} \qquad \frac{}{\Rightarrow p, \neg p} \qquad \frac{}{\bot \Rightarrow} \qquad \frac{}{\Rightarrow \top} \qquad \frac{}{\neg\top \Rightarrow} \qquad \frac{}{\Rightarrow \neg\bot}$$

along with the structural rules (including the cut), as well as the conjunction and disjunction rules of **LK**. Note that implication rules are not included. We define $\mathbf{LK}^{-}_{nn}$ analogously to $\mathbf{LK}_{nn}$, but without the cut rule. Both systems are sound and complete with respect to the valid sequents consisting of formulas in negation normal form. Therefore, we can consider $\mathbf{LK}_{nn}$ and $\mathbf{LK}^{-}_{nn}$ as proof systems for CPC_n, the set of all valid $\mathcal{L}_b$-formulas in negation normal form. ⌟

▶ **Example 28** (Resolution **R**). A *resolution refutation* π for a sequence of clauses $\mathcal{C}$ is a sequence $\pi = C_1, \ldots, C_n$ of clauses, where $C_n = \varnothing$, and each C_i is either a member of $\mathcal{C}$ or is derived from earlier clauses C_j and C_k (with $j, k < i$) via the resolution inference rule:

$$\frac{C \cup \{p\} \qquad D \cup \{\neg p\}}{C \cup D}$$

For example, the following refutation:

$$\dfrac{q \qquad \dfrac{p, \neg q \qquad \neg p}{\neg q}}{\varnothing}$$

is a refutation for the inconsistent sequence of clauses $\{p, \neg q\}$, $\{\neg p\}$, and $\{q\}$. Since resolution is complete for inconsistent sequences of clauses, we consider it a proof system for the set of inconsistent sequences of clauses. ⌟

Not all proof systems are dynamic or purely logical entities, as is traditionally assumed. Some proof systems are *static*, meaning they do not involve a sequence of inferential or transformational steps. Others are *non-logical*, operating not on formulas or sequents but on other mathematical structures, such as polynomial equalities or linear inequalities. In the following, we present examples that illustrate both of these alternative approaches: static and non-logical, individually as well as in combination.

▶ **Example 29** (Truth Table **TT**). The most straightforward example of a static proof system for CPC is the *truth table*. Formally, define $\mathbf{TT}(u, w)$ to hold if the string $w \in \{0,1\}^*$ is a code for a propositional formula ϕ, and u is the code for the truth table of ϕ that assigns the value 1 to ϕ under all possible truth assignments to its atomic variables. It is evident that **TT** is a proof system for CPC. ⌟

▶ **Example 30** (Cutting Planes **CP**). Interpret any clause $C = \{p_{i_1}, \ldots, p_{i_m}, \neg p_{j_1}, \ldots, \neg p_{j_n}\}$ as the linear inequality I_C defined by $\sum_{r=1}^{m} x_{i_r} - \sum_{s=1}^{n} x_{j_s} \geq 1 - n$. It is straightforward to see that an assignment $\bar{a} \in \{0,1\}$ satisfies C if and only if I_C holds after substituting $\bar{x}$ by $\bar{a}$. A **CP**-*refutation* for the sequence $\mathcal{C}$ of clauses is a sequence of the following rules, starting from the inequalities in $\{I_C \mid C \in \mathcal{C}\}$ and ending with the inequality $0 \geq 1$:

$$\frac{}{x_i \geq 0} \qquad \frac{}{-x_i \geq -1} \quad \text{(Axioms)} \qquad \frac{\sum_i a_i x_i \geq b \qquad \sum_i c_i x_i \geq d}{\sum_i (a_i + c_i) x_i \geq b + d} \quad \text{(Addition)}$$

$$\frac{\sum_i a_i x_i \geq b}{\sum_i c a_i x_i \geq cb} \quad \text{(Multiplication)} \qquad \frac{\sum_i a_i x_i \geq b}{\sum_i \frac{a_i}{c} x_i \geq \left\lfloor \frac{b}{c} \right\rfloor} \quad \text{(Division)}$$

where $c \geq 0$ in the multiplication rule and $c > 0$ and c divides all a_i's in the division rule. For instance, the following is a **CP**-refutation for the set of clauses $\{p_1, \neg p_2\}, \{\neg p_1\}, \{p_2\}$:

$$\dfrac{\dfrac{x_2 \geq 1 \qquad \dfrac{x_1 - x_2 \geq 0 \qquad -x_1 \geq 0}{-x_2 \geq 0}}{0 \geq 1}}{}$$

It is known that **CP** is sound and complete for unsatisfiable sequences of clauses. Soundness is easy by the connection between C and I_C. For completeness, the easiest proof is by simulating resolution refutations in **CP** to get completeness of **CP** from the completeness of **R** [21]. ⌟

▶ **Example 31** (Nullstellensatz **NS**). Let F be either $\mathbb{Q}$ or a finite field. We can translate any $\mathcal{L}_b$-formula $\phi(p_1, \ldots, p_n)$ into a polynomial $P_\phi \in F[x_1, \ldots, x_n]$ as follows:

$$P_\top = 0, \qquad P_\bot = 1, \qquad P_{p_i} = 1 - x_i,$$
$$P_{\neg\phi} = 1 - P_\phi, \qquad P_{\phi \vee \psi} = P_\phi \cdot P_\psi, \qquad P_{\phi \wedge \psi} = P_\phi + P_\psi - P_\phi \cdot P_\psi.$$

The key property of this translation is that for any assignment $\bar{a} \in \{0,1\}$ to the atomic variables in ϕ, ϕ holds if and only if $P_\phi(\bar{a}) = 0$. Therefore, given a finite sequence Φ of $\mathcal{L}_b$-formulas, the set of polynomials $\{P_\phi \mid \phi \in \Phi\} \cup \{x_i^2 - x_i \mid i \in \mathbb{N}\}$ has a common root in F if and only if Φ is satisfiable. We now define an **NS**-*refutation* of a sequence $\mathcal{C} = C_1, \ldots, C_r$ of clauses over variables $\{p_1, \ldots, p_s\}$ as a sequence of polynomials $g_1, \ldots, g_r, h_1, \ldots, h_s$ over F such that $\sum_{i=1}^r P_{\phi_i} g_i + \sum_{i=1}^s (x_i^2 - x_i) h_i = 1$, where $\phi_i = \bigvee C_i$. We claim that **NS** is a proof system for the set of unsatisfiable sequences of clauses. To establish the soundness and completeness of this system, we rely on Hilbert's Nullstellensatz, which asserts that a set $\mathcal{F}$ of polynomials over F has no common root in the algebraic closure of F if and only if there exist polynomials $g_1, \ldots, g_k$ over F such that $\sum_i f_i g_i = 1$ for some $f_i \in \mathcal{F}$. Since we are only concerned with Boolean assignments, passing to the algebraic closure of F does not affect the correctness of the system. For a concrete example, let $F = \mathbb{Q}$ and consider the unsatisfiable sequence of clauses $\{p_1, \neg p_2\}, \{\neg p_1\}, \{p_2\}$. This sequence translates into the following sequence of polynomials: $(1 - x_1)x_2, x_1, 1 - x_2$. Then, the sequence $(1, x_2, 1)$ is an **NS**-refutation since $(1 - x_1)x_2 \cdot 1 + x_1 \cdot x_2 + (1 - x_2) \cdot 1 = 1$. ⌟

▶ Remark 32. The previous example illustrates that the soundness and completeness of a proof system may rely on a mathematical theorem, in this case Hilbert's Nullstellensatz. Motivated by this observation, one can generalize the notion of a proof system by allowing any sufficiently strong, sound mathematical first-order theory T with a polynomial-time axiomatization to serve as a proof system for CPC. Specifically, we may define a proof of

a propositional formula ϕ as a first-order proof in T of the encoding of the sentence "the formula ϕ is a tautology." For example, taking $T = \mathsf{ZFC}$ yields a powerful proof system that permits the use of all of classical mathematics in a propositional proof. ⌟

4.1 Simulations and Relative Efficiency

For any language $L \subseteq \{0,1\}^*$, all proof systems for L are, by definition, sound and complete. That is, they prove exactly the same set of strings, i.e., the ones in L. This naturally raises the question: in what sense can we meaningfully distinguish between different proof systems for the same L? The key lies in their *efficiency*, that is, how succinct the proofs they produce are. For example, our collective experience is that the presence of the cut rule in $\mathbf{LK}$ allows for significantly shorter proofs compared to its cut-free counterpart $\mathbf{LK}^-$. This makes $\mathbf{LK}$ *stronger* than $\mathbf{LK}^-$. The following formalizes this notion of relative strength:

▶ **Definition 33.** *Let $L_1 \subseteq L_2$ and let P_1 and P_2 be proof systems for L_1 and L_2, respectively. We say that P_2* simulates *P_1 and write $P_2 \geq P_1$ (or $P_1 \leq P_2$) if there exists a polynomial p such that whenever $P_1(u,w)$ holds, there exists $v \in \{0,1\}^*$ with $|v| \leq p(|u|,|w|)$ such that $P_2(v,w)$ holds, for any strings $u, w \in \{0,1\}^*$. We say that P_1 and P_2 are* equivalent *and write $P_1 \equiv P_2$ if $L_1 = L_2$, $P_1 \geq P_2$, and $P_2 \geq P_1$.*

▶ Remark 34. The concrete proof systems we have defined so far operate over different fragments of different languages. However, there are canonical ways to translate formulas from one language into another, which sometimes allow us to map one fragment into another. For instance, a clause can naturally be viewed as a depth-one formula in $\mathcal{L}_b^u$, and arbitrary conjunctions and disjunctions in an $\mathcal{L}_b^u$-formula can be simulated using only binary connectives of $\mathcal{L}_b$. When comparing proof systems via simulations, we take such translations into account and consider the proof systems up to these transformations. ⌟

Some simulations are trivial. For instance, it is clear that $\mathbf{LK}_d \leq \mathbf{LK}_{d+1}$, as we can read a proof in $\mathbf{LK}_d$ as a proof in $\mathbf{LK}_{d+1}$. Similarly, using the above-mentioned transformations of formulas, we have $\mathbf{LK}_d \leq \mathbf{LK}$, and by transforming refutations into proofs, one can easily see that $\mathbf{R} \leq \mathbf{LK}_1$. In the following, we present less trivial simulations. For more, see Figure 3.

▶ **Theorem 35** ([16, 34])**.** *Let L be a si or modal logic. Then, any two standard Frege systems for L are equivalent. The same also holds for standard substitution Frege systems.*

Proof. We prove only the Frege case; the other is similar. Let $\mathbf{F}_1$ and $\mathbf{F}_2$ be two standard Frege systems for L. We show that $\mathbf{F}_2 \geq \mathbf{F}_1$. Let R be a rule of inference in $\mathbf{F}_1$ that derives ϕ from the assumptions $\phi_1, \ldots, \phi_k$. Since $\mathbf{F}_1$ is strongly sound, we have $\phi_1, \ldots, \phi_k \vdash_L \phi$. By the strong completeness of $\mathbf{F}_2$, there exists an $\mathbf{F}_2$-proof of ϕ from $\phi_1, \ldots, \phi_k$. Fix such a proof and denote it by π_R. Let $\pi = \psi_1, \ldots, \psi_m$ be an $\mathbf{F}_1$-proof of the formula ψ. To construct an $\mathbf{F}_2$-proof π' of ψ, proceed step-by-step: for each line ψ_i of the proof, find the rule $R \in \mathbf{F}_1$ and the substitution σ such that ψ_i is obtained from earlier formulas ψ_j via R and σ. Then, add

the $\mathbf{F}_2$-proof $\sigma(\pi_R)$ and continue with the next line of π. Since $\mathbf{F}_1$ has only finitely many rules, there are only finitely many $\mathbf{F}_2$-proofs π_R whose substitutions we need. As substitutions expand a proof only polynomially, the size of π' is polynomially bounded in the size of π. Hence, $\mathbf{F}_2 \geq \mathbf{F}_1$. Similarly, $\mathbf{F}_1 \geq \mathbf{F}_2$, which implies $\mathbf{F}_1 \equiv \mathbf{F}_2$. ◀

▶ Remark 36. Since any two standard Frege systems for L are equivalent, we may refer to *the* standard Frege system for L and denote it by *L-Frege* or L-$\mathbf{F}$. The same applies to standard substitution Frege systems for L, which we denote by L-$\mathbf{SF}$. ⌟

▶ **Theorem 37.** **LK** *(resp.* **LJ***) and* CPC*-Frege (resp.* IPC*-Frege) are equivalent. The same also holds for all modal calculi of Table 1 and their standard Frege systems.*

Proof. The standard proof of the equivalence between Hilbert-style systems and sequent calculi for these logics can be carried out in polynomial time, and therefore yields a polynomially bounded simulation. For more details, see [21]. ◀

▶ **Theorem 38.** $\mathbf{LK}_{nn}$ *and* **LK** *are equivalent. The same also holds for* $\mathbf{LK}^{-}_{nn}$ *and* $\mathbf{LK}^{-}$*.*

Proof. We only prove the first case, as the second is analogous. First, observe that any $\mathbf{LK}_{nn}$-proof is, in fact, an **LK**-proof, if we read $\neg\phi$ as $\phi \to \bot$. The only difference lies in the axioms: one needs to verify that the axioms $(p, \neg p \Rightarrow)$, $(\Rightarrow p, \neg p)$, $(\Rightarrow \neg\bot)$, and $(\neg\top \Rightarrow)$ are provable in **LK**, which is clearly the case. Note that the canonical **LK**-proofs of any instance of these axioms are of polynomial size in the end sequent. Hence, the entire transformation yields a polynomially bounded **LK**-proof. For the converse direction, it suffices to transform any **LK**-proof of a sequent $\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2$ into an $\mathbf{LK}_{nn}$-proof of the sequent $\Gamma_1^n, (\neg\Delta_1)^n \Rightarrow (\neg\Gamma_2)^n, \Delta_2^n$, where ϕ^n denotes the negation normal form of ϕ. Constructing such a transformation with polynomially bounded size is straightforward. ◀

4.2 Absolute Efficiency

In the previous subsection, we introduced the relative power of proof systems in terms of their efficiency. A natural question is: given a language L, is there an *absolutely efficient* proof system for L in which every $w \in L$ has a *short proof*? Of course, the term "short" must be understood relative to the length of w itself, since proofs are typically at least as long as the statements they establish.

▶ **Definition 39.** *A proof system P for a language L is called* polynomially bounded *(or* p-bounded*, for short) if there exists a polynomial p such that, for every $w \in L$, there exists $u \in \{0,1\}^*$ satisfying $P(u, w)$ and $|u| \leq p(|w|)$.*

▶ **Theorem 40.** *L has a p-bounded proof system iff $L \in \mathbf{NP}$.*

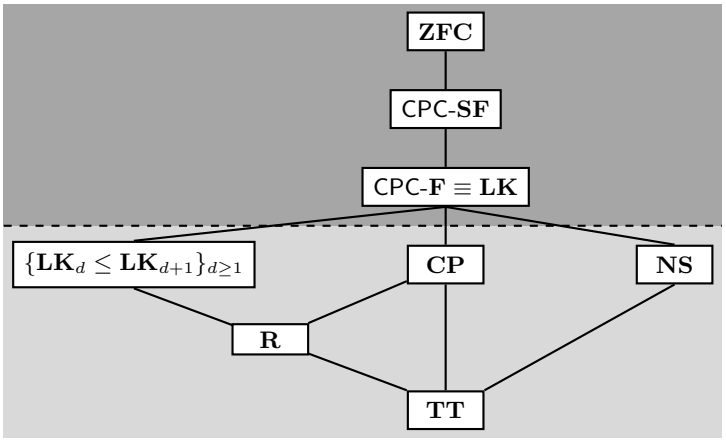


Figure 3 Some proof systems for CPC [21]. If there is a line between two proof systems, the higher one simulates the lower. For the proof systems in the region below the dashed line, an exponential lower bound is known. For those in the region above the dashed line, it is not even known whether they are not p-bounded.

Proof. Let P be a p-bounded proof system for a language L. Then, there exists a polynomial p such that for any $w \in \{0,1\}^*$, we have $w \in L$ iff there exists $u \in \{0,1\}^*$ such that $|u| \leq p(|w|)$ and $P(u,w)$. The right-hand side clearly defines an **NP** predicate, so we conclude that $L \in \mathbf{NP}$. For the converse direction, suppose $L \in \mathbf{NP}$. Then there exists a polynomial-time predicate R and a polynomial p such that $w \in L$ iff $\exists u\,[|u| \leq p(|w|) \wedge R(u,w)]$. Set $P(u,w) := R(u,w) \wedge (|u| \leq p(|w|))$. It is clear that P is a p-bounded proof system for L. ◀

▶ **Corollary 41.** CPC *(resp.* IPC *or any modal logic in Table 1) has a p-bounded proof system iff* $\mathbf{CoNP} = \mathbf{NP}$ *(resp.* $\mathbf{PSPACE} = \mathbf{NP}$*).*

Proof. As CPC is **CoNP**-complete, we have CPC $\in$ **NP** if and only if $\mathbf{CoNP} = \mathbf{NP}$. For IPC or any modal logic in Table 1, the argument is similar, using the **PSPACE**-completeness of these logics instead. ◀

It is widely believed that $\mathbf{NP} \neq \mathbf{CoNP}$ and $\mathbf{NP} \neq \mathbf{PSPACE}$. This implies that there can be no p-bounded proof system for CPC, IPC, or any modal logic in Table 1. However, since the non-existence of such systems is equivalent to these major complexity-theoretic conjectures, we currently lack the tools to prove it. Of course, this does not prevent us from proving that specific, well-known proof systems for these logics (such as Resolution, Cutting Planes, or L-Frege) are not p-bounded. The general setting for proving that a proof system for $L \subseteq \{0,1\}^*$ is not p-bounded is as follows. First, a definition:

▶ **Definition 42.** *Let $n_0 \in \mathbb{N}$ be a number, $L \subseteq \{0,1\}^*$, P be a proof system for L, and $\{w_n\}_{n \geq n_0}$ be a sequence of short binary strings in L. We say that the sequence $\{w_n\}_{n \geq n_0}$*

has a short P-proof *if there exists a sequence* $\{u_n\}_{n\geq n_0}$ *of binary strings such that* $P(u_n, w_n)$ *and* $|u_n| \leq p(n)$ *for all* $n \geq n_0$.

To show that P is not p-bounded, it suffices to find a sequence of short strings in L with no short P-proofs, since in any p-bounded proof system, every sequence of short binary strings in L must have a short P-proof. Arguably, the only known sufficiently general technique for constructing such sequences is *feasible interpolation*, as explained in the introduction. In the remainder of this chapter, we will explain this method and use it to show that some of the proof systems we have encountered so far are not p-bounded.

5 Interpolants and their Computation

In this section, we examine the complexity of computing a Craig interpolant for a given implication in classical logic. We then introduce alternative forms of Craig interpolation appropriate for intuitionistic logic and certain modal logics, and investigate the computational complexity of these interpolants as well. For more on the computation of interpolants, see Chapter 1 and Chapter 4.

5.1 Computing Craig Interpolants in Classical Logic

Let $\{\phi_n(\bar{p}, \bar{q}) \to \psi_n(\bar{p}, \bar{r})\}_{n\geq n_0}$ be a sequence of short classical tautologies. A sequence $\{C_n(\bar{p})\}_{n\geq n_0}$ of circuits computes[3] a Craig interpolant for $\{\phi_n(\bar{p}, \bar{q}) \to \psi_n(\bar{p}, \bar{r})\}_{n\geq n_0}$ when the formula $[C_n(\bar{p})]$ serves as a Craig interpolant for $\phi_n(\bar{p}, \bar{q}) \to \psi_n(\bar{p}, \bar{r})$, for all $n \geq n_0$. Spelling out, this means that for any Boolean assignment $\bar{a} \in \{0, 1\}$ to the atoms in $\bar{p}$, if $C_n(\bar{a}) = 0$, then $\neg\phi_n(\bar{a}, \bar{q})$ is a tautology, and if $C_n(\bar{a}) = 1$, then $\psi_n(\bar{a}, \bar{r})$ is a tautology. We say that $\{\phi_n(\bar{p}, \bar{q}) \to \psi_n(\bar{p}, \bar{r})\}_{n\geq n_0}$ has *an easy Craig interpolant* if there is a sequence $\{C_n\}_{n\geq n_0}$ of circuits that computes an interpolant such that the size of C_n is polynomially bounded by n, for any $n \geq n_0$. Otherwise, we say that it has *hard Craig interpolants*.

The most obvious way to compute a Craig interpolant for the sequence $\{\phi_n(\bar{p}, \bar{q}) \to \psi_n(\bar{p}, \bar{r})\}_{n\geq n_0}$ is to define the circuit $C_n(\bar{p})$ as either the formula $\bigvee_{\bar{b}\in\{0,1\}} \phi_n(\bar{p}, \bar{b})$ or the formula $\bigwedge_{\bar{c}\in\{0,1\}} \psi_n(\bar{p}, \bar{c})$. It is straightforward to verify that both of these formulas serve as interpolants. However, the size of these formulas grows exponentially with n. Naturally, we are keen to find more efficient methods to compute an interpolant, ideally using poly-size circuits. However, there are some indications that such computations may be inherently hard. Let us explain why, using disjoint **NP** pairs:

[3] The most natural approach to computing an interpolant is via an algorithm that takes a valid implication as input and returns an interpolant as output. This computational task, along with its connection to computational complexity, was first studied by Mundici [28–30]. However, for our purposes, it is too universal and not fine-grained enough to distinguish between implications with easily computable interpolants and those with hard ones. We therefore introduce the more local version presented here to allow for finer control.

▶ **Lemma 43.** *Let (L, R) be a disjoint* **NP** *pair, and let $\{\phi_n(\bar{p}, \bar{q})\}_n$ and $\{\psi_n(\bar{p}, \bar{r})\}_n$ be sequences of short propositional formulas representing L and R, respectively, as given by Theorem 6. Then the sequence $\{\phi_n(\bar{p}, \bar{q}) \to \neg\psi_n(\bar{p}, \bar{r})\}_n$ consists of short tautologies. Moreover, any sequence $\{C_n\}_n$ of circuits computing Craig interpolants for this sequence also computes a separator for the pair (L, R).*

Proof. By construction in Theorem 6, $w \in L$ (resp. $w \in R$) if and only if $\phi_n(w, \bar{q})$ (resp. $\psi_n(w, \bar{r})$) is satisfiable, where $n = |w|$. Since $L \cap R = \emptyset$, it follows that no w can simultaneously belong to both L and R, for any $w \in \{0,1\}^*$. Therefore, the conjunction $\phi_n(\bar{p}, \bar{q}) \wedge \psi_n(\bar{p}, \bar{r})$ is unsatisfiable, because any assignment that maps $\bar{p}$ to $w \in \{0,1\}^*$ would make both $\phi_n(w, \bar{q})$ and $\psi_n(w, \bar{r})$ satisfiable, which is a contradiction. Thus, we derive a sequence $\{\phi_n(\bar{p}, \bar{q}) \to \neg\psi_n(\bar{p}, \bar{r})\}_n$ of short tautologies. Now, suppose that $\{C_n(\bar{p})\}_n$ is a family of circuits that computes an interpolant for the sequence $\{\phi_n(\bar{p}, \bar{q}) \to \neg\psi_n(\bar{p}, \bar{r})\}_n$. Therefore, for any $w \in \{0,1\}^n$, if $C_n(w) = 0$, then $\neg\phi_n(w, \bar{q})$ is a tautology, implying that $w \in L^c$. Similarly, if $C_n(w) = 1$, then $w \in R^c$. Therefore, the set $S = \{w \in \{0,1\}^* \mid C_{|w|}(w) = 1\}$ acts as a separator for (L, R) and $\{C_n\}_n$ computes S. ◀

▶ **Corollary 44.** *If hard disjoint* **NP** *pairs exist or, in particular, one-way protocols exist, then there is a sequences of short propositional tautologies that has hard Craig interpolants.*

Proof. Assuming that hard disjoint **NP** pairs exist, the claim follows immediately from Lemma 43. For the one-way protocol variant, it suffices to apply Corollary 16. ◀

One can apply Lemma 43 to the disjoint **NP** pairs introduced earlier to obtain concrete sequences of tautologies that are believed to have hard Craig interpolants:

▶ **Example 45** (RSA and Diffie-Hellman Tautologies)**.** Let $\mathrm{RSA}^1_n(\bar{p}, \bar{q})$ and $\mathrm{RSA}^0_n(\bar{p}, \bar{r})$ be the propositional formulas that encode the disjoint **NP** pair $(\mathrm{RSA}_1, \mathrm{RSA}_0)$ defined in Example 17. Then, by Lemma 43, we obtain the sequence $\{\mathrm{RSA}^1_n(\bar{p}, \bar{q}) \to \neg\mathrm{RSA}^0_n(\bar{p}, \bar{r})\}_n$ of short tautologies. Similarly, if $\mathrm{DH}^1_n(\bar{p}, \bar{q})$ and $\mathrm{DH}^0_n(\bar{p}, \bar{r})$ are the propositional translations of the disjoint **NP** pair $(\mathrm{DH}_1, \mathrm{DH}_0)$, we obtain the sequence $\{\mathrm{DH}^1_n(\bar{p}, \bar{q}) \to \neg\mathrm{DH}^0_n(\bar{p}, \bar{r})\}_n$ of short tautologies. If the RSA and Diffie-Hellman protocols are secure against $\mathbf{P}/\mathbf{poly}$-adversaries, then the corresponding disjoint **NP** pairs must be hard, by Example 17. Therefore, both sequences would have hard Craig interpolants, by Lemma 43. ⌟

▶ **Example 46** (Clique-Coloring Tautologies)**.** Let $0 < \epsilon < 1/3$ be a real number. Using Examples 2 and 3, define the families $\mathrm{Clique}^\epsilon_n(\bar{p}, \bar{q})$ and $\mathrm{Color}^\epsilon_n(\bar{p}, \bar{r})$ of clauses, where $\bar{p}$ has $\binom{n}{2}$ atoms and describes a simple graph G on n vertices, $\bar{q}$ represents a $\lfloor n^{2/3} \rfloor$-clique inside G, and $\bar{r}$ represents an $\lfloor n^{2/3-2\epsilon} \rfloor$-coloring of G. Then, for large enough n, $\mathrm{Clique}^\epsilon_n(\bar{p}, \bar{q}) \cup \mathrm{Color}^\epsilon_n(\bar{p}, \bar{r})$ is an unsatisfiable sequence of clauses, as $\lfloor n^{2/3} \rfloor > \lfloor n^{2/3-2\epsilon} \rfloor$. Similarly, by transforming the clauses into disjunctions and slightly abusing notation, we can write $\mathrm{Clique}^\epsilon_n(\bar{p}, \bar{q}) \to \neg\mathrm{Color}^\epsilon_n(\bar{p}, \bar{r})$ as the corresponding tautology. ⌟

For the Clique-Coloring tautologies, we can unconditionally show that computing a Craig interpolant is hard if we restrict our computational device to monotone circuits:

▶ **Theorem 47.** *Let $0 < \epsilon < 1/3$ be a real number and let $\{C_n\}_n$ be a sequence of monotone circuits computing a Craig interpolant for $\{\text{Clique}^{\epsilon}_n(\bar{p},\bar{q}) \to \neg\text{Color}^{\epsilon}_n(\bar{p},\bar{r})\}_n$. Then, the size of C_n must be at least $2^{\Omega(n^{1/3-\epsilon})}$.*

Proof. Similar to the proof of Lemma 43, it is easy to see that C_n separates the pair $(\text{Clique}_{\lfloor n^{2/3} \rfloor, n}, \text{Color}_{\lfloor n^{2/3-2\epsilon} \rfloor, n})$. The lower bound follows from Theorem 19. ◀

▶ Remark 48. Any valid formula of the form $\phi(\bar{p},\bar{q}) \to \psi(\bar{p},\bar{r})$, where either $\phi(\bar{p},\bar{q})$ or $\psi(\bar{p},\bar{r})$ is monotone in $\bar{p}$, has a monotone Craig interpolant. This is a special case of a more general property of classical logic known as the *Lyndon interpolation property*; see Chapter 1 for more details. Using this property, since $\text{Clique}^{\epsilon}_n(\bar{p},\bar{q})$ is monotone in $\bar{p}$ (as shown in Example 2), a monotone interpolant exists. The content of Theorem 47 is therefore that any monotone circuit computing such an interpolant must have exponential size.

5.2 Non-classical Alternatives to Craig Interpolation

In classical logic, a Craig interpolant for $(\phi(\bar{p},\bar{q}) \to \psi(\bar{p},\bar{r})) \equiv (\neg\phi(\bar{p},\bar{q}) \vee \psi(\bar{p},\bar{r}))$ can be viewed as a mechanism that, given an assignment $\bar{a}$ to the shared variables $\bar{p}$, determines whether $\neg\phi(\bar{a},\bar{q})$ or $\psi(\bar{a},\bar{r})$ is a tautology. Thus, computing a Craig interpolant corresponds to deciding between these two disjuncts based on the assignment $\bar{a}$, a task that we have argued is computationally hard, by leveraging the hardness of disjoint **NP** pairs.

However, in many non-classical logics, there is often some form of the *disjunction property*. That is, if $\phi \vee \psi$ is derivable in the logic, then either ϕ or ψ must be derivable. In such settings, deciding between the disjuncts no longer depends on the assignment $\bar{a}$, making this task computationally trivial. Consequently, in non-classical contexts, we need to replace Craig interpolants with more refined notions that preserve the same intuitive purpose but are better suited to the specific logic in question. In this subsection, we follow [13] and introduce two such alternatives: one for the propositional language and one for the modal language.

▶ **Definition 49** (Propositional Disjunctive Interpolation)**.** *Let L be a si logic. An L-*propositional disjunctive interpolant *(L-*PDI*, for short) for $(\phi \to \psi \vee \theta) \in L$ is a pair (I, J) of formulas such that $V(I), V(J) \subseteq V(\phi)$, and the formulas $\phi \to (I \vee J)$, $I \to \psi$, and $J \to \theta$ are all in L. An L-*monotone propositional disjunctive interpolant *(L-*mPDI*, for short) is defined similarly, requiring additionally that ϕ, I, and J are all monotone. We say that L has the* (monotone) propositional disjunctive interpolation property *if any $(\phi \to \psi \vee \theta) \in L$ (resp. for monotone ϕ) has an L-*PDI *(resp. L-*mPDI*). We denote the propositional disjunctive interpolation property by* PDIP *and its monotone version by* mPDIP.[4]

[4] This notion is frequently employed in the literature on the proof complexity of non-classical logics as a

▶ Remark 50. For si logics, PDIP generalizes the disjunction property. To see why, it is enough to set $\phi = \top$. Then, both I and J must be variable-free, and hence they are intuitionistically equivalent to either $\top$ or $\bot$. Since $I \vee J$ is provable, one of I or J must be equivalent to $\top$. Therefore, either ψ or θ is provable. Using this observation, one may argue that PDIP is actually a generalization of the disjunction property rather than of Craig interpolation. This is somewhat true. However, as we will see, there is a connection between this notion and Craig interpolation. Therefore, it is probably fair to say that PDIP is a generalization of the classically-equivalent disjunctive form of Craig interpolation, where we work with disjunctions rather than implications. ⌟

▶ **Theorem 51.** IPC *has both* PDIP *and* mPDIP*.*

Proof. For PDIP, we prove the following apparently stronger claim: For any **LJ**-provable sequent $\Gamma \Rightarrow \phi \vee \psi$, there are formulas I and J such that $V(I), V(J) \subseteq V(\Gamma)$, and the sequents $(\Gamma \Rightarrow I \vee J)$ and $(I \Rightarrow \phi)$ and $(J \Rightarrow \psi)$ are all provable in **LJ**. The proof is by induction on a cut-free **LJ**-proof of the sequent $\Gamma \Rightarrow \phi \vee \psi$. The axiom case is easy. For the rules, we only explain the cases where the last rule is $(R\vee_1)$ or $(L\vee)$. The others are similar. For $(R\vee_1)$, assume that $\Gamma \Rightarrow \phi \vee \psi$ is proved by applying $(R\vee_1)$ on $\Gamma \Rightarrow \phi$. Then, it is enough to set $I = \bigwedge \Gamma$ and $J = \bot$. If the last rule is $(L\vee)$:

$$\frac{\Gamma, \theta_1 \Rightarrow \phi \vee \psi \qquad \Gamma, \theta_2 \Rightarrow \phi \vee \psi}{\Gamma, \theta_1 \vee \theta_2 \Rightarrow \phi \vee \psi}\, L\vee$$

using the induction hypothesis, there are I_i's and J_i's, for $i \in \{1, 2\}$ such that $V(I_i), V(J_i) \subseteq V(\Gamma \cup \{\theta_i\})$ and $(\Gamma, \theta_i \Rightarrow I_i \vee J_i)$, $(I_i \Rightarrow \phi)$, $(J_i \Rightarrow \psi)$ are all provable in **LJ**. Define $I = I_1 \vee I_2$ and $J = J_1 \vee J_2$. It is clear that $V(I), V(J) \subseteq V(\Gamma \cup \{\theta_1 \vee \theta_2\})$, $\mathbf{LJ} \vdash I \Rightarrow \phi$ and $\mathbf{LJ} \vdash J \Rightarrow \psi$. Moreover, it is easy to see that $(\Gamma, \theta_i \Rightarrow I \vee J)$ and hence $(\Gamma, \theta_1 \vee \theta_2 \Rightarrow I \vee J)$ is provable in **LJ**. This completes the proof of the claim and hence of PDIP. For the monotone case, it is easy to see that in the above construction, whenever Γ is monotone, so are I and J. ◀

▶ **Definition 52** (Modal Disjunctive Interpolation)**.** *Let L be a modal logic. An L-*modal disjunctive interpolant *(*L-MDI*, for short) for $(\phi \to \Box\psi \vee \Box\theta) \in L$ is a pair (I, J) of formulas such that $V(I), V(J) \subseteq V(\phi)$, and the formulas $\phi \to (I \vee J)$, $I \to \Box\psi$, and $J \to \Box\theta$ are all in L. An L-*monotone modal disjunctive interpolant *(*L-mMDI*, for short) is defined similarly, requiring additionally that ϕ, I, and J are all monotone. We say that L has the* (monotone) modal disjunctive interpolation property *if any $(\phi \to \Box\psi \vee \Box\theta) \in L$ (resp. for monotone ϕ) has an L-*MDI *(resp. L-*mMDI*). We denote the modal disjunctive interpolation property by* MDIP *and its monotone version by* mMDIP*.*

technical tool. We believe that, like Craig interpolation, both this notion and its modal counterpart are of independent interest and deserve further logical exploration. As a first step in this direction, we propose a name for them and briefly examine their existence.

▶ **Theorem 53.** *All modal logics in Table 1 have both* MDIP *and* mMDIP*.*

Proof. Let G be a sequent calculus in Table 1. Then, it is enough to use an induction on a cut-free G-proof of a sequent $\Gamma \Rightarrow \Box\Delta, \Lambda$, to show the apparently stronger claim that for any partition of Δ into $\Delta = \Delta_1, \ldots, \Delta_k$, there are formulas $I_1, \ldots, I_k$ such that $V(I_i) \subseteq V(\Gamma \cup \Lambda)$ and the sequents $(\Gamma \Rightarrow I_1, \ldots, I_k, \Lambda)$ and $(I_i \Rightarrow \Box\Delta_i)$, for $i \leq k$, are all provable in G. Moreover, if Γ is monotone and $\Lambda = \emptyset$, all I_i's are also monotone. The proof of this claim is easy and the crucial point that it uses is that the modal rules in G are all single-conclusion. ◀

Let L be a si logic and $\{\phi_n \to \psi_n \vee \theta_n\}_{n\geq n_0}$ be a sequence of short formulas in L. We say that a sequence $\{(C_n, D_n)\}_{n\geq n_0}$ of pairs of circuits computes an L-PDI of $\{\phi_n \to \psi_n \vee \theta_n\}_{n\geq n_0}$ if $([C_n], [D_n])$ is an L-PDI for $\phi_n \to \psi_n \vee \theta_n$, for every $n \geq n_0$. Similarly, if L is a modal logic, for a sequence $\{\phi_n \to \Box\psi_n \vee \Box\theta_n\}_{n\geq n_0}$ of short formulas in L, we say that a sequence $\{(C_n, D_n)\}_{n\geq n_0}$ of pairs of $\mathcal{L}_\Box$-circuits computes an L-MDI of $\{\phi_n \to \Box\psi_n \vee \Box\theta_n\}_{n\geq n_0}$ if $([C_n], [D_n])$ is an L-MDI of $\phi_n \to \Box\psi_n \vee \Box\theta_n$ for every $n \geq n_0$. Similar to the classical case, if there is a sequence of poly-size circuits for the computation, we say that the sequence $\{\phi_n \to \psi_n \vee \theta_n\}_{n\geq n_0}$ (resp. $\{\phi_n \to \Box\psi_n \vee \Box\theta_n\}_{n\geq n_0}$) has an easy L-PDI (resp. L-MDI). Otherwise, we say that the sequence has hard L-PDI's (resp. L-MDI's).

In the following, we show that, similar to the case of Craig interpolants in classical logic, the disjunctive interpolants can be also hard to compute. To prove this, we introduce a way to transform classical implications to theorems of IPC (resp. K) such that any computation of a propositional (resp. modal) disjunctive interpolant of the latter gives us the computation of a Craig interpolant of the former. First, let us transform the formulas:

▶ **Theorem 54.** *If* $\mathsf{CPC} \vdash \phi(\bar{p}, \bar{r}) \to \psi(\bar{p}, \bar{s})$*, then*

$$\mathsf{IPC} \vdash \bigwedge_i (p_i \vee \neg p_i) \to (\neg\phi(\bar{p}, \bar{r}) \vee \neg\neg\psi(\bar{p}, \bar{s})), \quad \mathsf{K} \vdash \bigwedge_i (\Box p_i \vee \Box\neg p_i) \to (\Box\neg\phi(\bar{p}, \bar{r}) \vee \Box\psi(\bar{p}, \bar{s})).$$

If either $\phi(\bar{p}, \bar{r})$ *or* $\psi(\bar{p}, \bar{s})$ *is monotone in* $\bar{p}$*, then*

$$\mathsf{IPC} \vdash \bigwedge_i (p_i \vee q_i) \to (\neg\phi(\neg\bar{p}, \bar{r}) \vee \neg\neg\psi(\bar{q}, \bar{s})), \quad \mathsf{K} \vdash \bigwedge_i (\Box p_i \vee \Box q_i) \to (\Box\neg\phi(\neg\bar{p}, \bar{r}) \vee \Box\psi(\bar{q}, \bar{s})).$$

Proof. We only prove the intuitionistic case; the modal case is similar. Let $\bar{p} = p_1, \ldots, p_m$. Then, for any $I \subseteq \{1, \ldots, m\}$, it is easy to see that either $\bigwedge_{i\in I} p_i \wedge \bigwedge_{i\notin I} \neg p_i \to \neg\phi_n(\bar{p}, \bar{r})$ or $\bigwedge_{i\in I} p_i \wedge \bigwedge_{i\notin I} \neg p_i \to \psi_n(\bar{p}, \bar{s})$ is a classical tautology. By Glivenko's theorem, we can see that either $\bigwedge_{i\in I} p_i \wedge \bigwedge_{i\notin I} \neg p_i \to \neg\phi_n(\bar{p}, \bar{r})$ or $\bigwedge_{i\in I} p_i \wedge \bigwedge_{i\notin I} \neg p_i \to \neg\neg\psi_n(\bar{p}, \bar{s})$ is an intuitionistic tautology, which implies $\mathsf{IPC} \vdash \bigwedge_{i=1}^m (p_i \vee \neg p_i) \to \neg\phi_n(\bar{p}, \bar{r}) \vee \neg\neg\psi_n(\bar{p}, \bar{s})$. For the monotone case, we explain only the situation where ψ is monotone in $\bar{p}$. By monotonicity, the formula $\bigwedge_{i=1}^m (\neg p_i \to q_i) \to (\psi(\neg\bar{p}, \bar{s}) \to \psi(\bar{q}, \bar{s}))$ is valid. Therefore, $\bigwedge_{i=1}^m (p_i \vee q_i) \to (\psi(\neg\bar{p}, \bar{s}) \to \psi(\bar{q}, \bar{s}))$ is also valid. As $\phi(\bar{p}, \bar{r}) \to \psi(\bar{p}, \bar{s})$ is valid, we obtain $\bigwedge_{i=1}^m (p_i \vee q_i) \to (\phi(\neg\bar{p}, \bar{r}) \to \psi(\bar{q}, \bar{s}))$ as valid. Now, for any $I \subseteq \{1, \ldots, m\}$, since $\bigwedge_{i\in I} p_i \wedge \bigwedge_{i\notin I} q_i \to \bigwedge_i (p_i \vee q_i)$ holds, we also have the validity of $\bigwedge_{i\in I} p_i \wedge \bigwedge_{i\notin I} q_i \to (\phi(\neg\bar{p}, \bar{r}) \to \psi(\bar{q}, \bar{s}))$. Therefore, either

$\bigwedge_{i\in I} p_i \to \neg\phi(\neg\bar{p},\bar{r})$ or $\bigwedge_{i\notin I} q_i \to \psi(\bar{q},\bar{s})$ is valid. By Glivenko's theorem, we can see that either $\bigwedge_{i\in I} p_i \to \neg\phi(\neg\bar{p},\bar{r})$ or $\bigwedge_{i\notin I} q_i \to \neg\neg\psi(\bar{q},\bar{s})$ is provable in IPC. Hence, $\mathsf{IPC} \vdash \bigwedge_{i=1}^{m}(p_i \vee q_i) \to (\neg\phi(\neg\bar{p},\bar{r}) \vee \neg\neg\psi(\bar{q},\bar{s}))$. ◀

▶ **Corollary 55.** *For any $0 < \epsilon < 1/3$, we have $\mathsf{IPC} \vdash \bigwedge_i(p_i \vee q_i) \to \neg\mathrm{Clique}_n^{\epsilon}(\neg\bar{p},\bar{r}) \vee \neg\mathrm{Color}_n^{\epsilon}(\bar{q},\bar{s})$ and $\mathsf{K} \vdash \bigwedge_i(\Box p_i \vee \Box q_i) \to \Box\neg\mathrm{Clique}_n^{\epsilon}(\neg\bar{p},\bar{r}) \vee \Box\neg\mathrm{Color}_n^{\epsilon}(\bar{q},\bar{s})$.*

Proof. As $\mathrm{Clique}_n^{\epsilon}(\bar{p},\bar{r})$ is monotone in $\bar{p}$, by Example 2, we can use Theorem 54. ◀

▶ **Theorem 56.** *Let L_1 be a si and L_2 be a consistent modal logic and $\{\phi_n(\bar{p},\bar{r}) \to \psi_n(\bar{p},\bar{s})\}_n$ be a sequence of short classical tautologies that has hard Craig interpolants. Then, the sequence $\{\bigwedge_i(p_i \vee \neg p_i) \to (\neg\phi_n(\bar{p},\bar{r}) \vee \neg\neg\psi_n(\bar{p},\bar{s}))\}_n$ have hard L_1-PDI's and the sequence $\{\bigwedge_i(\Box p_i \vee \Box\neg p_i) \to (\Box\neg\phi_n(\bar{p},\bar{r}) \vee \Box\psi_n(\bar{p},\bar{s}))\}_n$ have hard L_2-MDI's.*

Proof. For the si case, if $(C_n(\bar{p}), D_n(\bar{p}))$ computes an L_1-PDI for the formula $\bigwedge_i(p_i \vee \neg p_i) \to \neg\phi_n(\bar{p},\bar{r}) \vee \neg\neg\psi_n(\bar{p},\bar{s})$, then the formulas $\bigwedge_i(p_i \vee \neg p_i) \to [C_n(\bar{p})] \vee [D_n(\bar{p})]$, $[C_n(\bar{p})] \to \neg\phi(\bar{p},\bar{r})$, and $[D_n(\bar{p})] \to \neg\neg\psi(\bar{p},\bar{s})$ are in $L_1 \subseteq \mathsf{CPC}$ and hence valid. We claim that $D_n(\bar{p})$ computes a classical Craig interpolant for $\phi_n(\bar{p},\bar{r}) \to \psi_n(\bar{p},\bar{s})$. The reason is that, as $\bigwedge_i(p_i \vee \neg p_i) \to [C_n(\bar{p})] \vee [D_n(\bar{p})]$ is valid, so is $[C_n(\bar{p})] \vee [D_n(\bar{p})]$. Therefore, $\neg[C_n(\bar{p})] \to [D_n(\bar{p})]$ is valid. Moreover, we have the validity of $\phi_n(\bar{p},\bar{r}) \to \neg[C_n(\bar{p})]$, which implies $\phi_n(\bar{p},\bar{r}) \to [D_n(\bar{p})]$. As $[D_n(\bar{p})] \to \neg\neg\psi(\bar{p},\bar{s})$ and hence $[D_n(\bar{p})] \to \psi(\bar{p},\bar{s})$ is valid, the proof is complete. Finally, if C_n and D_n are of polynomial size in n, we can compute a classical Craig interpolant for $\{\phi_n(\bar{p},\bar{r}) \to \psi_n(\bar{p},\bar{s})\}_n$ by a sequence $\{D_n\}_n$ of poly-size circuits, which is a contradiction.

For the modal case, if $(C_n(\bar{p}), D_n(\bar{p}))$ computes an L_2-MDI for the formula $\bigwedge_i(\Box p_i \vee \Box\neg p_i) \to (\Box\neg\phi_n(\bar{p},\bar{r}) \vee \Box\psi_n(\bar{p},\bar{s}))$, then the formulas

$$\bigwedge_i(\Box p_i \vee \Box\neg p_i) \to [C_n(\bar{p})] \vee [D_n(\bar{p})], \quad [C_n(\bar{p})] \to \Box\neg\phi_n(\bar{p},\bar{r}), \quad \text{and} \quad [D_n(\bar{p})] \to \Box\psi_n(\bar{p},\bar{s})$$

are all in L_2. As L_2 is consistent, it is a subset of either the logic $\mathsf{CPC} + \Box\theta \leftrightarrow \theta$ or the logic $\mathsf{CPC} + \Box\theta$. For the first case, by applying the forgetful translation and the conservativity of $\mathsf{CPC} + \Box\theta \leftrightarrow \theta$ over CPC, we get the validity of $[C_n^f(\bar{p})] \vee [D_n^f(\bar{p})]$, $[C_n^f(\bar{p})] \to \neg\phi_n(\bar{p},\bar{r})$, and $[D_n^f(\bar{p})] \to \psi_n(\bar{p},\bar{s})$. Similar to the previous case, it is clear that $D_n^f(\bar{p})$ computes a classical Craig interpolant for $\phi_n(\bar{p},\bar{r}) \to \psi_n(\bar{p},\bar{s})$. In the other case, we do the same, but use the collapse translation to show that $D_n^c(\bar{p})$ computes a classical Craig interpolant for $\phi_n(\bar{p},\bar{r}) \to \psi_n(\bar{p},\bar{s})$. As the sizes of both $D_n^f(\bar{p})$ and $D_n^c(\bar{p})$ are less than or equal to the size of D_n, if the size of C_n and D_n are polynomially bounded in n, then, we can compute a classical Craig interpolant for $\{\phi_n(\bar{p},\bar{r}) \to \psi_n(\bar{p},\bar{s})\}_n$ by a sequence of poly-size circuits which is a contradiction. ◀

▶ **Corollary 57.** *Let L_1 be an si logic, and L_2 be a consistent modal logic. If hard disjoint* **NP** *pairs exist or, in particular, one-way protocols exist, then there is a sequence of short* IPC*-tautologies (resp.* K*-tautologies) with hard L_1-PDI's (resp. L_2-MDI's).*

As with classical Craig interpolants, restricting to monotone circuits yields an unconditional hardness result:

▶ **Theorem 58.** *Let $0 < \epsilon < 1/3$ be a real number, L_1 be an si logic, and L_2 be a consistent modal logic, and let $\{(C_n(\bar{p}, \bar{q}), D_n(\bar{p}, \bar{q}))\}_n$ be a sequence of pairs of monotone circuits (resp. $\mathcal{L}_\Box$-circuits) computing an L_1-PDI (resp. L_2-MDI) of $\bigwedge_i(p_i \vee q_i) \to \neg\mathrm{Clique}^\epsilon_n(\neg\bar{p}, \bar{r}) \vee \neg\mathrm{Color}^\epsilon_n(\bar{q}, \bar{s})$ (resp. $\bigwedge_i(\Box p_i \vee \Box q_i) \to \Box\neg\mathrm{Clique}^\epsilon_n(\neg\bar{p}, \bar{r}) \vee \Box\neg\mathrm{Color}^\epsilon_n(\bar{q}, \bar{s})$). Then, the size of D_n is at least $2^{\Omega(n^{1/3-\epsilon})}$.*

Proof. For the si case, similar to the proof of Theorem 56, it is easy to show that $\{D_n(\bar{\top}, \bar{p})\}_n$ computes a Craig interpolant for $\{\mathrm{Clique}^\epsilon_n(\bar{p}, \bar{r}) \to \neg\mathrm{Color}^\epsilon_n(\bar{p}, \bar{s})\}_n$. As D_n is monotone, so is $D_n(\bar{\top}, \bar{p})$. As the size of $D_n(\bar{\top}, \bar{p})$ is the same as the size of D_n, it is enough to use Theorem 47 to prove the lower bound. For the modal case, we claim that either $\{D^f_n(\bar{\top}, \bar{p})\}_n$ or $\{D^c_n(\bar{\top}, \bar{p})\}_n$ computes a Craig interpolant for $\{\mathrm{Clique}^\epsilon_n(\bar{p}, \bar{r}) \to \neg\mathrm{Color}^\epsilon_n(\bar{p}, \bar{s})\}_n$. To prove the claim, as any consistent modal logic is either a subset of $\mathsf{CPC} + \Box\theta \leftrightarrow \theta$ or $\mathsf{CPC} + \Box\theta$, there are two cases to consider: If $L_2 \subseteq \mathsf{CPC} + \Box\theta \leftrightarrow \theta$, then using an argument similar to that of the proof of Theorem 56, we can see that $\{D^f_n(\bar{\top}, \bar{p})\}_n$ computes a Craig interpolant for $\{\mathrm{Clique}^\epsilon_n(\bar{p}, \bar{r}) \to \neg\mathrm{Color}^\epsilon_n(\bar{p}, \bar{s})\}_n$. If $L_2 \subseteq \mathsf{CPC} + \Box\theta$, then $\{D^c_n(\bar{\top}, \bar{p})\}_n$ does the same task. As $D^f_n(\bar{\top}, \bar{p})$ or $D^c_n(\bar{\top}, \bar{p})$ are monotone and their size is smaller than or equal to that of D_n, the theorem follows from Theorem 47. ◀

6 Feasible Interpolation (Classical)

In the previous section, we have seen that computing a Craig interpolant can be a hard task. In such computation, the only information available to us is that the implication in question is a classical tautology. However, the computation may get significantly easier if we also have access to a proof of the implication. To get an idea, consider the cut-free proof system $\mathbf{LK}^-$. Given a proof in this system, it is relatively straightforward to *extract* an interpolant from the proof using the standard Maehara method. This observation leads to the following definition:

▶ **Definition 59** (Krajíček [23])**.** *A proof system P for* CPC *is said to have* (monotone) feasible interpolation *if there is a polynomial s such that for any P-proof $u \in \{0,1\}^*$ of an implication $\phi(\bar{p}, \bar{q}) \to \psi(\bar{p}, \bar{r})$ with the code w, (where ϕ or ψ is monotone in $\bar{p}$), there is a (monotone) circuit C of size bounded by $s(|u|, |w|)$ such that $[C]$ is a Craig interpolant for $\phi(\bar{p}, \bar{q}) \to \psi(\bar{p}, \bar{r})$.*

As discussed in the introduction, feasible interpolation for a proof system P can help to show that P is not p-bounded. The main idea is that if we know that computing a Craig interpolant is hard (e.g., using the existence of a disjoint **NP** pair) and if it gets easier with access to P-proofs, the P-proofs must be too long! This way, feasible interpolation transforms the harder task of finding a lower bound for proof length into a relatively easier task of finding an upper bound for the circuit size for Craig interpolants extracted from the proofs.

▶ **Theorem 60.** *Let P be a proof system for* CPC *that has feasible interpolation. Then, if $\{\phi_n \to \psi_n\}_n$ is a sequence of short tautologies that has hard Craig interpolants, then $\{\phi_n \to \psi_n\}_n$ does not have short P-proofs.*

Proof. Let $\{\pi_n\}_n$ be a sequence of short P-proofs of $\{\phi_n \to \psi_n\}_n$. Using feasible interpolation of P on π_n's, we get a sequence $\{C_n\}_n$ of circuits with size bounded by $(|\pi_n|+|\phi_n \to \psi_n|)^{O(1)} \leq n^{O(1)}$ to compute a Craig interpolant for $\{\phi_n \to \psi_n\}_n$, which is impossible. ◀

▶ Remark 61. As a method, feasible interpolation might appear restrictive in its capacity, as it can only demonstrate the non-existence of short P-proofs for sequences of short tautologies in *implicational form*. However, this restriction is not essential, as there exists a technique for reducing arbitrary tautologies to this form. For more information, see [10]. ⌟

▶ **Corollary 62.** *If hard disjoint* **NP** *pairs exist or in particular, if one-way protocols exist, any proof system for* CPC *with feasible interpolation is not p-bounded.*

Proof. It follows from Corollary 44 and Theorem 60. ◀

▶ **Corollary 63.** *If* $\mathbf{NP} \not\subseteq \mathbf{P/Poly}$*, then no proof system for* CPC *with feasible interpolation is p-bounded.*

Proof. Assume that P is a p-bounded proof system for CPC that has feasible interpolation. By Corollary 41, we have $\mathbf{NP} = \mathbf{CoNP}$. Hence, $\mathbf{NP} \cap \mathbf{CoNP} = \mathbf{NP} \not\subseteq \mathbf{P/Poly}$. This implies the existence of a hard disjoint **NP** pair. Now, use Corollary 62 to reach a contradiction. ◀

For the monotone case, as we have a lower bound for monotone circuits computing a Craig interpolant for Clique-Coloring tautologies by Theorem 47, we can prove an unconditional lower bound for the proof length:

▶ **Theorem 64.** *Let $0 < \epsilon < 1/3$ be a real number. If a proof system P for* CPC *has monotone feasible interpolation, then any P-proof of $\mathrm{Clique}^{\epsilon}_n(\bar{p}, \bar{q}) \to \neg\mathrm{Color}^{\epsilon}_n(\bar{p}, \bar{r})$ has size at least $2^{\Omega(n^{1/3-\epsilon})}$. Hence, no proof system for* CPC *with monotone feasible interpolation is p-bounded.*

Proof. The argument is similar to that of Theorem 60. ◀

In the next subsection, we will use Theorem 64 to show that some of the proof systems for CPC introduced earlier are not p-bounded. Before proceeding in that direction, let us first highlight another interesting aspect related to feasible interpolation. As we will see later, there are certain proof systems for which we believe feasible interpolation does not hold. Since feasible interpolation is a key technique for proving lower bounds, its absence might initially seem like a setback. However, the lack of feasible interpolation actually reveals an interesting property of a proof system: its *non-automatability*:

▶ **Definition 65** (Automatability)**.** *A proof system P for* CPC *is said to be* automatable *if there exists an algorithm M and a polynomial k such that, reading any classical tautology ϕ, M finds a P-proof of ϕ in time $k(|\pi_\phi|, |\phi|)$, where π_ϕ denotes the shortest P-proof of ϕ.*

▶ Remark 66. To automate a proof system P for CPC, we naturally seek an algorithm that, given a tautology ϕ, produces a P-proof of ϕ. Ideally, we would like this algorithm to run in polynomial time. However, any such algorithm must spend at least $|\pi_\phi|$ steps, which may be very large. Therefore, the best we can reasonably ask for is that the algorithm runs in time polynomial in both the lengths of π_ϕ and ϕ, and not just in the length of ϕ. ⌟

The following Theorem extends Theorem 41 by incorporating the automatability condition, thereby strengthening the conclusion from $\mathbf{NP} = \mathbf{CoNP}$ to $\mathbf{NP} = \mathbf{P}$.

▶ **Theorem 67.** CPC *has a proof system that is both p-bounded and automatable iff* $\mathbf{NP} = \mathbf{P}$.

Proof. If $\mathbf{NP} = \mathbf{P}$, then $\mathsf{CPC} \in \mathbf{P}$. Consider the predicate $P(w, u)$ defined as follows: $P(w, u)$ holds if $w = 0$ and u is the code of a valid formula. Since $\mathsf{CPC} \in \mathbf{P}$, the predicate P is polynomial-time computable and thus serves as a proof system for CPC. Moreover, every P-proof has length one, so this proof system is clearly p-bounded and automatable.

Conversely, let P be a proof system for CPC that is both p-bounded and automatable. We show that this implies $\mathsf{CPC} \in \mathbf{P}$, and hence $\mathbf{NP} = \mathbf{P}$. Since P is p-bounded, any valid formula ϕ has a P-proof of size polynomial in $|\phi|$. In other words, the size of the shortest P-proof of ϕ, denoted by $|\pi_\phi|$, is bounded by $|\phi|^{O(1)}$. Therefore, by the automatability of P, there exists an algorithm that finds a P-proof of ϕ in time polynomial in $|\phi|$. Finally, to decide the validity of ϕ, it suffices to run this polynomial-time algorithm: if it outputs a P-proof, then ϕ is valid; otherwise, it is not. Thus, $\mathsf{CPC} \in \mathbf{P}$ and $\mathbf{NP} = \mathbf{P}$. ◀

To state the connection between feasible interpolation and automatability, we need the following definition:

▶ **Definition 68.** *A proof system P for* CPC *is called* substitutable *if there is a polynomial l such that for any P-proof $u \in \{0,1\}^*$ of $\phi(\bar{p}, \bar{q})$ and any assignment $\bar{a} \in \{0,1\}$, there is a P-proof for $\phi(\bar{p}, \bar{a})$ of size bounded by $l(|u|, |\phi|)$. The proof system P is called* closed under variable-free modus ponens*, if there is a polynomial m such that for any variable-free tautology ϕ and any P-proof $u \in \{0,1\}^*$ of $\phi \to \psi$, there is a P-proof for ψ of size bounded by $m(|u|, |\phi \to \psi|)$.*

For instance, it is easy to see that the proof systems $\mathbf{LK}_d$, **LK** and CPC-**SF** are substitutable and closed under variable-free modus ponens, for any $d \geq 0$.

▶ **Theorem 69.** *Let P be a substitutable and closed under variable-free modus ponens proof system for* CPC*. If P does not have feasible interpolation, then it is not automatable.*

Proof. Let P be automatable and M and k be the corresponding algorithm and polynomial. Moreover, let l and m be the polynomials witnessing that P is substitutable and closed under variable-free modus ponens, respectively. W.l.o.g., we can assume that k, l, and m are all monotone. We first prove that for any P-proof u of a tautology in the form $\phi(\bar{p}, \bar{q}) \to \psi(\bar{p}, \bar{r})$ with the code w and any assignment $\bar{a} \in \{0,1\}$ for $\bar{p}$, if $\phi(\bar{a}, \bar{q})$ is satisfiable, then $\psi(\bar{a}, \bar{r})$ has

a P-proof of size bounded by $m(l(|u|,|w|),|w|)$. To prove that, let $\bar{b} \in \{0,1\}$ be a satisfying assignment for $\phi(\bar{a},\bar{q})$. Then, as P is substitutable, there is a P-proof of $\phi(\bar{a},\bar{b}) \to \psi(\bar{a},\bar{r})$ of size bounded by $l(|u|,|w|)$. Then, as $\phi(\bar{a},\bar{b})$ is a variable-free tautology and P is closed under variable-free modus ponens, there is a P-proof for $\psi(\bar{a},\bar{r})$ of size bounded by $m(l(|u|,|w|),|w|)$. This completes the proof of the claim.

Now, define the algorithm N as follows. It reads u, w, and v as three inputs. There are two cases. First, if u is a P-proof of a formula in the form $\phi(\bar{p},\bar{q}) \to \psi(\bar{p},\bar{r})$ with the code w and v is an assignment $\bar{a} \in \{0,1\}$ for $\bar{p}$, then N runs M on $\psi(\bar{a},\bar{r})$ for $k(m(l(|u|,|w|),|w|),|w|)$ many steps. If M halts before this bound, N returns one; otherwise, zero. Second, if the inputs are not in the mentioned form, N outputs zero. It is clear that N runs in polynomial time. We claim that in the first case, N outputs the value of a Craig interpolant for the implication $\phi(\bar{p},\bar{q}) \to \psi(\bar{p},\bar{r})$, i.e., that if N outputs one, then $\psi(\bar{a},\bar{r})$ is a tautology, and if it outputs zero, then $\neg\phi(\bar{a},\bar{q})$ is a tautology. To prove this, if M halts in $k(m(l(|u|,|w|),|w|),|w|)$ many steps, it finds a P-proof for $\psi(\bar{a},\bar{r})$, which implies that $\psi(\bar{a},\bar{r})$ is a tautology. If it returns zero, we claim that $\psi(\bar{a},\bar{r})$ does not have a P-proof of size bounded by $m(l(|u|,|w|),|w|)$. There are two cases to consider. If $\psi(\bar{a},\bar{r})$ is not a tautology, it has no P-proof and we are done. If it is a tautology, then the time M needs to halt on $\psi(\bar{a},\bar{r})$ is bounded by $k(|\pi_{\psi(\bar{a},\bar{r})}|,|w|)$, where $\pi_{\psi(\bar{a},\bar{r})}$ is the shortest P-proof of $\psi(\bar{a},\bar{r})$. If a P-proof of $\psi(\bar{a},\bar{r})$ with length bounded by $m(l(|u|,|w|),|w|)$ exists, we must have $|\pi_{\psi(\bar{a},\bar{r})}| \leq m(l(|u|,|w|),|w|)$, which implies $k(|\pi_{\psi(\bar{a},\bar{r})}|,|w|) \leq k(m(l(|u|,|w|),|w|),|w|)$. This means that M would have halted before reaching $k(m(l(|u|,|w|),|w|),|w|)$ many steps, which is a contradiction. Now, as $\psi(\bar{a},\bar{r})$ does not have a P-proof of size bounded by $m(l(|u|,|w|),|w|)$, by the above observation, we conclude that $\phi(\bar{a},\bar{q})$ is unsatisfiable. Hence, $\neg\phi(\bar{a},\bar{q})$ is a tautology. This completes the proof that N computes a Craig interpolant of $\phi(\bar{p},\bar{q}) \to \psi(\bar{p},\bar{r})$.

Finally, as N runs in polynomial time, there is a sequence $\{C_n\}_n$ of poly-size circuits such that $N(u,w,v) = C_{|u|+|w|+|v|}(u,w,v)$. For any P-proof u of $\phi(\bar{p},\bar{q}) \to \psi(\bar{p},\bar{r})$ with the code w and n many atoms in $\bar{p}$, define the circuit $D_{u,w}$ by $D_{u,w}(x_1,\ldots,x_n) = C_{|u|+|w|+n}(u,w,x_1,\ldots,x_n)$. It is clear that $[D_{u,w}]$ is an interpolant of $\phi(\bar{p},\bar{q}) \to \psi(\bar{p},\bar{r})$ and as $n \leq |w|$, its size is bounded by $(|u|+|w|)^{O(1)}$. Hence, P has feasible interpolation. ◀

To use Theorem 69, we need a way to prove that a proof system P for CPC fails to have feasible interpolation. For that purpose, it is enough to use Theorem 60 in a backward manner. Combining with Theorem 69, we have:

▶ **Corollary 70.** *Let P be a proof system for* CPC *and $\{\phi_n \to \psi_n\}_n$ be a sequence of short tautologies that has hard Craig interpolants. Then, if $\{\phi_n \to \psi_n\}_n$ has a short P-proof, P does not have feasible interpolation. If P is also substitutable and closed under variable-free modus ponens, then it is not automatable.*

Note that Corollary 70 suggests that if proof systems become strong enough to provide short proofs for implications with hard Craig interpolants, then feasible interpolation starts to fail. In the next subsection, we will present some instances of such strong proof systems

that are believed to lack feasible interpolation and, as they are substitutable and closed under variable-free modus ponens, also non-automatable.

6.1 Applications

In this subsection, we apply the concept of feasible interpolation to demonstrate that certain weak concrete proof systems are not p-bounded. We also use the failure of feasible interpolation to show that some strong concrete proof systems are non-automatable.

▶ **Theorem 71** (Krajíček [23]). $\mathbf{LK}^-_{nn}$ *has both feasible and monotone feasible interpolation.*

Proof. By recursion on an $\mathbf{LK}^-_{nn}$-proof π of $\Gamma \Rightarrow \Delta$, one can easily construct a circuit C on the variables in $V(\Gamma) \cap V(\Delta)$ such that both $\Gamma \Rightarrow [C]$ and $[C] \Rightarrow \Delta$ are valid, and the size of C is bounded by $|\pi|^{O(1)}$. Since the system $\mathbf{LK}^-_{nn}$ lacks negation and implication rules, it is clear that if $\Gamma(\bar{p}, \bar{q})$ or $\Delta(\bar{p}, \bar{r})$ is monotone in $\bar{p}$, then the constructed circuit will only use conjunctions and disjunctions over the atoms in $\bar{p}$. Hence, C will be monotone. See Chapter 5 for the detailed inductive construction. ◀

One can use the (monotone) feasible interpolation for $\mathbf{LK}^-_{nn}$ to prove the same for resolution. It is enough to simulate $\mathbf{R}$ by $\mathbf{LK}^-_{nn}$. For any clause C in variables $\bar{p} \cup \bar{q}$, define $C^{\bar{p}}$ as the sub-clause of C consisting of the literals constructed from $\bar{p}$, and let d be the dualization operator, i.e., $d(p) = \neg p$, $d(\neg p) = p$, and $C^d = \{l^d \mid l \in C\}$ for any clause C. Then:

▶ **Lemma 72** (Krajíček [23]). *There is a polynomial $P(n)$ such that for any resolution refutation π of the clause E from the clauses $\{C_i(\bar{p}, \bar{q})\}_i \cup \{D_j(\bar{p}, \bar{r})\}_j$, there is an $\mathbf{LK}^-_{nn}$-proof of size at most $P(|\pi|)$ of a sequent $\Gamma, d(E^{\bar{q}}) \Rightarrow \Delta, E^{\bar{p},\bar{r}}$ where Γ consists of the formulas $\bigvee C_i$ or unsatisfiable formulas in $\bar{q}$, and Δ consists of the formulas $\bigwedge \neg D_j$ or tautologies in $\bar{p}$ and $\bar{r}$.*

Proof. The construction of the claimed $\mathbf{LK}^-_{nn}$-proof is by recursion on the structure of π. If π is just a clause $C_i(\bar{p}, \bar{q})$ or $D_j(\bar{p}, \bar{r})$, it is enough to use the canonical $\mathbf{LK}^-_{nn}$-proofs for either $(\bigvee C_i, d(C_i^{\bar{q}}) \Rightarrow C_i^{\bar{p}})$ or $(\Rightarrow D_j, \bigwedge \neg D_j)$. For the resolution rule, let the last rule in π be the resolution rule applied on $C \cup \{s\}$ and $D \cup \{\neg s\}$ to prove $C \cup D$. Then there are two cases to consider: either $s \in \bar{q}$ or $s \in \bar{p} \cup \bar{r}$. In the first case, by the induction hypothesis, we have $\mathbf{LK}^-_{nn}$-proofs of $\Gamma, d(C^{\bar{q}}), \neg s \Rightarrow C^{\bar{p},\bar{r}}, \Delta$ and $\Gamma', d(D^{\bar{q}}), s \Rightarrow D^{\bar{p},\bar{r}}, \Delta'$. Therefore, using the rules of $\mathbf{LK}^-_{nn}$, we can easily reach the sequent $\Gamma, \Gamma', d(C^{\bar{q}}), d(D^{\bar{q}}), s \vee \neg s \Rightarrow C^{\bar{p},\bar{r}}, D^{\bar{p},\bar{r}}, \Delta, \Delta'$. If $s \in \bar{p} \cup \bar{r}$, then we have $\Gamma, d(C^{\bar{q}}) \Rightarrow s, C^{\bar{p},\bar{r}}, \Delta$ and $\Gamma', d(D^{\bar{q}}) \Rightarrow D^{\bar{p},\bar{r}}, \neg s, \Delta'$. Hence, in $\mathbf{LK}^-_{nn}$, we can easily reach the sequent $\Gamma, \Gamma', d(C^{\bar{q}}), d(D^{\bar{q}}) \Rightarrow C^{\bar{p},\bar{r}}, D^{\bar{p},\bar{r}}, s \wedge \neg s, \Delta, \Delta'$. This completes the simulation. Moreover, note that the transformation is polynomially bounded. ◀

▶ **Corollary 73** (Krajíček [23]). $\mathbf{R}$ *has both feasible and monotone feasible interpolation.*

Proof. Let π be a resolution refutation for $\{C_i(\bar{p},\bar{q})\}_i \cup \{D_j(\bar{p},\bar{r})\}_j$. Then, by Lemma 72, there is an $\mathbf{LK}^-_{nn}$-proof of size $|\pi|^{O(1)}$ for $\Gamma \Rightarrow \Delta$, where Γ consists of the formulas $\bigvee C_i$ or unsatisfiable formulas in $\bar{q}$ and Δ consists of the formulas $\bigwedge \neg D_j$ or tautologies in $\bar{p}$ and $\bar{r}$. By Theorem 71, there is a circuit of size $|\pi|^{O(1)}$ to compute a Craig interpolant for $\Gamma \Rightarrow \Delta$. As unsatisfiable formulas on the right hand side and tautologies on the left hand side of a sequent has no effect on its interpolant, we get a circuit of size $|\pi|^{O(1)}$ to compute an interpolant for $\{C_i(\bar{p},\bar{q})\}_i \cup \{D_j(\bar{p},\bar{r})\}_j$. For the monotone case, assume that $\{C_i\}_i$ is monotone in $\bar{p}$. Since Γ consists of the formulas $\bigvee_i C_i$ and $s \vee \neg s$ for $s \in \bar{q}$, and C_i's are monotone in $\bar{p}$, the multiset Γ is also monotone in $\bar{p}$. Therefore, by the monotone feasible interpolation property of $\mathbf{LK}^-_{nn}$, we obtain a monotone circuit to interpolate. ◀

▶ Remark 74. There are also direct proofs that extract Craig interpolants from resolution refutations. See [21] and Chapter 1 for a comprehensive explanation. ⌟

▶ **Corollary 75** (Krajíček [23])**.** *Let $0 < \epsilon < 1/3$ be a real number. Then, any $\mathbf{LK}^-$-proof of $\mathrm{Clique}^\epsilon_n(\bar{p},\bar{q}) \to \neg\mathrm{Color}^\epsilon_n(\bar{p},\bar{r})$ or any resolution refutation of $\mathrm{Clique}^\epsilon_n(\bar{p},\bar{q}) \cup \mathrm{Color}^\epsilon_n(\bar{p},\bar{r})$ has size at least $2^{\Omega(n^{1/3-\epsilon})}$. Consequently, $\mathbf{LK}^-$ and $\mathbf{R}$ are not p-bounded.*

Proof. By Theorem 71 and Corollary 73, both $\mathbf{LK}^-_{nn}$ and $\mathbf{R}$ have monotone feasible interpolation. Thus, we can apply Corollary 64 to obtain the claimed lower bounds for $\mathbf{LK}^-_{nn}$ and $\mathbf{R}$. Then, we can replace $\mathbf{LK}^-_{nn}$ by $\mathbf{LK}^-$ by the equivalence $\mathbf{LK}^- \equiv \mathbf{LK}^-_{nn}$ from Theorem 38. ◀

▶ Remark 76. For many proof systems for CPC, such as **CP** and **NS**, the monotone feasible interpolation technique can be used to demonstrate that they are not p-bounded. However, in each case, it is necessary to transition from monotone circuits to more sophisticated computational models and establish a lower bound on their ability to separate certain disjoint **NP** pairs. For example, in the case of Cutting Planes, Pudlák employs monotone real circuits and proves that **CP** admits monotone feasible interpolation with respect to these circuits. A real monotone circuit is one that can use any monotone function on real numbers as gates. Pudlák then establishes an exponential lower bound for the Clique-Color disjoint **NP** pair, concluding that $\mathrm{Clique}^\epsilon_n(\bar{p},\bar{q}) \cup \mathrm{Color}^\epsilon_n(\bar{p},\bar{r})$ admits exponentially long refutations in **CP**, and therefore, **CP** is not p-bounded [31]. For more on these variations, see [21]. ⌟

▶ Remark 77. A natural question is whether feasible interpolation is *complete* for showing the hardness of the hard implicational tautologies; that is, whether for every sequence of short implicational tautologies lacking short P-proofs, feasible interpolation can always be used to demonstrate this non-existence. Thanks to an elegant example constructed by Erfan Khaniki (personal communication), we know that there exists a sequence of short valid implications with an easy Craig interpolant, yet no short resolution refutation. This shows that feasible interpolation, as a method, is incomplete. While presenting this example is beyond the scope of this chapter, we felt it worthwhile to mention this incompleteness phenomenon, even informally. ⌟

To prove the lack of (monotone) feasible interpolation by applying Theorem 64 or Corollary 70, we must provide short proofs for implications with hard Craig interpolants. To that goal, we need some upper bound results to show that certain proof systems are sufficiently strong:

▶ **Theorem 78.** *There exist $k, l \in \mathbb{N}$ such that for any sufficiently large $n \in \mathbb{N}$, we have:*

- *[21, 26]* $\mathrm{PHP}_n^{m(n)}$ *have* $\mathbf{LK}_2$*-proofs of size less than* $2^{l(\log n)^k}$*, if* $n^2 \leq m(n) \leq 2^{(\log n)^k}$*.*
- *[25]* $\mathrm{RSA}_n^1(\bar{p}, \bar{q}) \to \neg\mathrm{RSA}_n^0(\bar{p}, \bar{r})$ *has* CPC-**SF** *proofs of size less than* n^k*.*
- *[5, 6]* $\mathrm{DH}_n^1(\bar{p}, \bar{q}) \to \neg\mathrm{DH}_n^0(\bar{p}, \bar{r})$ *has* $\mathbf{LK}$*-proofs of size less than* n^k*. Moreover, there exists* $d \geq 2$ *such that it also has* $\mathbf{LK}_d$*-proofs of size less than* 2^{n^ϵ}*, for some* $\epsilon > 0$*.*

Proof. These upper bounds are typically obtained by reasoning in a non-propositional framework (often an appropriate fragment of arithmetic), and then translating the argument (fragment) into propositional proofs. For the first item, one should prove the pigeonhole principle in its usual first-order form via a weak form of induction [21]. For the other two items, which correspond to disjoint **NP** pairs, the main reason why the implications are tautologies stems from the disjointness of the **NP** pairs, itself a consequence of semi-injectivity in the corresponding one-way protocols. Thus, to obtain propositional proofs for these implications, it suffices to express the earlier proofs of semi-injectivity for RSA and Diffie–Hellman protocols in a propositional form within the corresponding propositional proof systems [21]. ◀

Now, we are ready to prove the lack of (monotone) feasible interpolation:

▶ **Theorem 79** (Krajíček [23])**.** *Let P be a proof system for* CPC *such that $P \geq \mathbf{LK}_2$. Then, P does not have monotone feasible interpolation.*

Proof. By Theorem 78, there exists $k \in \mathbb{N}$ such that the formula $\mathrm{PHP}_m^{l(m)}$ has an $\mathbf{LK}_2$-proof of size less than $2^{(\log m)^k}$, provided $m^2 \leq l(m) \leq m^3$. We use this pigeonhole principle to construct an $\mathbf{LK}_2$-proof of the tautology $\mathrm{Clique}_n^{1/6}(\bar{p}, \bar{q}) \to \neg\mathrm{Color}_n^{1/6}(\bar{p}, \bar{r})$ of size $2^{(\log n)^{O(1)}}$. We omit the detailed construction, but the main idea is as follows: Assume $\mathrm{Clique}_n^{1/6}(\bar{p}, \bar{q}) \wedge \mathrm{Color}_n^{1/6}(\bar{p}, \bar{r})$. Then, by Examples 2 and 3, $\bar{q}$ and $\bar{r}$ effectively describe an injective mapping from the $\lfloor n^{2/3} \rfloor$ vertices of an $\lfloor n^{2/3} \rfloor$-clique to the $\lfloor n^{1/3} \rfloor$ colors of a $\lfloor n^{1/3} \rfloor$-coloring. However, since $\lfloor n^{1/3} \rfloor^2 \leq \lfloor n^{2/3} \rfloor \leq \lfloor n^{1/3} \rfloor^3$, using a short reasoning in $\mathbf{LK}_2$, one can show that this violates an instance of the pigeonhole principle $\mathrm{PHP}_m^{l(m)}$, when $m = \lfloor n^{1/3} \rfloor$ and $l(m) = \lfloor n^{2/3} \rfloor$. As this pigeonhole principle has an $\mathbf{LK}_2$-proof of size $O(2^{(\log m)^k})$, the instance has an $\mathbf{LK}_2$-proof of size bounded by $2^{(\log n)^{O(1)}}$. This then implies that the tautology $\mathrm{Clique}_n^{1/6}(\bar{p}, \bar{q}) \to \neg\mathrm{Color}_n^{1/6}(\bar{p}, \bar{r})$ has an $\mathbf{LK}_2$-proof of size $2^{(\log n)^{O(1)}}$. Since $P \geq \mathbf{LK}_2$, the same upper bound holds for P-proofs. But if P has monotone feasible interpolation, then by Theorem 64, any P-proof of this tautology must be of size at least $2^{\Omega(n^{1/6})}$, leading to a contradiction. Therefore, P does not have monotone feasible interpolation. ◀

▶ **Theorem 80.** • *[25] Assume that* RSA *is secure against* $\mathbf{P/poly}$ *adversaries. Then, no proof system for* CPC *that simulates* CPC-**SF** *has the feasible interpolation property. In particular,* CPC-**SF** *is not automatable.*

- *[5, 6] Assume that the Diffie–Hellman protocol is secure against* $\mathbf{P/poly}$ *adversaries. Then, no proof system for* CPC *that simulates* **LK** *has the feasible interpolation property. In particular,* **LK** *is not automatable. Moreover, if we assume that the Diffie–Hellman protocol is secure against circuits of size* 2^{n^ϵ} *for any* $\epsilon > 0$*, then there exists* $d \geq 2$ *such that* $\mathbf{LK}_d$ *does not have the feasible interpolation property, and hence is not automatable.*

Proof. We only prove the first part, as the second follows by a similar argument. Let P be a proof system for CPC that simulates CPC-**SF**. By Theorem 78, the sequence $\{\mathrm{RSA}^1_n(\bar{p},\bar{q}) \to \neg\mathrm{RSA}^0_n(\bar{p},\bar{r})\}_n$ has a short CPC-**SF**-proof, and hence a short P-proof. Assuming that RSA is secure against $\mathbf{P/poly}$ adversaries, it follows from Example 45 that this sequence has hard Craig interpolants. Therefore, by Theorem 70, the system P cannot have the feasible interpolation property. To conclude non-automatability, recall that CPC-**SF** is substitutable and closed under variable-free modus ponens. ◀

7 Feasible Interpolation (Non-classical)

In this section, we adapt the feasible interpolation technique to the non-classical setting by replacing Craig interpolants with disjunctive interpolants. Using this modified framework, we demonstrate that many proof systems, including **LJ** (or equivalently, IPC-Frege), as well as L-Frege for logics $L \subseteq$ S4 or $L \subseteq$ GL, are not p-bounded.

▶ **Definition 81.** *Let* L *be a si logic. A proof system* P *for* L *is called to have* (monotone) feasible PDIP *if there is a polynomial* p *such that for any* P*-proof* $u \in \{0,1\}^*$ *of an implication* $\phi \to \psi \vee \theta$ *(where* ϕ *is monotone) with the code* w*, there are (monotone) circuits* C *and* D *of size bounded by* $p(|u|,|w|)$ *such that* $([C],[D])$ *is an* L-PDI *for* $\phi \to \psi \vee \theta$*. For a modal logic* L*, the definition is similar using formulas in the form* $\phi \to \Box\psi \vee \Box\theta$*,* $\mathcal{L}_\Box$*-circuits and* L-MDI*, instead.*

▶ **Theorem 82.** *Let* L *be a si (resp. consistent modal) logic. If hard disjoint* **NP** *pairs exist, or in particular, if one-way protocols exist, then no proof system for* L *with feasible* PDIP *(resp.* MDIP*) is p-bounded.*

Proof. We prove only the si case; the modal case is analogous. If hard disjoint **NP** pairs exist, or in particular, if one-way protocols exist, Corollary 44 implies the existence of a sequence $\{\phi_n(\bar{p},\bar{r}) \to \psi_n(\bar{p},\bar{s})\}_n$ of short classical tautologies with hard Craig interpolants. By Theorems 54 and 56, $\{\Phi_n\}_n = \{\bigwedge_i (p_i \vee \neg p_i) \to \neg\phi_n(\bar{p},\bar{r}) \vee \neg\neg\psi_n(\bar{p},\bar{s})\}_n$ is a sequence of short formulas in IPC $\subseteq L$ that has hard L-PDI's. Now, if P is p-bounded, then the sequence $\{\Phi_n\}_n$ has a short P-proof. Using the feasible PDIP, this would yield a sequence $\{(C_n,D_n)\}_n$ of pairs of poly-size circuits computing an L-PDI of $\{\Phi_n\}_n$, which is impossible. ◀

▶ Remark 83. One may ask whether it is possible to use the original Craig interpolants to gain insights into non-classical proof systems, similar to what was done for the classical case, or whether it is truly necessary to move to disjunctive interpolants. In the non-classical

setting, we are primarily interested in strong proof systems (e.g., Frege systems), which, as in the classical case, lack feasible interpolation in the sense used in the previous section. To see why, for example, in the case of IPC, one can use the feasible version of Glivenko's theorem together with Theorem 78 to show that the sequence $\{\neg\neg \mathrm{DH}^1_n(\bar{p},\bar{q}) \to \neg \mathrm{DH}^0_n(\bar{p},\bar{r})\}_n$ has a short **LJ**-proof. However, any intuitionistic Craig interpolant for this implication is also a classical Craig interpolant for $\mathrm{DH}^1_n(\bar{p},\bar{q}) \to \neg \mathrm{DH}^0_n(\bar{p},\bar{r})$, which is conjectured to be hard, if the Diffie–Hellman protocol is secure against $\mathbf{P/poly}$. Therefore, assuming this security, the system **LJ** does not admit feasible interpolation in the sense considered in the previous section, and hence we are pushed to use other types of interpolation, such as the disjunctive ones. ⌟

▶ **Theorem 84.** *Let $0 < \epsilon < 1/3$ be a real number, L be a si (resp. consistent modal) logic, and P be a proof system for L. If P has monotone feasible* PDIP *(resp.* MDIP*), then any P-proof of $\bigwedge_i (p_i \vee q_i) \to \neg \mathrm{Clique}^\epsilon_n(\neg\bar{p},\bar{r}) \vee \neg \mathrm{Color}^\epsilon_n(\bar{q},\bar{s})$ (resp. $\bigwedge_i (\Box p_i \vee \Box q_i) \to \Box\neg \mathrm{Clique}^\epsilon_n(\neg\bar{p},\bar{r}) \vee \Box\neg \mathrm{Color}^\epsilon_n(\bar{q},\bar{s})$) has size at least $2^{\Omega(n^{1/3-\epsilon})}$. Hence, P is not p-bounded.*

Proof. The proof is a direct consequence of Theorem 58. ◀

In the next two subsections, we prove monotone feasible PDIP (resp. MDIP) for certain proof systems for IPC (resp. consistent modal logics) in order to apply Theorem 84.

7.1 Super-intuitionistic Logics

Our main objective in this subsection is to follow the strategy presented in [13] to show that the proof system **LJ** has monotone feasible PDIP. We begin with a definition and some preliminary observations.

▶ **Definition 85.** *An $\mathcal{L}_p$-formula is called an* implicational Horn formula *if it is either an atom or has the form $\bigwedge_{i=1}^k p_i \to r$, where the p_i's and r are atomic formulas.*

▶ **Lemma 86.** *Let Γ be a sequence of implicational Horn formulas, Π and Δ be sequences of monotone $\mathcal{L}_p$-formulas and q and r be atoms. Then:*
(i) If the sequent $(\Gamma, \Pi \Rightarrow \Delta)$ is valid, then $\Gamma, \Pi \Rightarrow \bigvee \Delta$ is provable in **LJ**.
(ii) If the sequent $(\Gamma \Rightarrow q \vee r)$ is valid, then either $(\Gamma \Rightarrow q)$ or $(\Gamma \Rightarrow r)$ is valid.

Proof. For part (i), we proceed by induction on a cut-free **LK**-proof of $\Gamma, \Pi \Rightarrow \Delta$. The key observation is that since Δ contains no implications and the formulas in $\Gamma \cup \Pi$ do not involve implications of depth greater than one, the classical rule $(R\to)$ is never applied in the proof. Hence, the classical derivation is already valid intuitionistically. For part (ii), apply part (i) to obtain $\mathbf{LJ} \vdash \Gamma \Rightarrow q \vee r$. Then, an induction on the cut-free **LJ**-proof of this sequent shows that either $\mathbf{LJ} \vdash \Gamma \Rightarrow q$ or $\mathbf{LJ} \vdash \Gamma \Rightarrow r$. ◀

The second property of implicational Horn formulas is the following crucial theorem, proved by Hrubeš [13]:

▶ **Theorem 87** (Hrubeš [13]). *For any sequence Γ of implicational Horn formulas, a finite set of atoms $P = \{p_1, \ldots, p_m\}$, and an atom q, there exists a monotone circuit $C(x_1, \ldots, x_m)$ of size $(|\Gamma| + m)^{O(1)}$ such that $C(\bar{a}) = 1$ if and only if the sequent $(\Gamma, \{p_i \in P \mid a_i = 1\} \Rightarrow q)$ is valid, for any $\bar{a} \in \{0,1\}^m$.*

Our strategy for proving monotone feasible PDIP for **LJ** is to first establish a similar, yet technically simpler, claim in which the disjunction on the right-hand side ranges over atoms, and the left-hand side can have an additional sequence of implicational Horn formulas. We then define a suitable translation function to reduce the general case to this specific setting. This translation is a powerful machine for extracting feasible data from intuitionistic proofs and is of independent interest. For more, see [1].

▶ **Theorem 88.** *For any sequence Γ of implicational Horn formulas, a monotone $\mathcal{L}_p$-formula ϕ and atoms q and r, if $\Gamma, \phi \Rightarrow q \vee r$ is valid, then there are monotone circuits C and D in the variables of ϕ and of size $(|\Gamma| + |\phi|)^{O(1)}$ such that the sequents $(\Gamma, \phi \Rightarrow [C] \vee [D])$, $(\Gamma, [C] \Rightarrow q)$ and $(\Gamma, [D] \Rightarrow r)$ are all valid.*

Proof. Let $P = V(\phi)$ and $C(\bar{x})$ be the monotone circuit from Theorem 87 for P, Γ, and q and define $D(\bar{x})$ similarly but using r instead of q. We claim that $C(\bar{p})$ and $D(\bar{p})$ are the circuits we want. First, it is clear that the size of $C(\bar{p})$ and $D(\bar{p})$ is bounded by $(|\Gamma| + |\phi|)^{O(1)}$. Second, define T as the set of the assignments $\bar{a} \in \{0,1\}$ for the atoms in P that makes ϕ true. For any $\bar{a} \in T$, as ϕ is monotone, the sequent $(\{p_i \in P \mid a_i = 1\} \Rightarrow \phi)$ is valid. Hence, as $\Gamma, \phi \Rightarrow q \vee r$ is valid, so is $(\Gamma, \{p_i \in P \mid a_i = 1\} \Rightarrow q \vee r)$. As $\Gamma \cup \{p_i \mid a_i = 1\}$ consists of implicational Horn formulas, by Theorem 86, part (ii), either $\Gamma, \{p_i \in P \mid a_i = 1\} \Rightarrow q$ or $\Gamma, \{p_i \in P \mid a_i = 1\} \Rightarrow r$ are valid. Therefore, for any $\bar{a} \in T$, either $C(\bar{a}) = 1$ or $D(\bar{a}) = 1$. This means that $\phi \Rightarrow [C(\bar{p})] \vee [D(\bar{p})]$ is valid. Third, we claim that $\Gamma, [C(\bar{p})] \Rightarrow q$ is valid. The reason is that for any assignment for the atoms in the sequent that assigns $\bar{a}$ for $\bar{p}$, if $C(\bar{a}) = 1$, then by definition of $C(\bar{x})$, the sequent $\Gamma, \{p_i \mid a_i = 1\} \Rightarrow q$ is valid. Now, using the same assignment, as every atoms in the set $\{p_i \mid a_i = 1\}$ becomes true by mapping p_i to a_i, the sequent $\Gamma \Rightarrow q$ becomes true under the assignment. Hence, $\Gamma, [C(\bar{p})] \Rightarrow q$ is valid. Similarly, we can prove that $\Gamma, [D(\bar{p})] \Rightarrow r$ is valid. ◀

For the translation part, define $\mathcal{L}_p^+$ as the extension of $\mathcal{L}_p$ by new atoms of the form $\langle\phi\rangle$, for each formula $\phi \in \mathcal{L}_p$. Clearly, there is a canonical substitution that substitutes each atom $\langle\phi\rangle$ with the corresponding formula ϕ. We refer to this substitution as the *standard substitution*, and denote it by s. Now, consider the following translation:

▶ **Definition 89.** *Define the translation $t : \mathcal{L}_p \to \mathcal{L}_p^+$ as follows: $\bot^t = \bot$, $\top^t = \langle\top\rangle$, $p^t = \langle p\rangle$ for any atomic formula p, and $(\phi \circ \psi)^t = (\phi^t \circ \psi^t) \wedge \langle\phi \circ \psi\rangle$, for any $\circ \in \{\wedge, \vee, \to\}$. For a multiset Γ, define $\Gamma^t = \{\gamma^t \mid \gamma \in \Gamma\}$.*

It is clear that the function t and the substitution s are polynomial-time computable. Moreover, for any formula $\phi \in \mathcal{L}_p$, we have $\mathbf{LJ} \vdash \phi^t \Rightarrow \langle\phi\rangle$ and $\mathbf{LJ} \vdash (\phi^t)^s \Leftrightarrow \phi$.

▶ **Lemma 90.** *There is a polynomial p such that for any* **LJ***-proof π of $\Omega \Rightarrow \Lambda$, there is a sequence Σ_π of implicational Horn formulas such that $|\Sigma_\pi| \leq p(|\pi|)$, $\mathbf{LJ} \vdash \Sigma_\pi, \Omega^t \Rightarrow \Lambda^t$, and $\mathbf{LJ} \vdash \Rightarrow \bigwedge \Sigma_\pi^s$.*

Proof. The construction of Σ_π in terms of π is by recursion on the structure of π. The case of axioms is easy. If the last rule in π is structural (including the cut) or a left rule, it suffices to set Σ_π as the union of the Σ's from the immediate subproofs of π. For the right rules, we explain only the case of $(R\to)$; the others are similar. Suppose π proves $\Gamma \Rightarrow \phi \to \psi$, applying the rule $(R\to)$ on the **LJ**-proof π' with the conclusion $\Gamma, \phi \Rightarrow \psi$. Then, by the induction hypothesis, we have the multiset $\Sigma_{\pi'}$ of implicational Horn formulas such that $\mathbf{LJ} \vdash \Sigma_{\pi'}, \Gamma^t, \phi^t \Rightarrow \psi^t$ and $\mathbf{LJ} \vdash \Rightarrow \bigwedge \Sigma^s_{\pi'}$. Hence, $\mathbf{LJ} \vdash \Sigma_{\pi'}, \Gamma^t \Rightarrow \phi^t \to \psi^t$. Then, set $\Sigma_\pi = \Sigma_{\pi'} \cup \{\bigwedge_{\gamma\in\Gamma}\langle\gamma\rangle \to \langle\phi \to \psi\rangle\}$. It is clear that Σ_π consists of implicational Horn formulas and that $\mathbf{LJ} \vdash \Rightarrow \bigwedge \Sigma^s_\pi$. Since $\mathbf{LJ} \vdash \gamma^t \Rightarrow \langle\gamma\rangle$ for any $\gamma \in \Gamma$, we obtain $\mathbf{LJ} \vdash \Sigma_\pi, \Gamma^t \Rightarrow \langle\phi \to \psi\rangle$. Therefore, as $(\phi \to \psi)^t = (\phi^t \to \psi^t) \wedge \langle\phi \to \psi\rangle$, we conclude that $\mathbf{LJ} \vdash \Sigma_\pi, \Gamma^t \Rightarrow (\phi \to \psi)^t$. This completes the construction of Σ_π. By inspecting the construction, it is easy to see that $|\Sigma_\pi| \leq |\pi|^{O(1)}$. ◀

▶ **Theorem 91.** **LJ** *has monotone feasible* PDIP*.*

Proof. Let π be an **LJ**-proof of $\Rightarrow \phi \to \psi \vee \theta$, where ϕ is monotone. Then, using cut, it is easy to construct an **LJ**-proof π' of $\phi \Rightarrow \psi \vee \theta$, such that $|\pi'| \leq |\pi|^{O(1)}$. Then, by Lemma 90, there is a sequence of implicational Horn formulas $\Sigma_{\pi'}$ such that $|\Sigma_{\pi'}| \leq |\pi'|^{O(1)} \leq |\pi|^{O(1)}$, $\mathbf{LJ} \vdash \Sigma_{\pi'}, \phi^t \Rightarrow (\psi \vee \theta)^t$, and $\mathbf{LJ} \vdash \Rightarrow \bigwedge \Sigma^s_{\pi'}$. By the definition of t, we have $\mathbf{LJ} \vdash \Sigma_{\pi'}, \phi^t \Rightarrow \psi^t \vee \theta^t$. As $\mathbf{LJ} \vdash \psi^t \Rightarrow \langle\psi\rangle$ and $\mathbf{LJ} \vdash \theta^t \Rightarrow \langle\theta\rangle$, we get $\mathbf{LJ} \vdash \Sigma_{\pi'}, \phi^t \Rightarrow \langle\psi\rangle \vee \langle\theta\rangle$, which implies the validity of $(\Sigma_{\pi'}, \phi^t \Rightarrow \langle\psi\rangle \vee \langle\theta\rangle)$. Let $q = \langle\psi\rangle$ and $r = \langle\theta\rangle$. Then, as ϕ and hence ϕ^t are monotone, by Theorem 88, there exist monotone circuits C' and D' on the variables of ϕ^t and of size bounded by $(|\Sigma_{\pi'}| + |\phi^t|)^{O(1)} \leq |\pi|^{O(1)}$, such that the sequents $(\Sigma_{\pi'}, \phi^t \Rightarrow [C'] \vee [D'])$, $(\Sigma_{\pi'}, [C'] \Rightarrow q)$, and $(\Sigma_{\pi'}, [D'] \Rightarrow r)$ are all valid and hence provable in **LJ**, by Theorem 86, part (i). Now note that the atoms of ϕ^t are of the form $\langle\alpha\rangle$, where α is a subformula of ϕ. As these α's are monotone, applying the standard substitution s to C' and D' yields two monotone circuits C and D on the variables of ϕ and of size bounded by $|\pi|^{O(1)}$. Since $\mathbf{LJ} \vdash \Rightarrow \bigwedge \Sigma^s_{\pi'}$ and $\mathbf{LJ} \vdash (\phi^t)^s \Leftrightarrow \phi$, by applying the standard substitution s, the sequents $(\phi \Rightarrow [C] \vee [D])$, $([C] \Rightarrow \psi)$, and $([D] \Rightarrow \theta)$ are all provable in **LJ**. Hence, $(\phi \to [C] \vee [D])$, $([C] \to \psi)$, and $([D] \to \theta)$ are all provable in IPC. ◀

▶ **Corollary 92** (Hrubeš [11–13])**.** *Let $0 < \epsilon < 1/3$ be a real number. Then, any* **LJ***-proof of $\bigwedge_i (p_i \vee q_i) \to \neg\mathrm{Clique}^\epsilon_n(\neg\bar{p}, \bar{r}) \vee \neg\mathrm{Color}^\epsilon_n(\bar{q}, \bar{s})$ has size at least $2^{\Omega(n^{1/3-\epsilon})}$. Therefore,* **LJ** *and hence* IPC*-Frege is not p-bounded.*

Proof. For the first part, we apply Theorem 84 and Theorem 91. For the second part, we recall that **LJ** is equivalent to the IPC-Frege system, by Theorem 37. ◀

For si logics of infinite branching, Jeřábek extended Hrubeš's result by combining reductions with a modification of the above argument:

▶ **Theorem 93** (Jeřábek [16]). *Let L be a si logic of infinite branching, i.e., either $\mathsf{L} \subseteq \mathsf{BD}_2$ or $\mathsf{L} \subseteq \mathsf{KC} + \mathsf{BD}_3$. Then, L-Frege is not p-bounded.*

▶ Remark 94. Here are two remarks. First, as we have seen, the formulas with exponentially long proofs we provided make serious use of disjunctions, and one may suspect that the use of disjunctions is crucial for such bounds. However, Jeřábek showed that a similar result is possible using only the implicational fragment of the language [17, 19]. Second, for a logic L weaker than IPC, it may seem obvious that L-Frege cannot be p-bounded, since L is weaker than IPC and even IPC-Frege proofs are already hard for the intuitionistic Clique-Coloring tautologies. However, this is not so straightforward. To show that L-Frege is not p-bounded, we must provide a sequence of short formulas *in* L that have no short L-Frege proofs, and there is no reason to assume that our intuitionistic Clique-Coloring tautologies belong to L. To address this issue, Jalali [15] employed a suitable descending technique to modify the intuitionistic Clique-Coloring tautologies so that they become provable in the Full Lambek substructural logic FL. She then showed that for any logic between FL and a superintuitionistic logic of infinite branching, the suitably defined L-Frege system is not p-bounded. ⌟

7.2 Modal Logics

Our task in this subsection is to apply a technique similar to that used in the previous subsection to show that **GL** and **S4** have monotone feasible MDIP. First, extend the language $\mathcal{L}_\Box$ to $\mathcal{L}_\Box^+$ by introducing new atoms $\langle\Box\phi\rangle$ for each formula $\phi \in \mathcal{L}_\Box$. Clearly, there is a canonical substitution s that substitutes each atom $\langle\Box\phi\rangle$ with $\Box\phi$. We refer to this substitution as the *standard substitution*. Now, consider the following translations:

▶ **Definition 95.** *For any $i \in \{0,1\}$, define $t_i : \mathcal{L}_\Box \to \mathcal{L}_\Box^+$ by: $\bot^{t_i} = \bot$, $\top^{t_i} = \top$, and $p^{t_i} = p$, for any atomic formula p; $(\phi \circ \psi)^{t_i} = (\phi^{t_i} \circ \psi^{t_i})$, for any $\circ \in \{\wedge, \vee, \to\}$; and $(\Box\phi)^{t_0} = \langle\Box\phi\rangle$, while $(\Box\phi)^{t_1} = \phi^{t_1} \wedge \langle\Box\phi\rangle$. For a multiset Γ and any $i \in \{0,1\}$, define $\Gamma^{t_i} = \{\gamma^{t_i} \mid \gamma \in \Gamma\}$.*

Note that both ϕ^{t_0} and ϕ^{t_1} are propositional formulas. The functions t_i, for any $i \in \{0,1\}$, and the substitution s are polynomial-time computable, and it is clear that $\mathbf{K} \vdash (\Box\phi)^{t_i} \Rightarrow \langle\Box\phi\rangle$, for any $i \in \{0,1\}$, $\mathbf{GL} \vdash (\phi^{t_0})^s \Leftrightarrow \phi$ and $\mathbf{S4} \vdash (\phi^{t_1})^s \Leftrightarrow \phi$, for any formula $\phi \in \mathcal{L}_\Box$.

▶ **Lemma 96.** *Let $L_0 = \mathsf{GL}$, $L_1 = \mathsf{S4}$, $G_0 = \mathbf{GL}$, and $G_1 = \mathbf{S4}$. Then, for any $i \in \{0,1\}$, there is a polynomial p such that for any G_i-proof π of $\Omega \Rightarrow \Lambda$, there is a sequence Σ_π of implicational Horn formulas such that $|\Sigma_\pi| \le |\pi|^{O(1)}$, the sequent $(\Sigma_\pi, \Omega^{t_i} \Rightarrow \Lambda^{t_i})$ is valid, and $G_i \vdash \Rightarrow \bigwedge \Sigma_\pi^s$.*

Proof. For any $i \in \{0,1\}$, the construction of Σ_π proceeds by recursion on the structure of π. The case for axioms is straightforward, by setting $\Sigma_\pi = \emptyset$. For the rules, if the last rule in π is a structural or propositional rule, it suffices to set Σ_π as the union of the Σ's of the immediate subproofs. The only interesting cases occur when the last rule in π is a modal rule. For $i = 0$, let π proves $\Box\Gamma \Rightarrow \Box\phi$, applying the rule (GL) on the proof π' of $\Gamma, \Box\Gamma, \Box\phi \Rightarrow \phi$.

Then, define $\Sigma_\pi = \{\bigwedge_{\gamma\in\Gamma}\langle\Box\gamma\rangle \to \langle\Box\phi\rangle\}$. It is clear that Σ_π consists of implicational Horn formulas, and the sequent $(\Sigma_\pi, (\Box\Gamma)^{t_0} \Rightarrow (\Box\phi)^{t_0})$ is valid. Moreover, we have $\mathbf{GL} \vdash \Rightarrow \bigwedge \Sigma^s_\pi$. For $i = 1$, if the last rule in π is $(LS4)$, set Σ_π as the Σ of the immediate subproof. For $(RS4)$, let π proves $\Box\Gamma \Rightarrow \Box\phi$, applying the rule $(RS4)$ on the proof π' of $\Box\Gamma, \Rightarrow \phi$. Then, by the induction hypothesis, there is a sequence $\Sigma_{\pi'}$ of implicational Horn formulas such that $(\Sigma_{\pi'}, (\Box\Gamma)^{t_1} \Rightarrow \phi^{t_1})$ is valid and $\mathbf{S4} \vdash \Rightarrow \bigwedge \Sigma^s_{\pi'}$. Define $\Sigma_\pi = \Sigma_{\pi'} \cup \{\bigwedge_{\gamma\in\Gamma}\langle\Box\gamma\rangle \to \langle\Box\phi\rangle\}$. It is clear that Σ_π consists of implicational Horn formulas and that $\mathbf{S4} \vdash \Rightarrow \bigwedge \Sigma^s_\pi$. Since $(\Box\phi)^{t_1} = (\phi^{t_1} \wedge \langle\Box\phi\rangle)$ and $(\Box\gamma)^{t_1} = (\gamma^{t_1} \wedge \langle\Box\gamma\rangle)$ for any $\gamma \in \Gamma$, it is easy to verify the validity of $(\Sigma_\pi, (\Box\Gamma)^{t_1} \Rightarrow (\Box\phi)^{t_1})$. Finally, checking the construction, we have $|\Sigma_\pi| \le |\pi|^{O(1)}$. ◀

▶ **Theorem 97.** **S4** *and* **GL** *have monotone feasible* MDIP*.*

Proof. For **S4**, let π be an **S4**-proof of $(\Rightarrow \phi \to \Box\psi \vee \Box\theta)$, where ϕ is a monotone $\mathcal{L}_\Box$-formula. Using the cut rule, it is easy to construct an **S4**-proof π' of $\phi \Rightarrow \Box\psi \vee \Box\theta$ such that $|\pi'| \le |\pi|^{O(1)}$. By Lemma 96, we obtain a sequence of implicational Horn formulas $\Sigma_{\pi'}$ such that $|\Sigma_{\pi'}| \le |\pi'|^{O(1)} \le |\pi|^{O(1)}$, the sequent $(\Sigma_{\pi'}, \phi^{t_1} \Rightarrow (\Box\psi)^{t_1} \vee (\Box\theta)^{t_1})$ and hence $(\Sigma_{\pi'}, \phi^{t_1} \Rightarrow \langle\Box\psi\rangle \vee \langle\Box\theta\rangle)$ is valid, and $\mathbf{S4} \vdash \Rightarrow \bigwedge \Sigma^s_{\pi'}$. Since ϕ^{t_1} is monotone, by Lemma 88, there exist monotone circuits C' and D' on the variables of ϕ^{t_1} and of size bounded by $(|\Sigma_{\pi'}| + |\phi^{t_1}|)^{O(1)} \le |\pi|^{O(1)}$, such that the sequents $(\Sigma_{\pi'}, \phi^{t_1} \Rightarrow [C'] \vee [D'])$, $(\Sigma_{\pi'}, [C'] \Rightarrow \langle\Box\psi\rangle)$, and $(\Sigma_{\pi'}, [D'] \Rightarrow \langle\Box\theta\rangle)$ are all valid. Now, applying the standard substitution, we obtain monotone $\mathcal{L}_\Box$-circuits C and D on the variables of ϕ and of size bounded by $|\pi|^{O(1)}$, such that the sequents $((\phi^{t_1})^s \Rightarrow [C] \vee [D])$, $([C] \Rightarrow \Box\psi)$, and $([D] \Rightarrow \Box\theta)$ are all provable in **S4**. Since $\mathbf{S4} \vdash \phi \Leftrightarrow (\phi^{t_1})^s$, it follows that the formulas $(\phi \to [C] \vee [D])$, $([C] \to \Box\psi)$, and $([D] \to \Box\theta)$ are all provable in S4. For **GL**, the proof is similar; it suffices to use the translation t_0 instead of t_1. ◀

▶ **Corollary 98** (Hrubeš [11–13])**.** *Let* $0 < \epsilon < 1/3$ *be a real number. Then, any* **GL***-proof or* **S4***-proof of* $\bigwedge_i(\Box p_i \vee \Box q_i) \to \Box\neg\mathrm{Clique}^\epsilon_n(\neg\bar{p}, \bar{r}) \vee \Box\neg\mathrm{Color}^\epsilon_n(\bar{q}, \bar{s})$ *has size at least* $2^{\Omega(n^{1/3-\epsilon})}$*. Therefore, the proof systems* **GL** *and* **S4** *and hence, for any modal logic L below* S4 *or* GL*, the system L-Frege is not p-bounded.*

Proof. For the first part, we apply Theorem 84 and Theorem 97. For the second part, use Theorem 37. ◀

For modal logics of infinite branching, we again have the following extension:

▶ **Theorem 99** (Jeřábek [16])**.** *For any modal logic L of infinite branching, L-Frege is not p-bounded.*

Acknowledgments

We wish to thank Pavel Pudlák, Jan Krajíček, and Erfan Khaniki, as well as our reviewers, for their very helpful suggestions and comments. We gratefully acknowledge the support of the Austrian Science Fund (FWF) project P33548 and the Dutch Research Council (NWO) project OCENW.M.22.258.

References

1 Amirhossein Akbar Tabatabai and Raheleh Jalali. Universal proof theory: Feasible admissibility in intuitionistic modal logics. *Annals of Pure and Applied Logic*, 176(2):103526, 2025.

2 Noga Alon and Ravi B Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7:1–22, 1987.

3 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

4 Olaf Beyersdorff and Oliver Kutz. Proof complexity of non-classical logics. In *European Summer School in Logic, Language and Information*, pages 1–54. Springer, 2010.

5 Maria Luisa Bonet, Carlos Domingo, Ricard Gavalda, Alexis Maciel, and Toniann Pitassi. Non-automatizability of bounded-depth Frege proofs. *computational complexity*, 13:47–68, 2004.

6 Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. No feasible interpolation for TC^0-Frege proofs. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 254–263. IEEE, 1997.

7 Sam Buss and Grigori Mints. The complexity of the disjunction and existential properties in intuitionistic logic. *Annals of Pure and Applied Logic*, 99(1-3):93–104, 1999.

8 Samuel R Buss and Pavel Pudlák. On the computational content of intuitionistic propositional proofs. *Annals of Pure and Applied Logic*, 109(1-2):49–64, 2001.

9 Stephen A Cook and Robert A Reckhow. The relative efficiency of propositional proof systems. *The journal of symbolic logic*, 44(1):36–50, 1979.

10 Pavel Hrubeš and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 121–131. IEEE, 2017.

11 Pavel Hrubeš. A lower bound for intuitionistic logic. *Annals of Pure and Applied Logic*, 146(1):72–90, 2007.

12 Pavel Hrubeš. Lower bounds for modal logics. *The Journal of Symbolic Logic*, 72(3):941–958, 2007.

13 Pavel Hrubeš. On lengths of proofs in non-classical logics. *Annals of Pure and Applied Logic*, 157(2-3):194–205, 2009.

14 Neil Immerman. *Descriptive complexity*. Springer Science & Business Media, 1998.

15 Raheleh Jalali. Proof complexity of substructural logics. *Annals of Pure and Applied Logic*, 172(7):102972, 2021.

16 Emil Jeřábek. Substitution Frege and extended Frege proof systems in non-classical logics. *Annals of Pure and Applied Logic*, 159(1-2):1–48, 2009.

17 Emil Jeřábek. Proof complexity of intuitionistic implicational formulas. *Annals of Pure and Applied Logic*, 168(1):150–190, 2017.

18 Emil Jeřábek. On the proof complexity of logics of bounded branching. *Annals of Pure and Applied Logic*, 174(1):103181, 2023.

19 Emil Jeřábek. A simplified lower bound for implicational logic. *Bulletin of Symbolic Logic*, 31(1):53–87, 2025.

20 Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer, 2012.

21 Jan Krajíček. *Proof complexity*, volume 170. Cambridge University Press, 2019.

22 Jan Krajíček. Lower bounds to the size of constant-depth propositional proofs. *The Journal of Symbolic Logic*, 59(1):73–86, 1994.

23 Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997.

24 Jan Krajíček and Pavel Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. In *Logic and Computational Complexity: International Workshop LCC'94 Indianapolis, IN, USA, October 13–16, 1994 Selected Papers*, pages 210–220. Springer, 1995.

25 Jan Krajíček and Pavel Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. *Information and Computation*, 140(1):82–94, 1998.

26 Alexis Maciel, Toniann Pitassi, and Alan R Woods. A new proof of the weak pigeonhole principle. *Journal of Computer and System Sciences*, 64:843–872, 2002.

27 David Makinson. Some embedding theorems for modal logic. *Notre Dame Journal of Formal Logic*, 12(2):252–254, 1971.

28 Daniele Mundici. Complexity of Craig's interpolation. *Fundamenta Informaticae*, 5(3-4):261–278, 1982.

29 Daniele Mundici. A lower bound for the complexity of Craig's interpolants in sentential logic. *Archiv für mathematische Logik und Grundlagenforschung*, 23(1):27–36, 1983.

30 Daniele Mundici. Tautologies with a unique Craig interpolant, uniform vs. nonuniform complexity. *Annals of Pure and Applied Logic*, 27(3):265–273, 1984.

31 Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997.

32 Pavel Pudlák and Jiri Sgall. Algebraic models of computation and interpolation for algebraic proof systems. *Proof complexity and feasible arithmetics*, 39:279–296, 1996.

33 Alexander Razborov. Lower bounds on the monotone complexity of some Boolean function. In *Soviet Math. Dokl.*, volume 31, pages 354–357, 1985.

34 Robert A Reckhow. *On the Lengths of Proofs In The Propositional Calculus*. PhD thesis, 1975.

Chapter 7

Interpolation in Model Theory

Jouko Väänänen
University of Helsinki, Finland

Abstract

We bring an abstract model theory perspective to interpolation. We ask, what is the role of interpolation in the study of extensions of first order logic, such as infinitary logics, generalized quantifiers and higher order logics? The abstract model theory approach reveals the basic connections between various interpolation properties in isolation, on their own, as well as with respect to other model theoretic properties, such as compactness.

Contents

1 Introduction 245
2 The Abstract Setting 246
3 Generalized Quantifiers 257
4 Infinitary Logics 260
5 Higher Order Logics 266
6 Conclusion 269
Acknowledgments 269
References 269

1 Introduction

The Craig Interpolation Theorem ([10]) was originally proved by proof theoretic methods. Even today proof theory or methods such as tableaux methods derived from proof theory are the most popular road to interpolation. The model theoretic version of interpolation is the Robinson property ([46]), which can be readily proved with purely model theoretic methods. The statement of the Robinson property resembles amalgamation which is at the heart of modern model theory. Neither interpolation nor the Robinson property seem to be central in current model theory but the idea of amalgamation is certainly central (see e.g.



© Jouko Väänänen;
licensed under Creative Commons License CC BY 4.0
In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 7, pages 245–272. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.g

[27]). Moreover, the model theoretic proof of the Robinson property is in principle based on the use of one form or another of saturated models and saturated models are certainly important in model theory.

In this chapter we take the axiomatic (or abstract) approach to model theory. We investigate which model theoretic properties of first order logic, which we denote in this chapter by $L_{\omega\omega}$, are behind interpolation and the Robinson property. For this to make sense we recall the concept of an abstract logic as well as a variety of extensions of first order logic, some with interpolation and some without. This indicates what it is in first order (or some other) logic that allows us to derive the interpolation theorem.

The topic of interpolation and its weaker forms in model theoretic languages is a vast topic and we will only touch upon some basic results in this chapter. The topic is so extensive because there are numerous extensions of first order logic as well as numerous weaker forms of interpolation. This leads to a huge potential for implications and counter-examples. We have chosen to include in this chapter only what we think as illustrative of the situation. A good reference to a more complete picture is still today [2].

Section 2 introduces the basic concepts of abstract model theory and the basic general results about compactness, interpolation, the Robinson property and the Beth Definability Theorem ([7]). We prove some basic relationships between these properties. The focus is on how much we have to assume of the logics for the relationships to hold. Section 3 concentrates on generalized quantifiers, wherein we present the basic facts. The amount of material here is huge but we focus on what we think is illuminating. Section 4 presents the basic facts about interpolation in infinitary logics ending with Shelah's relatively new logic L^1_κ, which has an interpolation property ([51]). Finally in Section 5 we discuss the matter of interpolation in higher order logics, including logics without negation such as existential second order logic and dependence logic ([53]).

We consider both single-sorted and many-sorted logic, as the difference between the two is relevant in the study of interpolation. Everything is many-sorted unless otherwise specified. As is well-known, many-sorted logic can be reduced to single-sorted logic, but this reduction is not so perfect that many-sorted vocabularies would be rendered useless.

The reader is assumed to know basic first order model theory. Otherwise we try to be self-contained. For details on generalized quantifiers, infinitary languages and abstract model theory, a good source is [2]. For unexplained concepts of set theory we refer to [29]. For basics of model theory we refer to [27].

2 The Abstract Setting

By abstract model theory we mean here the study of extensions of first order logic, such as logics with generalized quantifiers, infinitary logics and higher order logics. Interpolation by and large fails in such logics but there are notable exceptions, such as $L_{\omega_1\omega}$ (see Section 4), and in some cases interpolation can be established in a relative sense meaning that the

interpolant is found, not in the logic under consideration, but in a hopefully not too much bigger logic.

In its most general form, an *abstract logic*, semantically conceived, is just a triple $L = (S, F, \models)$ where S and F are classes and $\models$ is a subclass of $S \times F$. Elements of S are called the *structures* of L, elements of F are called the *sentences* of L, and the relation $\models$ is called the satisfaction relation of L. Note that we do not assume anything about the "structures" that constitute S. They need not be structures in the sense of ordinary first order model theory. They can also be Kripke models or valuations of propositional logic or whatnot. In this generality the concept of an abstract logic covers first order logic and its extensions by generalized quantifiers and infinitary operations, but also modal logic, propositional logic, topological logic [19], etc. Likewise, the elements of F, i.e. the "sentences" of L, need not be sentences in any usual sense. Normally sentences are identified with finite strings of symbols but sentences of infinitary logics, such as $L_{\omega_1\omega}$ are best thought of as sets (or perhaps trees). Finally, the satisfaction relation $\models$ does not have to have any inductive definition tying together S and F. All we assume is that it is a subclass of $S \times F$, which means that it is a class in the sense of set theory[1] i.e. it is a definable predicate of set-theory, perhaps from some parameters.

On this level of generality it is, of course, hardly possible to prove any deep results about abstract logics. But if some simple results could be proved, they would automatically apply to first order logic, propositional logic, modal logic, and so on, and would manifest similarity or a "family resemblance" of the logics from this point of view, while the logics mentioned live otherwise lightyears from each other. Indeed, as we shall see, some very basic facts about interpolation and model theoretic properties around it can be proved even in our very general setup. Whether the results that can be proved in this generality are interesting or not, is a reasonable question. Be that as it may, it is however notable that we can perfectly formulate many fundamental model theoretic *concepts* on this level of generality, even if no deep results can be proved.

The basic concepts of abstract logics are the following: A structure $\mathfrak{M} \in S$ is a *model of* $\varphi \in F$ if $\mathfrak{M} \models \varphi$, and a *model of* $\Sigma \subseteq F$, in symbols $\mathfrak{M} \models \Sigma$, if $\mathfrak{M}$ is a model of each $\varphi \in \Sigma$. The sentence $\varphi \in F$, or a set $\Sigma \subseteq F$, *is consistent* if it has a model. Whenever $\varphi, \psi \in F$, we write $\varphi \models \psi$, if for every model $\mathfrak{M} \in S$, $\mathfrak{M} \models \varphi$ implies $\mathfrak{M} \models \psi$, and $\varphi \equiv \psi$ if $\varphi \models \psi$ and $\psi \models \varphi$. Whenever $\Sigma \cup \{\psi\} \subseteq F$, we write $\Sigma \models \psi$, if $\mathfrak{M} \models \Sigma$ implies $\mathfrak{M} \models \psi$ for all $\mathfrak{M} \in S$. Two models $\mathfrak{M}, \mathfrak{M}' \in S$ are *L-equivalent*, in symbols $\mathfrak{M} \equiv_L \mathfrak{M}'$, if they are models of the same sentences of F. A set $\Sigma \subseteq F$ is called *complete* if for some $\mathfrak{M} \in S$ we have $\Sigma = \{\varphi \in F : \mathfrak{M} \models \varphi\}$. Any consistent set $\Sigma \subseteq F$ can be extended to a complete consistent Σ' by taking a model $\mathfrak{M}$ of Σ and letting $\Sigma' = \{\varphi \in F : \mathfrak{M} \models \varphi\}$.

Already this very general approach imposes an equivalence relation $\equiv_L$ on S. Different abstract logics may impose different equivalence relations on the same class S. Intuitively

[1] Nothing essential would change if we worked in class theory instead of set theory.

speaking a stronger logic imposes a finer equivalence relation, if S is fixed. Respectively, the structures in S impose relationships between sentences in F, e.g. the relation $\varphi \models \psi$.

We can identify a sentence with the class of its models. Likewise, we can identify a structure with the class of sentences it is a model of. Such identifications would lead to further abstraction without necessarily being helpful. But this demonstrates a certain duality between structures and sentences, well-known and much studied in the development of mathematical logic. Of course, these basic relationships between structures and sentences depend heavily on what is our (abstract) logic L.

▶ **Definition 1.** *We say that an abstract logic L satisfies the* Compactness Theorem *or is* compact*, if for every set $\Sigma \subseteq F$ the following holds: If every finite $\Gamma \subseteq \Sigma$ is consistent, then Σ itself is consistent. If this holds for all countable Σ, we say that L satisfies the* Countable Compactness Theorem *or is* countably compact*.*

To formulate interpolation we introduce the *vocabulary* (class) function τ which has $S \cup F$ as its domain. The values of τ are arbitrary sets but intuitively $\tau(\varphi)$ is the set of relation, constant, proposition, function, etc -symbols occurring in the sentence φ. Intuitively, $\tau(\mathfrak{M})$ is the set of relation, constant, proposition, function, etc -symbols that are interpreted in $\mathfrak{M}$. We assume always that $\mathfrak{M} \models \varphi$ implies $\tau(\varphi) \subseteq \tau(\mathfrak{M})$. If $\Sigma \subseteq F$, we write $\tau(\Sigma)$ for $\bigcup_{\varphi \in \Sigma} \tau(\varphi)$. We could go ahead and formulate natural axioms that τ should satisfy in order to correspond to our intuition about the vocabulary of a sentence or of a structure. However, we skip that here, as it is not relevant for us now. We just call the quadruple $(S, F, \models, \tau)$ an abstract logic. For such a quadruple we can now define the concept of interpolation:

▶ **Definition 2.** *The abstract logic $L = (S, F, \models, \tau)$ has the* Interpolation property *if for every $\varphi, \psi \in F$, the relation $\varphi \models \psi$ implies the existence of $\theta \in F$ such that*

1. $\varphi \models \theta$
2. $\theta \models \psi$
3. $\tau(\theta) \subseteq \tau(\varphi) \cap \tau(\psi)$.

In the sense of this definition, first order logic, the infinitary logic $L_{\omega_1\omega}$ ([30], see section 4 below), Shelah's L^1_κ ([51], see section 4 below), single-sorted second order logic (see section 5), propositional logic, modal logic and topological logic ([19]) all have the interpolation property. Still, considering how many extensions of first order logic have been introduced, see e.g. [2], it is a little surprising how few of them have the interpolation property. This circumstance has led to a study of weaker forms of interpolation as well as of the possible reasons why many logics fail to satisfy interpolation.

For the history of the birth of the Interpolation property of first order logic we refer to [13].

Assuming the Compactness Theorem, there is an alternative formulation of interpolation:

▶ **Definition 3.** *The abstract logic $L = (S, F, \models, \tau)$ has the* Robinson (Consistency) property *([46]) if for every complete $\Sigma_0 \subseteq F$, every consistent $\Sigma_1 \subseteq F$ extending Σ_0, and every consistent $\Sigma_2 \subseteq F$ extending Σ_0, such that $\tau(\Sigma_1) \cap \tau(\Sigma_2) = \tau(\Sigma_0)$, the set $\Sigma_1 \cup \Sigma_2$ is consistent.*

The model theoretic proof of the Robinson property for first order logic is based on the following simple idea: Take an infinite saturated model $\mathfrak{M}_1$ of Σ_1 and a saturated model $\mathfrak{M}_2$ of Σ_2 such that $\mathfrak{M}_1$ and $\mathfrak{M}_2$ have the same cardinality. Now $\mathfrak{M}_1 \restriction \tau(\Sigma_0)$ and $\mathfrak{M}_2 \restriction \tau(\Sigma_0)$ are saturated elementary equivalent (since Σ_0 is complete) models of the same cardinality. Hence they are isomorphic. The isomorphism can be used to transfer the interpretations of symbols in $\tau(\Sigma_2) \setminus \tau(\Sigma_0)$ from $\mathfrak{M}_2$ to $\mathfrak{M}_1$. This yields an expansion $\mathfrak{N}$ of $\mathfrak{M}_1$ such that $\mathfrak{N} \restriction \tau(\Sigma_1) = \mathfrak{M}_1$ and $\mathfrak{N} \restriction \tau(\Sigma_2) \cong \mathfrak{M}_2$. Thus $\mathfrak{N}$ is a model of $\Sigma_1 \cup \Sigma_2$. It should be noted that this proof is an overkill but worth knowing because it is probably the quickest proof. Indeed, saturated models are only known to exist if we assume some amount of the Generalized Continuum Hypothesis or alternatively the existence of strongly inaccessible cardinals, neither of which is really needed for the result. The use of saturated models can be avoided by more refined methods, for example special models ([9]) or recursively saturated models ([4]), which always exist. Alternatively, we present below in Theorem 17 a gentler proof due to Lindström. We will also prove below in Theorem 5 the Robinson property from compactness and interpolation in a very general setting.

To connect the Interpolation property and the Robinson property we need to assume a little bit about abstract logics. To this end we define:

▶ **Definition 4.** **1.** *The abstract logic $L = (S, F, \models, \tau)$ is* closed under conjunction *if for every $\varphi, \psi \in F$ there is $\theta \in F$, denoted $\varphi \wedge \psi$, such that $\tau(\theta) = \tau(\varphi) \cup \tau(\psi)$ and for all $\mathfrak{M} \in S$:*

$$\mathfrak{M} \models \varphi \wedge \psi \iff \mathfrak{M} \models \varphi \text{ and } \mathfrak{M} \models \psi.$$

2. *L is* closed under negation *if for every $\varphi \in F$ there is $\theta \in F$, denoted $\neg\varphi$, such that $\tau(\theta) = \tau(\varphi)$ and for all $\mathfrak{M} \in S$:*

$$\mathfrak{M} \models \neg\varphi \iff \mathfrak{M} \not\models \varphi.$$

3. *If L is closed under both conjunction and negation, we denote $\neg(\neg\varphi \wedge \neg\psi)$ by $\varphi \vee \psi$ and $(\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$ by $\varphi \leftrightarrow \psi$.*

▶ **Theorem 5.** *If L is closed under negation and conjunction, and satisfies the Compactness Theorem, then the following conditions are equivalent:*

1. *L has the Interpolation property.*

2. *L has the Robinson property.*

Rather than proving this, we formulate and prove a stronger version. Since in abstract model theory there are numerous examples of the failure of the Interpolation property as well as of the Robinson property, it makes sense to introduce relative versions of both. To this end we need the concept of a *sublogic*:

▶ **Definition 6.** 1. *A logic $L = (S, F, \models, \tau)$ is a sublogic of $L' = (S', F', \models', \tau')$, in symbols $L \leq L'$, if $S = S'$ and for every $\varphi \in F$ there is $\varphi' \in F'$ such that $\varphi \equiv \varphi'$ i.e. for all $\mathfrak{M} \in S$, $\mathfrak{M} \models \varphi$ if and only if $\mathfrak{M} \models' \psi$, and in addition, $\tau(\varphi) = \tau'(\varphi')$. We usually identify φ and φ' and thereby $F \subseteq F'$.*

2. *Logics L and L' are* equivalent, *$L \equiv L'$, if $L \leq L'$ and $L' \leq L$.*

Note that here both logics in the above definition have the same class S of structures. The more general case of different classes of structures can be dealt with by means of the so-called Chu-transform, see [15]. Note that a sublogic of a (countably) compact logic is (countably) compact.

▶ **Definition 7.** *Suppose $L = (S, F, \models, \tau)$ and $L' = (S', F', \models', \tau')$ are abstract logics such that $L \leq L'$.*

1. $\mathrm{Craig}(L, L')$ *holds if for every $\varphi, \psi \in F$, the relation $\varphi \models \psi$ implies the existence of $\theta \in F'$ such that for all $\mathfrak{M} \in S$, $\mathfrak{M} \models \varphi \Rightarrow \mathfrak{M} \models' \theta$, $\mathfrak{M} \models' \theta \Rightarrow \mathfrak{M} \models \psi$, and $\tau(\theta) \subseteq \tau(\varphi) \cap \tau(\psi)$. By* $\mathrm{Craig}(L)$ *we mean* $\mathrm{Craig}(L, L)$.
2. $\mathrm{Rob}(L, L')$ *holds if for every complete $\Sigma_0 \subseteq F'$, every $\Sigma_1 \subseteq F$, such that $\Sigma_0 \cup \Sigma_1$ is consistent, and every $\Sigma_2 \subseteq F$, such that $\Sigma_0 \cup \Sigma_2$ is consistent, if $\tau(\Sigma_0 \cup \Sigma_1) \cap \tau(\Sigma_0 \cup \Sigma_2) = \tau(\Sigma_0)$, then the set $(\Sigma_0 \cap F) \cup \Sigma_1 \cup \Sigma_2$ is consistent. By* $\mathrm{Rob}(L)$ *we mean* $\mathrm{Rob}(L, L)$.

▶ **Theorem 8.** *If $L \leq L'$ are closed under negation and conjunction, and L' satisfies the Compactness Theorem, then the following are equivalent:*

1. $\mathrm{Craig}(L, L')$.
2. $\mathrm{Rob}(L, L')$.

Proof. We follow the standard proof, as e.g. in [2, Chapter II, 7.1.5.].

(1) implies (2): Suppose a complete $\Sigma_0 \subseteq F'$ is given. Suppose also $\Sigma_1 \subseteq F$, such that $\Sigma_0 \cup \Sigma_1$ is (wlog) complete, and $\Sigma_2 \subseteq F$, such that $\Sigma_0 \cup \Sigma_2$ is (wlog) complete, are given, and furthermore $\tau(\Sigma_0 \cup \Sigma_1) \cap \tau(\Sigma_0 \cup \Sigma_2) = \tau(\Sigma_0)$. Finally, suppose the set $(\Sigma_0 \cap F) \cup \Sigma_1 \cup \Sigma_2$ is inconsistent. By the Compactness Theorem of L' there are some $\{\eta_0, \ldots, \eta_k\} \subseteq \Sigma_0 \cap F$, $\{\varphi_1, \ldots, \varphi_n\} \subseteq \Sigma_1$ and $\{\psi_1, \ldots, \psi_m\} \subseteq \Sigma_2$ such that $\{\eta_0, \ldots, \eta_k\} \cup \{\varphi_1, \ldots, \varphi_n\} \cup \{\psi_1, \ldots, \psi_m\}$ is inconsistent. Since $\Sigma_0 \cup \Sigma_1$ and $\Sigma_0 \cup \Sigma_2$ are individually both consistent, we have $n > 0$ and $m > 0$. Let $\eta = \eta_1 \wedge \ldots \wedge \eta_n$, $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n$ and $\psi = \neg(\psi_1 \wedge \ldots \wedge \psi_m)$. Thus $\eta \wedge \varphi \models \psi$. Let $\theta \in F$ such that $\eta \wedge \varphi \models \theta$, $\theta \models \psi$ and $\tau(\theta) \subseteq \tau(\eta \wedge \varphi) \cap \tau(\psi)$. Since $\Sigma_0 \cup \Sigma_1$ is complete, $\theta \in \Sigma_0 \cup \Sigma_1$. Since $\Sigma_0 \cup \Sigma_2$ is complete and consistent, $\neg\theta \in \Sigma_0 \cup \Sigma_2$. Since Σ_0 is complete, $\{\theta, \neg\theta\} \subseteq \Sigma_0$, contrary to the consistency of Σ_0.

(2) implies (1): Suppose $\varphi \models \psi$, where $\varphi, \psi \in F$ and $\tau_0 = \tau(\varphi) \cap \tau(\psi)$. Let Σ_0 be the set of $\theta \in F'$ such that $\varphi \models \theta$ and $\tau(\theta) \subseteq \tau_0$. We show now that $\Sigma_0 \models \psi$. Otherwise there is a model $\mathfrak{M}$ of $\Sigma_0 \cup \{\neg\psi\}$. Let Σ_0^* be the set of $\theta \in F'$ such that $\mathfrak{M} \models \theta$ and $\tau(\theta) \subseteq \tau_0$. Note that Σ_0^* is complete. The set $\Sigma_0^* \cup \{\psi\}$ is consistent for otherwise the Compactness Theorem gives $\{\theta_1, \ldots, \theta_k\} \subseteq \Sigma_0^*$ such that $\psi \models \neg(\theta_1 \wedge \ldots \wedge \theta_k)$ implying $\neg(\theta_1 \wedge \ldots \wedge \theta_k) \in \Sigma_0$, a contradiction. Now both $\Sigma_0^* \cup \{\neg\psi\}$ and $\Sigma_0^* \cup \{\psi\}$ are consistent. By (2), $(\Sigma_0^* \cap F) \cup \{\varphi\} \cup \{\neg\psi\}$

is consistent, a contradiction. Having now proved $\Sigma_0 \models \psi$ we use again the Compactness Theorem. We obtain $\{\theta_1, \ldots, \theta_k\} \subseteq \Sigma_0$ such that $\{\theta_1, \ldots, \theta_k\} \models \psi$. Letting $\theta = \theta_1 \wedge \ldots \wedge \theta_k$ yields $\theta \in F'$, $\varphi \models \theta$, $\theta \models \psi$ and $\tau(\theta) \subseteq \tau(\varphi) \cap \tau(\psi)$. ◀

In the above theorem it is enough to assume that L' is countably compact, provided that L' satisfies the condition that there are only countably many sentences with a given countable vocabulary and the property $\mathrm{Rob}(L, L')$ is formulated for countable vocabularies only.

▶ **Definition 9.** *An abstract logic* $(S, F, \models, \tau)$ *is* classical*, if:*

1. $\tau(\varphi)$ *consists, for all* $\varphi \in F$*, of relation, constant and function symbols.*
2. S *is a class of structures* $\mathfrak{M}$ *in the sense of first order logic and* $\tau(\mathfrak{M})$ *has its usual meaning as the vocabulary of the structure* $\mathfrak{M}$*.*
3. *If* $\mathfrak{M} \in S$ *and* $a_1, \ldots, a_n \in M$*, we assume that the structure* $(\mathfrak{M}, a_1, \ldots, a_n)$*, obtained from* $\mathfrak{M}$ *by distinguishing the elements* $a_1, \ldots, a_n$ *as interpretations of new constant symbols* $c_1, \ldots, c_n$ *is also in* S*.*
4. *If* σ *is a subvocabulary of* $\tau(\mathfrak{M})$*, then the* reduct $\mathfrak{M} \restriction \sigma$ *is in* S*.*
5. *If* $\mathfrak{M} \in S$ *and* $\varphi \in F$*, then* $\mathfrak{M} \models \varphi$ *iff* $\mathfrak{M} \restriction \tau(\varphi) \models \varphi$*.*
6. L *satisfies renaming*[2] *in the sense of Definition 1.1.1 of [2].*
7. *A* formula[3] $\varphi(x_1, \ldots, x_n)$ *of* L *is a sentence* $\varphi(c_1, \ldots, c_n)$*, where* $c_1, \ldots, c_n$ *are new constant symbols.*

A classical logic L *is* atomic *if for every relation symbol* $R(x_1, \ldots, x_n)$ *and constant symbols* $c_1, \ldots, c_n$ *there is a sentence* $\varphi(c_1, \ldots, c_n) \in F$*, such that* $\tau(\varphi(c_1, \ldots, c_n)) = \{R, c_1, \ldots, c_n\}$ *and for all* $\mathfrak{M} \in S$ *with* $\tau(\varphi) \subseteq \tau(\mathfrak{M})$*, and all* $a_1, \ldots, a_n \in M$*.*

$$(\mathfrak{M}, a_1, \ldots, a_n) \models \varphi(c_1, \ldots, c_n) \iff (a_1, \ldots, a_n) \in R^{\mathfrak{M}}.$$

We write $\varphi(c_1, \ldots, c_n)$ *as* $R(c_1, \ldots, c_n)$*.*

Apart from ordinary first order logic and its extension by generalized quantifiers, infinitary logical operations, and higher order quantifiers, first order logic with finite models is a classical atomic abstract logic. Also first order logic with ordered models, i.e. models with a distinguished linear order $<$, which is an element of every vocabulary considered, and does not disappear in a reduct, is a classical atomic abstract logic. The two-variable fragment of first order logic, where vocabularies consist only of binary and unary predicates symbols, constant symbols and contain no function symbols, and only two variables are used overall, is a classical atomic abstract logic.

A stronger but still natural and prevalent property of logics is *regularity* in the sense of [2, Chapter II].

---

[2] Essentially, renaming says that we can change symbols in a formula and truth is preserved, if we make the respective changes in structures.

[3] We use constant symbols to play the role of variables.

For classical logics it makes sense to talk about the cardinality of the models. In particular, we can formulate the important *Löwenheim property*: Every sentence with a model has a countable model.

An EC(L)-class ("E" for "elementary") is a subclass K of S such that for some $\varphi \in F$, $\mathfrak{M} \in K$ if and only if $\mathfrak{M} \models \varphi$. We then write $\tau(K) = \tau(\varphi)$. A PC(L)-class ("P" for "projective") is a subclass K of S such that $\tau(\mathfrak{M})$ is a fixed τ for $\mathfrak{M} \in K$, and for some $\varphi \in F$, $\mathfrak{M} \in K$ if and only if $\mathfrak{N} \models \varphi$ for some $\mathfrak{N}$ such that $\mathfrak{N} \restriction \tau = \mathfrak{M}$[4]. We then write $\tau(K) = \tau$. Here $\tau(\varphi)$ may have more sorts than τ[5]. If no new sort occur in $\tau(\varphi)$, then PC($L_{\omega\omega}$)-definability coincides with existential second order definability. We may consider the family of all PC(L)-classes an abstract logic in the obvious sense.

Craig [11] showed that his interpolation theorem has an equivalent formulation as a separation property:

▶ **Definition 10.** *The* separation property $\mathrm{Sep}(L, L')$ *holds if any two disjoint PC(L)-classes K_0 and K_1 with the same vocabulary τ can be separated by an EC(L')-class K, i.e. $K_0 \subseteq K$ and $K \cap K_1 = \emptyset$ with $\tau(K) = \tau$. A logic L satisfies the* separation property *if it satisfies* $\mathrm{Sep}(L, L)$. *Two logics L and L' satisfy the* Souslin-Kleene interpolation property, $\mathrm{SK}(L, L')$, *if every PC(L)-class K whose complement in the class $\{\mathfrak{M} \in S : \tau(\mathfrak{M}) = \tau(K)\}$ is also a PC(L)-class, is an EC(L')-class.*

Naturally $\mathrm{Sep}(L, L')$ implies $\mathrm{SK}(L, L')$, but not conversely ([28]). Note that $\mathrm{Sep}(L, L')$ implies $L \leq L'$, if L is closed under negation. The family of all PC(L)-classes can be construed as a logic itself, as is carefully explained in ([40]). This logic, denoted $\Delta(L)$, has similar regularity properties as L. For example, it is classical, atomic and closed under conjunction and negation if L is. It is the smallest extension of a regular abstract logic L to a regular abstract logic with the Souslin-Kleene Interpolation property ([40]). In fact, the Souslin-Kleene Interpolation property is also called the Δ-interpolation property (Chapter 2). If we limit ourselves to PC(L)-classes that do not add new sorts (i.e. the defining formula does not have new sorts, only new predicates, functions and constants) we obtain a more restrictive extension $\Delta^1_1(L)$ which behaves very much in the same way as $\Delta(L)$. Of course, $\Delta^1_1(L) \subseteq \Delta(L)$.

▶ **Example 11.** The class of models (M, E), where $E \subseteq M \times M$ is an equivalence relation with uncountably many uncountable classes, is definable in $\Delta(L(Q_1))$ but not in $L(Q_1)$ (see Theorem 19, especially its proof). Hence $\Delta(L(Q_1))$ is a proper extension of $L(Q_1)$.

We prove the following simple result, generalizing a result from [11], in some detail because we assume in a sense the minimal amount of the logics, so it is not so clear that the classical proof works. It is perhaps interesting to see where the different assumptions are used.

---

[4] For example, the class K of infinite models of the empty vocabulary is a PC($L_{\omega\omega}$)-class as we can let φ be the first order sentence of vocabulary $\{<\}$ which says that $<$ is a linear order without last element.

[5] This will be relevant when we discuss second order logic.

▶ **Proposition 12.** *Assume L and L' are classical and satisfy $S = S'$. Suppose also that L is closed under negation. The following conditions are equivalent:*

1. $\mathrm{Craig}(L, L')$.
2. $\mathrm{Sep}(L, L')$

Proof. Assume (1). Suppose K_0 and K_1 are disjoint $\mathrm{PC}(L)$-classes and $\tau = \tau(K_0) = \tau(K_1)$. Let $\varphi_0 \in F$ with $\mathfrak{M} \in K_0$ if and only if $\mathfrak{N} \models \varphi_0$ for some $\mathfrak{N}$ such that $\mathfrak{N} \restriction \tau = \mathfrak{M}$ and $\varphi_1 \in F$ with $\mathfrak{M} \in K_1$ if and only if $\mathfrak{N} \models \varphi_1$ for some $\mathfrak{N}$ such that $\mathfrak{N} \restriction \tau = \mathfrak{M}$. Using the renaming property of L we can change the non-logical symbols of φ_1, other than those in τ, to something completely new, and thereby make sure $\tau = \tau(\varphi_0) \cap \tau(\varphi_1)$. Using closure of L under negation, we can find $\neg\varphi_1 \in F$. Since K_0 and K_1 are disjoint, $\varphi_0 \models \neg\varphi_1$. By (1), there is $\theta \in F'$ such that $\varphi_0 \models \theta$, $\theta \models \varphi_1$ and $\tau(\theta) \subseteq \tau$. Now the $\mathrm{EC}(L')$-class $K = \{\mathfrak{M} \in S : \mathfrak{M} \models \theta\}$ separates K_0 and K_1 i.e. $K_0 \subseteq K$ and $K \cap K_1 = \emptyset$.

Assume (2). Suppose $\varphi \models \psi$, where $\varphi, \psi \in F$. Let $\tau = \tau(\varphi) \cap \tau(\psi)$. Since L is closed under negation, $\neg\psi \in F$. Let $K_0 = \{\mathfrak{M} \restriction \tau : \mathfrak{M} \in S, \mathfrak{M} \models \varphi\}$ and $K_1 = \{\mathfrak{M} \restriction \tau : \mathfrak{M} \in S, \mathfrak{M} \models \neg\psi\}$. Now $K_0 \cap K_1 = \emptyset$. By (2) there is an $\mathrm{EC}(L')$-class K such that $K_0 \subseteq K$, $K \cap K_1 = \emptyset$ and $\tau(K) = \tau$. Let $\theta \in F'$ such that $\tau(\theta) = \tau$ and $K = \{\mathfrak{M} \in S : \tau(\mathfrak{M}) = \tau \text{ and } \mathfrak{M} \models \theta\}$. Now $\varphi \models \theta$, $\theta \models \psi$, and $\tau(\theta) \subseteq \tau(\varphi) \cap \tau(\psi)$. ◀

The assumption about being closed under negation is essential as the following example from [24] shows: Let L be first order logic, with negation only in front of atomic formulas, added with the generalized quantifier "there are infinitely many x such that ...", i.e. the quantifier Q_0. Every $\mathrm{PC}(L)$-class is $\mathrm{PC}(L_{\omega\omega})$, as can be proved by induction on $\mathrm{PC}(L)$-definitions, and therefore separation holds for L, even $\mathrm{Sep}(L, L_{\omega\omega})$. On the other hand, the proof of Theorem 19 shows that $\mathrm{Craig}(L)$ fails.

A famous consequence of the Craig Interpolation Theorem is the Beth Definability Theorem ([7]), which can be formulated as follows in the abstract logic setting:

▶ **Definition 13.** *Suppose L and L' are classical, atomic and closed under conjunction, and negation.*

1. *We say that* $\mathrm{Beth}(L, L')$ *holds if every $\varphi \in F$ and every predicate symbol $P \in \tau(\varphi)$ of arity n satisfy: If φ' denotes φ with P renamed as $P' \notin \tau(\varphi)$ and $\varphi \wedge \varphi' \models P(c_1, \ldots, c_n) \leftrightarrow P'(c_1, \ldots, c_n)$, where $c_1, \ldots, c_n \notin \tau(\varphi)$, then there is $\theta \in F'$ such that $P \notin \tau(\theta)$ and $\varphi \models P(c_1, \ldots, c_n) \leftrightarrow \theta$. A logic L has the* Beth property, $\mathrm{Beth}(L)$, *if* $\mathrm{Beth}(L, L)$ *holds.*
2. *We say that* $\mathrm{WBeth}(L, L')$ *holds if the above condition for* $\mathrm{Beth}(L, L')$ *holds with the additional assumption that every $\mathfrak{M} \in S$ with $\tau(\mathfrak{M}) = \tau(\varphi) \setminus \{P\}$ can be expanded to a model $\mathfrak{N} \in S$ such that $\mathfrak{N} \models \varphi$. A logic L has the* weak Beth property *([21]),* $\mathrm{WBeth}(L)$, *if* $\mathrm{WBeth}(L, L)$ *holds.*

Clearly, $\mathrm{Beth}(L, L')$ implies $\mathrm{WBeth}(L, L')$. Note also, that $\mathrm{WBeth}(L, L')$ implies $L \leq L'$.

It is interesting to note that the fact that first order logic has the Beth property, was proved by Beth well before the Craig Interpolation Theorem was published. The Beth property is

sometimes formulated with a theory in place of our single sentence φ. The two formulations are equivalent for $L_{\omega\omega}$, thanks to the Compactness Property.

The following establishes an easy connection between interpolation and the Beth property:

▶ **Proposition 14.** *Suppose L and L' are classical abstract logics, closed under conjunction and negation, and $L \leq L'$. Then* $\mathrm{Craig}(L, L')$ *implies* $\mathrm{Beth}(L, L')$.

Proof. Suppose $\varphi \in F$ such that if φ' denotes φ with P renamed as P', which is not in $\tau(\varphi)$, then $\varphi \wedge \varphi' \models P(c_1, \ldots, c_n) \leftrightarrow P'(c_1, \ldots, c_n)$. Thus $\varphi \wedge P(c_1, \ldots, c_n) \models \varphi' \rightarrow P'(c_1, \ldots, c_n)$. By $\mathrm{Craig}(L, L')$ there is $\theta \in F'$ such that $\varphi \wedge P(c_1, \ldots, c_n) \models \theta$, $\theta \models \varphi' \rightarrow P'(c_1, \ldots, c_n)$, and $\tau(\theta) = (\tau(\varphi) \setminus \{P\}) \cup \{c_1, \ldots, c_n\}$. Then $\varphi \models P(c_1, \ldots, c_n) \leftrightarrow \theta$. ◀

Even compactness does not help to prove the converse: The logic $L(Q^{\mathrm{cof}}_{\leq 2^\omega})$ (see section 3) has an extension (the so-called "Beth-closure" of $L(Q^{\mathrm{cof}}_{\leq 2^\omega})$) which has the Beth property and is compact but it does not satisfy interpolation, not even Souslin-Kleene interpolation [49]. However, there is a stronger form of the Beth property, *projective* $\mathrm{Beth}(L)$, which is actually equivalent to $\mathrm{Craig}(L)$ for regular logics ([2, p. 76]). Interpolation cannot be weakened to Souslin-Kleene interpolation in Proposition 14, as the Δ-closure of $L(Q^{\mathrm{cof}}_{\leq 2^\omega})$ does not have the Beth property [38, 50].

▶ **Definition 15.** **1.** *A classical abstract logic $L = (S, F, \models, \tau)$ is* fully classical *if S consists of* all *structures of first order logic.*

2. *The logic L satisfies* relativization *if for every $\varphi \in F$ and every formula $\psi(x) \in F$ there is a sentence $\theta \in F$ such that for all structures $\mathfrak{M}$ we have $\mathfrak{M} \models \theta$ if and only if the relativization*[6] $\mathfrak{M}^{(A)}$ *of $\mathfrak{M}$ to A satisfies φ, where A is the set of $a \in M$ such that $\mathfrak{M} \models \psi(a)$, and it is assumed that A is closed under the functions of $\mathfrak{M}$ and also contains the constants of $\mathfrak{M}$.*

3. *The classical logic L satisfies* isomorphism closure, *if $\mathfrak{M} \models \varphi \iff \mathfrak{N} \models \varphi$ whenever $\mathfrak{M}, \mathfrak{N} \in S$, $\mathfrak{M} \cong \mathfrak{N}$, and $\varphi \in F$.*

Examples of fully classical abstract logics satisfying relativization and isomorphism closure are first order logic, $L_{\omega_1\omega}$, second order logic as well as many extensions of first order logic by generalized quantifiers.

▶ **Theorem 16** ([37])**.** *Assume that the abstract logic L is fully classical, satisfies both relativization and isomorphism closure, and has the property that the vocabulary of each sentence is finite.*[7] *If L has the (many-sorted) Robinson property then it satisfies the Compactness Theorem.*

---

[6] The relativization $\mathfrak{M}^{(A)}$ of $\mathfrak{M}$ to A is the structure which has A as the domain and interpretations of the non-logical symbols as follows: The n-ary relation symbols R in the vocabulary of $\mathfrak{M}$ are interpreted as $R^{\mathfrak{M}} \cap A^n$. The n-ary function symbols f are interpreted as restrictions of $f^{\mathfrak{M}}$ to A^n. Finally, the constant symbols are interpreted in the same way as in $\mathfrak{M}$.

[7] This is called the "Finite Occurrence property". It essentially means that each sentence is a finite string a symbols. It follow easily from compactness.

Proof. We present a rough sketch only and refer to [37] for a detailed proof. We indicate the main idea of proving countable compactness which should give a good idea of the proof in the general case. Suppose $T = \{\varphi_n : n < \omega\}$ is a counter-example to countable compactness. For each n there is $\mathfrak{M}_n \models \{\varphi_m : m < n\}$. W.l.o.g. the domains of the models $\mathfrak{M}_n$ are disjoint and the structures $\mathfrak{M}_n$ are relational. Let $\mathfrak{A}$ be the disjoint union of the models $\mathfrak{M}_n$, $n < \omega$. We expand $\mathfrak{A}$ with a new sort s consisting of a copy of the natural numbers with their natural order $<$ as well as a function f which maps each element of M_n to the n'th natural number in the order $<$. Let $\mathfrak{A}'$ be the expansion. Now in any model $\mathfrak{B}$ that is L-equivalent to $\mathfrak{A}'$ the order-type of $<^{\mathfrak{B}}$ must be ω, for if there were a non-standard number b in the sort s part of B, its pre-image $(f^{\mathfrak{B}})^{-1}(b)$ would determine a model in which each φ_n is true, contrary to the inconsistency of T. Let τ_0 consist of the sort $\{s\}$ and unary predicates P_n, $n < \omega$, of sort s. Let $\mathfrak{A}^*$ be the expansion of $\mathfrak{A}'$ by letting $P_n^{\mathfrak{A}^*}$ consist of the first n elements of $<$ for all $n < \omega$. Let $\mathfrak{A}''$ be a new $\tau_0 \cup \{c\}$-structure, where c is a new constant of sort s, which has $\omega + 1$ as its domain and $P_n^{\mathfrak{A}''} = \{0, \ldots, n-1\}$ as well as $c^{\mathfrak{A}''} = \omega$. By isomorphism closure and the finite occurrence assumption, $\mathfrak{A}^* \restriction \tau_0 \equiv_L \mathfrak{A}'' \restriction \tau_0$. By the Robinson property, there is $\mathfrak{B}$ such that $\mathfrak{B} \restriction \tau(\mathfrak{A}^*) \equiv_L \mathfrak{A}^*$ and $\mathfrak{B} \restriction \tau(\mathfrak{A}'') \equiv_L \mathfrak{A}''$. This contradicts the inconsistency of T, as $(f^{\mathfrak{B}})^{-1}(c^{\mathfrak{B}})$ gives rise to a model of each φ_n. ◀

The assumption of the finite occurrence property in Theorem 16 can be considerably weakened, see [37].

To present an elementary proof of the Robinson property of first order logic, we recall the following game, due to A. Ehrenfeucht [16]: Let L be a finite relational vocabulary and $\mathfrak{A}, \mathfrak{B}$ L-structures such that $A \cap B = \emptyset$. We use $\mathrm{EF}_n(\mathfrak{A}, \mathfrak{B})$ to denote the n-move Ehrenfeucht-Fraïssè game on $\mathfrak{A}$ and $\mathfrak{B}$. During each round of the game player I first picks an element from one of the models, and then player II picks an element from the other model. In this way a relation $p = \{(a_1, b_1), \ldots, (a_n, b_n)\} \subseteq A \times B$ is built. If p is a partial isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$, player II is the winner of this play. Player II has a winning strategy in this game if and only if the models $\mathfrak{A}$ and $\mathfrak{B}$ satisfy the same first order sentences of quantifier rank at most n. If player II has a winning strategy τ in $\mathrm{EF}_n(\mathfrak{A}, \mathfrak{B})$, the set I_i of sets of positions $\{(a_1, b_1), \ldots, (a_j, b_j)\} \subseteq A \times B$ which can be continued to a position $\{(a_1, b_1), \ldots, (a_{n-i}, b_{n-i})\} \subseteq A \times B$ in which player II has used τ, form an increasing chain $I_n \subseteq I_{n-1} \subseteq \ldots \subseteq I_0$ known as a *back-and-forth sequence*, introduced by R. Fraïssè [20]. The name derives from the fact that if $p \in I_{i+1}$, $|p| = k$, and $a \in M$ (or $b \in M'$), then there is $b \in M'$ (respectively, $a \in M$) such that $p \cup \{(a, b)\} \in I_i$. In this case we say that I_{i+1} satisfies the k-back-and-forth condition w.r.t. I_i for k-sequences between the models $\mathfrak{A}$ and $\mathfrak{B}$. Conversely, if a back-and-forth sequence $I_n \subseteq I_{n-1} \subseteq \ldots \subseteq I_0$ exists, then player II can use it to win the game $\mathrm{EF}_n(\mathfrak{A}, \mathfrak{B})$. So back-and-forth sequences and winning strategies of II go hand in hand. This explains why the game is generally called the Ehrenfeucht-Fraïssè game. The infinite version, where players play ω moves, is denoted $\mathrm{EF}(\mathfrak{A}, \mathfrak{B})$.

The following proof of the Robinson property for $L_{\omega\omega}$, using the back-and-forth method just described, is due to Per Lindström (see [34]). This is the argument that became the proof of the so-called Lindström's Theorem ([33]).

▶ **Theorem 17** ([33]). *Suppose L is a regular abstract logic in the sense of [2, Chapter II] such that L is compact, has the Löwenheim property, and has the property that the vocabulary of each sentence is finite.*[8] *Then L has the Robinson property and, in fact, $L \equiv L_{\omega\omega}$.*

Proof. In the Robinson property, suppose Σ_1 and Σ_2 are consistent extensions of a complete theory Σ_0. Let τ_1 be the vocabulary of Σ_1, τ_2 that of Σ_2, and τ that of Σ_0, where $\tau = \tau_1 \cap \tau_2$. By assumption, there is a model $\mathfrak{M}_1$ of Σ_1 and a model $\mathfrak{M}_2$ of Σ_2 and then $\mathfrak{M}_1 \restriction \tau \equiv \mathfrak{M}_2 \restriction \tau$. Thus there is, for any $n \in \mathbb{N}$, a back-and-forth sequence $(I_i : i \leq n)$ for $\mathfrak{M}_1 \restriction \tau$ and $\mathfrak{M}_2 \restriction \tau$. Let τ_2' be a copy of τ_2 such that $\tau_2 \cap \tau_2' = \emptyset$. Let τ' be the vocabulary resulting from τ in this translation. Let $\mathfrak{M}_2'$ be the translation of $\mathfrak{M}_2$ to the vocabulary τ_2'. Let R be a new unary and $<$ a new binary predicate symbol. Let Γ be the set of first order sentences which state:

1. The complete first order τ_1-theory of $\mathfrak{M}_1$.
2. The complete first order τ_2'-theory of $\mathfrak{M}_2$.
3. $(R, <)$ is a non-empty linear order in which every element with a predecessor has an immediate predecessor.
4. First order sentences which state, by means of new $2n+1$-ary predicates I^n, that there is, for each $n < \omega$, an n-back-and-forth sequence $I^n(i, \cdot, \cdot)$, parametrized by $i \in R$, between the τ-reduct and the τ'-reduct of the universe, meaning that if $i - 1$ denotes the predecessor (if it exists) of i in R, then $I^n(i, \cdot, \cdot)$ satisfies the n-back-and-forth condition w.r.t. $I^{n+1}(i-1, \cdot, \cdot)$ for n-sequences between the τ-reduct and the τ'-reduct of the universe.

For all $n \in \mathbb{N}$ there is a model of Γ with $(R, <)$ of length n. By the Compactness Theorem, there is a model $\mathfrak{N}$ of Γ with $(R, <)$ non-well-founded. By the Löwenheim property we may assume $\mathfrak{N}$ is countable. Let $\mathfrak{N}_1 = \mathfrak{N} \restriction \tau_1$ and $\mathfrak{N}_2 = \mathfrak{N} \restriction \tau_2'$. Since $(R, <)^{\mathfrak{N}}$ is non-well-founded, player II has a winning strategy for the infinite game $\mathrm{EF}(\mathfrak{N}_1, \mathfrak{N}_2)$. Since $\mathfrak{N}_1$ and $\mathfrak{N}_2$ are countable, $\mathfrak{N}_1 \cong \mathfrak{N}_2$. This implies that $\mathfrak{N}_1$ can be expanded to a model of $\Sigma_1 \cup \Sigma_2$. This ends the proof of Robinson property.

Lindström observed that the above argument works for any logic which is countably compact and has the Löwenheim property. But are there such extensions of $L_{\omega\omega}$? To see why there are none, let us write $\mathfrak{M} \sim_n \mathfrak{M}'$ if player II has a winning strategy in $\mathrm{EF}_n(\mathfrak{A}, \mathfrak{B})$. This equivalence relation divides the class of all models with a finite vocabulary L into a finite number of equivalence classes C, each definable by a first order sentence of quantifier rank at most n, namely the conjunction of all first order sentence of quantifier rank at most n that

[8] This actually follows from compactness with an easy argument.

are true in one (hence all) models in C. Suppose now $\varphi \in F$ is not first order definable, i.e. there is no first order ψ such that $\varphi \equiv \psi$ and $\tau(\varphi) = \tau(\psi)$. By our assumption, $\tau(\varphi)$ is finite. There is no n such that the class of models of φ is closed under $\sim_n$. In other words, for all n there are models $\mathfrak{M}_n$ and $\mathfrak{N}_n$ such that $\mathfrak{M}_n \sim_n \mathfrak{N}_n$, $\mathfrak{M}_n \models \varphi$ and $\mathfrak{N}_n \models \neg\varphi$. By countable compactness there are elementarily equivalent $\mathfrak{M}$ and $\mathfrak{N}$ such that $\mathfrak{M} \models \varphi$ and $\mathfrak{M} \models \neg\varphi$. Now we can continue as above and obtain two isomorphic models, one of φ and the other of $\neg\varphi$, a contradiction. ◀

An amusing corollary of the above Lindström Theorem is that $L_{\omega\omega}$ satisfies the Souslin-Kleene Interpolation theorem: Consider $\Delta(L_{\omega\omega})$. It is a regular logic, compact and has the Löwenheim property. Hence $\Delta(L_{\omega\omega}) \equiv L_{\omega\omega}$. There is no similarly easy way (as far as is known) to deduce the Beth property or the Craig Interpolation property directly from Lindström's Theorem.

The proof of Theorem 17 actually gives the following slightly more general result:

▶ **Theorem 18** ([18], see also [24]). *Suppose L is a regular abstract logic, except that L is* not *assumed to be closed under negation. If L is compact, satisfies the Löwenheim property, and has the property that the vocabulary of each sentence is finite, then any two disjoint $EC(L)$-classes can be separated by an $EC(L_{\omega\omega})$-class.*

For Lindström Theorems for fragments of first order logic we refer to [6].

3 Generalized Quantifiers

Sometimes one may want a logic that can express some particular property, be it cardinality, cofinality, connectedness, well-foundedness, or some other structural property. One way to accomplish this is by means of the concept of a generalized quantifier, introduced in [44] and extended in [32]. Generalized quantifiers allow one to add a particular feature to a logic. Unsurprisingly, there is no guarantee that such an addition results in a nice logic, e.g. a logic with interpolation. To obtain a nice logic one probably has to add many generalized quantifiers. But then another problem arises. Any logic (closed under substitution) can be represented as the result of adding a number of generalized quantifiers to first order logic. Since it is unlikely that every logic is "nice" in any sense, it makes sense to limit oneself to adding just one or finitely many generalized quantifiers.

The first generalized quantifier to be considered (in [44]) was

$$Q_\alpha x\varphi(x) \iff |\{a : \varphi(a)\}| \geq \aleph_\alpha.$$

Here α is a fixed ordinal chosen in advance. The best-known cases are $\alpha = 0$ or $\alpha = 1$. The extension $L(Q_\alpha)$ of first order logic by this quantifier has the Löwenheim property for $\alpha = 0$ and is countably compact for $\alpha = 1$. The following simple result, due to Keisler, is from [35]:

▶ **Theorem 19.** *$L(Q_\alpha)$ does not have the interpolation property.*

Proof. Let E be a binary relation symbol and S, S' unary relation symbols. Let φ the conjunction of

(1) $\forall x(xEx) \wedge \forall x\forall y(xEy \to yEx) \wedge \forall x\forall y\forall z((xEy \wedge yEz) \to xEz))$.

(2) $\forall x\exists y(xEy \wedge S(y))$.

(3) $\forall x\forall y((S(x) \wedge S(y) \wedge xEy) \to x = y)$.

(4) $Q_\alpha xS(x)$.

Let ψ the sentence

(5) $\forall x\exists y(S'(y) \wedge xEy) \to Q_\alpha xS'(x)$.

Clearly, $\varphi \models \psi$. Suppose $\varphi \models \theta$ and $\theta \models \psi$, where $\theta \in L(Q_1)$ has vocabulary $\{E\}$ only. Then θ says that E is an equivalence relation with at least $\aleph_\alpha$ equivalence classes. But we now use an Ehrenfeucht-Fraïssè game to show that such a θ cannot exist. In the Ehrenfeucht-Fraïssè game of $L(Q_\alpha)$ on two models $\mathfrak{M}$ and $\mathfrak{N}$ players build a partial isomorphism between $\mathfrak{M}$ and $\mathfrak{N}$. Suppose a partial isomorphism p has been built. In addition to the usual moves of the Ehrenfeucht-Fraïssè game of first order logic, player I has the option of choosing a subset X of cardinality $\geq \aleph_\alpha$ of one of the models, say $\mathfrak{M}$, after which player II chooses a subset Y of size $\geq \aleph_\alpha$ of the other model, in this case $\mathfrak{N}$. After this player I chooses an element $y \in Y$ and player II responds by choosing an element $x \in X$. The game continues now with the partial mapping $p \cup \{(x, y)\}$ while X and Y are abandoned. It is not hard to show that if II has a winning strategy in such a game of any finite length, then the models are $L(Q_\alpha)$-equivalent (see e.g. [54, Chapter 10]). For the current case we can use a model $\mathfrak{M}$ which is an equivalence relation with $\aleph_\alpha$ classes, each of them of size $\aleph_\alpha$. For $\mathfrak{N}$ we can use an equivalence relation with $\aleph_0$ classes, each of size $\aleph_\alpha$. The winning strategy of player II is the following. Suppose player I has played a set $X \subseteq M$, and player II now has to choose the set $Y \subseteq N$. Let A be the union of all equivalence classes of elements of dom(p), and B the union of all equivalence classes of elements of ran(p). If $X \setminus A \neq \emptyset$, we let II choose $N \setminus B$. This is clearly a move that keeps II in the game. We can therefore assume w.l.o.g. that $X \subseteq A$. Since $X \setminus \text{dom(p)}$ is infinite, there is $a \in \text{dom(p)}$ such that $X \setminus \text{dom(p)}$ meets the equivalence class of a. Now we let II choose Y to be what is in the equivalence class of $p(a)$ outside ran(p). This is clearly a winning strategy for any finite number of moves. This contradicts the fact that θ separates $\mathfrak{M}$ and $\mathfrak{N}$. ◀

The proof also shows that even Souslin-Kleene Interpolation fails for $L(Q_\alpha)$. In fact, the proof shows[9] that $\Delta(L(Q_\alpha)) \not\leq L_{\infty\omega}(Q_\alpha)$. The Beth theorem fails for $L(Q_1)$ ([21]), but surprisingly, it is consistent, relative to the consistency of ZF, that the weak Beth property holds for $L(Q_1)$ ([43]).

▶ **Theorem 20** ([45]). *$L(Q_0)$ does not have the weak Beth property.*

[9] This is because the strategy of player II works not only for any finite game but even for the game of length ω.

Proof. The crucial property of $L(Q_0)$ here is that it can express "every natural number has only finitely many predecessors". Let $\varphi \in L(Q_0)$ say of its models (A, E, R) that (A, E) is a model of a suitable finite part T of ZFC and either the natural numbers of (A, E) have non-standard elements[10] and $R = \emptyset$ or the natural numbers of (A, E) are standard[11] and R is the set of pairs (η, f), where (A, E) satisfies

1. $\eta \in L(Q_0)$, and
2. f is a function such that the inductive clauses for satisfaction of $L(Q_0)$-formulae of the vocabulary $\{E\}$ hold.

An example of the inductive clauses here is:

- $(Q_0 x\eta, f) \in R$ if and only if for infinitely many y there exists $g \in A$ such that $g(z) = f(y)$ for variables $z \neq x$, $g(x) = y$ and $(\eta, g) \in R$.

If (A, E, R) and (A, E, R') are models of φ, and (w.l.o.g.) the integers of (A, E) are all standard, then one can use induction on formulas of $L(Q_0)$ of the vocabulary $\{E\}$ to prove that $R = R'$. Moreover, for all (A, E) there is always an R such that $(A, E, R) \models \varphi$. Suppose that there were a formula $\eta(x, y) \in L(Q_0)$ which defines R explicitly in models of φ. Let $\kappa > \omega$ be a cardinal such that $\mathfrak{M} = (H_\kappa, \in)$, where H_κ is the set of sets whose transitive closure has cardinality $< \kappa$, is a model of T. If now R is chosen such that $(H_\kappa, \in, R) \models \varphi$, then

$$R = \{(\varphi, f) : \varphi \in L(Q_0) \text{ and } f \text{ satisfies } \varphi \text{ in } \mathfrak{M}\}.$$

Combining this with the choice of $\eta(x, y)$ yields

$$\mathfrak{M} \models \eta(\varphi, f) \text{ if and only if } f \text{ satisfies } \varphi \text{ in } \mathfrak{M}.$$

The standard diagonal argument ends the proof. To this end, let $\xi(= \xi(x))$ be the formula $\neg\eta(x, f)$, where f is a term denoting the function which maps the variable x to x ($f = \{(x, x)\}$). We now have $\xi \in H_\kappa$ and

$$\mathfrak{M} \models \xi(\xi) \leftrightarrow \eta(\xi, \{(x, x)\}) \leftrightarrow \neg\xi(\xi).$$

This contradiction shows that φ constitutes a counter-example to the weak Beth property for the logic $L(Q_0)$. ◀

The point of the above proof is that with $L(Q_0)$ we can capture the standard natural numbers and thereby use induction on objects, which are formulas "merely" in the sense of the model (A, E). Otherwise it does not matter what the new quantifier Q_0 says. The same proof gives a result about any generalized quantifiers $Q^1, \ldots, Q^n$ in the sense of [32]: If $L(Q^1, \ldots, Q^n) \not\equiv L_{\omega\omega}$ has Löwenheim property, then the weak Beth property fails ([33]).

---

[10] I.e. elements which have infinitely many predecessors. This is where we use Q_0.
[11] I.e. they all have only finitely many predecessors. This is again where we use Q_0.

If we look how much we have to add to $L(Q_0)$ to obtain weak Beth (or even interpolation), the optimal answer is $\mathrm{WBeth}(L(Q_0), L_{\mathrm{HYP}})$, where HYP is the smallest admissible set containing ω (see [1] for details) and $L_{\mathrm{HYP}} = L_{\omega_1\omega} \cap \mathrm{HYP}$. Thus we have to resort to infinitary propositional operations to obtain Beth definability for $L(Q_0)$, and then we get full interpolation as a bonus.

If we want interpolation for a logic of the form $L(Q^1, \ldots, Q^n)$, we need to have countable (or at least "recursive") compactness. However, no proper extension of first order logic is known which has both interpolation and countable compactness.

Problem: Are FO quantifiers the only generalized quantifiers which give rise to a regular logic with the Craig Interpolation property? The same for the Beth property and Souslin-Kleene. Does weak Beth hold for $L(Q_1)$ provably in ZFC?

Problem: Is there any proper extension of first order logic that is (countably) compact and has the interpolation property?

One of the most notorious generalized quantifiers is the cofinality quantifier Q^{cof}_κ ([48]). For a regular κ it is defined as follows:

$$\mathfrak{M} \models Q^{\mathrm{cof}}_\kappa xy\varphi(x, y, \vec{a}) \iff \{(c, d) : \mathfrak{M} \models \varphi(c, d, \vec{a})\} \text{ is a linear order of cofinality } \kappa.$$

What is remarkable about this quantifier is that it is compact in a vocabulary of any cardinality, which is why it is called "fully compact". It has also a nice complete axiomatization. However, the logic $L(Q^{\mathrm{cof}}_\omega)$ does not have interpolation or Beth properties ([39]). Still, interestingly, there is a countably compact logic $L(aa)$ with $\mathrm{Craig}(L(Q^{\mathrm{cof}}_\omega), L(aa))$, see Theorem 27.

4 Infinitary Logics

Infinitary logics, such as $L_{\omega_1\omega}$, are based on a completely different idea than generalized quantifiers. Here we add new logical operations[12] which resemble the familiar logical operations of $L_{\omega\omega}$ but are "infinitary". While in the case of generalized quantifiers we did not mind if the meaning of formulas became "infinitary", as in "there are infinitely many" and "there are uncountably many", here we do not mind if the formulas themselves become syntactically "infinitary", as in $\varphi_0 \vee \varphi_1 \vee \ldots$. The advantage of this change of perspective is that we obtain lots of examples of extensions of first order logic with interpolation. Since we lose compactness (apart from so-called Barwise compactness), we cannot obtain the Robinson property, the model theoretic version of interpolation. The first proofs of interpolation for infinitary logics were indeed proof-theoretic.

[12] In $L_{\kappa\lambda}$ conjunctions and disjunctions of sets of formulas of size $< \kappa$ are allowed as well as quatification of sequences of variables of length $< \lambda$. Then $L_{\infty\lambda}$ is $\bigcup_\kappa L_{\kappa\lambda}$ and $L_{\infty\infty}$ is $\bigcup_\lambda L_{\infty\lambda}$.

A paradigmatic example of an $L_{\omega_1\omega}$-sentence is $\forall x \bigvee_n (x = s^n(0))$ which says of a unary function s and a constant 0 that every element is either 0 or obtained from 0 by iterating the function s, taking $s^0(0)$ to be 0.

We embrace now fully the many-sorted approach to logic as it yields a particularly powerful form of interpolation. Thus we have variables x^s of different sorts s. Let $\text{sort}(\tau)$ denote the set of sorts of the vocabulary τ.

Let us recall the auxiliary concept of a Hintikka set: Suppose τ is countable, C^s is a countable set of new constant symbols for $s \in \text{sort}(\tau)$, and $\tau' = \tau \cup C^*$, where $C^* = \bigcup_s C^s$. A *Hintikka set* is any set H of τ'-sentence of $L_{\omega_1\omega}$, which satisfies:

1. $t = t \in H$ for every constant τ'-term t.
2. If $\varphi(t) \in H$, $\varphi(t)$ atomic, and $t = t' \in H$, then $\varphi(t') \in H$.
3. If $\neg\varphi \in H$, then $\varphi\neg \in H$.[13]
4. If $\bigvee_n \varphi_n \in H$, then $\varphi_n \in H$ for some n.
5. If $\bigwedge_n \varphi_n \in H$, then $\varphi_n \in H$ for all n.
6. If $\exists x^s \varphi(x^s) \in H$, then $\varphi(c) \in H$ for some $c \in C^s$.
7. If $\forall x^s \varphi(x^s) \in H$, then $\varphi(c) \in H$ for all $c \in C^s$.
8. For every constant τ'-term t of sort s there is $c \in C^s$ such that $t = c \in H$.
9. There is no atomic φ such that $\varphi \in H$ and $\neg\varphi \in H$.

The Hintikka set H is a Hintikka set *for* a sentence φ of $L_{\omega_1\omega}$ (or $L_{\omega\omega}$) if $\varphi \in H$. The basic property of Hintikka sets is that if $\varphi \in L_{\omega_1\omega}$ does have a model $\mathfrak{M}$, then there is a Hintikka set for φ, built directly from formulas true in $\mathfrak{M}$; and conversely, if there is a Hintikka set H for a given $\varphi \in L_{\omega_1\omega}$, then φ has a model, as we shall now see. A model $\mathfrak{M}$ is built from H as follows: Define on $C^* = \bigcup_s C^s$ an equivalence relation $c \sim c'$ by the condition $c = c' \in H$. Let the domain of sort s in $\mathfrak{M}$ be $M_s = \{[c] : c \in C^s\}$. The interpretations in $\mathfrak{M}$ are defined by $c^{\mathfrak{M}} = [c]$, $f^{\mathfrak{M}}([c_{i_1}], \ldots, [c_{i_n}]) = [c]$ for $c \in C^*$ such that $f(c_{i_1}, \ldots, c_{i_n}) = c \in H$. For any constant term t of sort s there is a $c \in C^s$ such that $t = c \in H$. It is easy to see that $t^{\mathfrak{M}} = [c]$. If R is an n-ary predicate symbol, we let $(t_1, \ldots, t_n) \in R^{\mathfrak{M}}$ if and only if $R(t_1, \ldots, t_n) \in H$. By induction on $\varphi(x_1, \ldots, x_n)$ one can now easily prove that if $d_1 \ldots, d_n \in C^*$ then,

$$\begin{aligned} \varphi(d_1, \ldots, d_n) \in H \quad &\Rightarrow \quad \mathfrak{M} \models \varphi(d_1, \ldots, d_n) \\ \neg\varphi(d_1, \ldots, d_n) \in H \quad &\Rightarrow \quad \mathfrak{M} \not\models \varphi(d_1, \ldots, d_n). \end{aligned}$$

In particular, $\mathfrak{M} \models \varphi$ for the φ we started with, since $\varphi \in H$.

How do we find useful Hintikka sets? The basic tool is the auxiliary concept of a consistency property. Roughly speaking, a consistency property is a set Δ of (usually) finite sets S which are consistent and Δ has information about how to extend S to a Hintikka set.

---

[13] $\varphi\neg$ is $\neg\varphi$, if φ is atomic, $(\neg\varphi)\neg$ is φ, $(\bigwedge_n \varphi_n)\neg$ is $\bigvee_n \neg\varphi_n$, $(\bigvee_n \varphi_n)\neg$ is $\bigwedge_n \neg\varphi_n$, $(\forall x^s \varphi)\neg$ is $\exists x^s \neg\varphi$, and $(\exists x^s \varphi)\neg$ is $\forall x^s \neg\varphi$

A *consistency property* is any set Δ of countable sets S of τ-formulas of $L_{\omega_1\omega}$, which satisfies the conditions:

1. If $S \in \Delta$, then $S \cup \{t = t\} \in \Delta$ for every constant τ'-term t.
2. If $\varphi(t) \in S \in \Delta$, $\varphi(t)$ atomic, and $t = t' \in S$, then $S \cup \{\varphi(t')\} \in \Delta$.
3. If $\neg\varphi \in S \in \Delta$, then $S \cup \{\varphi\neg\} \in \Delta$.
4. If $\bigvee_n \varphi_n \in S \in \Delta$, then $S \cup \{\varphi_n\} \in \Delta$ for some n.
5. If $\bigwedge_n \varphi_n \in S \in \Delta$, then $S \cup \{\varphi_n\} \in \Delta$ for all n.
6. If $\exists x^s \varphi(x^s) \in S \in \Delta$, then $S \cup \{\varphi(c)\} \in \Delta$ for some $c \in C^s$.
7. If $\forall x^s \varphi(x^s) \in S \in \Delta$, then $S \cup \{\varphi(c)\} \in \Delta$ for all $c \in C^s$.
8. For every constant L'-term t of sort s there is $c \in C^s$ such that $S \cup \{t = c\} \in \Delta$.
9. There is no atomic formula φ such that $\varphi \in S$ and $\neg\varphi \in S$.

The consistency property Δ is a consistency property *for* a set T of infinitary τ-sentences if for all $S \in \Delta$ and all $\varphi \in T$ we have $S \cup \{\varphi\} \in \Delta$.

A *τ-fragment* of $L_{\omega_1\omega}$ is any set $\mathcal{F}$ of formulas of $L_{\omega_1\omega}$ in the vocabulary τ such that $\mathcal{F}$ is closed under substitutions of terms, $\mathcal{F}$ contains the atomic τ-formulas, $\neg\varphi \in \mathcal{F}$ if and only if $\varphi \in \mathcal{F}$, $\wedge\Phi \in \mathcal{F}$ if $\Phi \subseteq \mathcal{F}$ is finite, $\vee\Phi \in \mathcal{F}$ if $\Phi \subseteq \mathcal{F}$, is finite, $\wedge\Phi \in \mathcal{F}$ if and only if $\vee\Phi \in \mathcal{F}$, and then $\Phi \subseteq \mathcal{F}$, $\forall x^s\varphi \in \mathcal{F}$ if and only if $\varphi \in \mathcal{F}$, $\exists x^s\varphi \in \mathcal{F}$ if and only if $\varphi \in \mathcal{F}$. Note that a fragment is necessarily closed under subformulas. It is easy to see that if $\varphi \in L_{\omega_1\omega}$ with a countable vocabulary τ, then there is a countable fragment $\mathcal{F} \subseteq L_{\omega_1\omega}$ such that $\varphi \in \mathcal{F}$.

▶ **Lemma 21.** *Let T be a countable set of τ-sentences of $L_{\omega_1\omega}$ or $L_{\omega\omega}$ and $\mathcal{F} \subseteq L_{\omega_1\omega}$ a countable fragment such that $T \subseteq \mathcal{F}$. Suppose Δ is a consistency property for T. Then for any $S \in \Delta$ there is a Hintikka set H for T such that $S \subseteq H$.*

Proof. We can define H as the union of an increasing sequence S_n, $n < \omega$, where $S_0 = S$. The sequence is constructed straightforwardly in such a way, maintaining judicious bookkeeping, that in the end the set H is a Hintikka set. ◀

Let $\mathrm{Un}(\varphi)$ be all sorts s such that a variable of sort s occurs universally quantified in φ. Similarly $\mathrm{Un}(S)$ for a set S of formulas. Let $\mathrm{Ex}(\varphi)$ be all sorts s such that a variable of sort s occurs existentially quantified in φ. Similarly $\mathrm{Ex}(S)$ for a set S of formulas. For example, suppose φ is $\forall x^1 \exists x^0 (x^0 = x^1)$. Then $\mathrm{Un}(\varphi) = \{1\}$ and $\mathrm{Ex}(\varphi) = \{0\}$. Suppose φ is $\forall x^0 \forall y^3 (R(x^0, y^3) \leftrightarrow R'(x^0, y^3))$. Then $\mathrm{Un}(\varphi) = \{0, 3\}$ and $\mathrm{Ex}(\varphi) = \emptyset$.

▶ **Theorem 22** ([17]). *Suppose $\varphi \models \psi$, where φ and ψ are sentences of $L_{\omega_1\omega}$ in a relational vocabulary. Then there is a sentence θ of $L_{\omega_1\omega}$ such that*

1. *$\varphi \models \theta$ and $\theta \models \psi$*
2. *$\tau(\theta) \subseteq \tau(\varphi) \cap \tau(\psi)$*
3. *$\mathrm{Un}(\theta) \subseteq \mathrm{Un}(\varphi)$ and $\mathrm{Ex}(\theta) \subseteq \mathrm{Ex}(\psi)$.*

Proof. Let us assume that the claim of the theorem is false and derive a contradiction. Since $\varphi \models \psi$, the set $\{\varphi, \neg\psi\}$ has no models. We construct a consistency property for $\{\varphi, \neg\psi\}$. It then follows that $\{\varphi, \neg\psi\}$ has a model, which is a contradiction. Let $\tau_1 = \tau(\varphi)$, $\tau_2 = \tau(\psi)$, and $\tau = \tau_1 \cap \tau_2$. Suppose $C^s = \{c^s_n : n \in \mathbb{N}\}$ is a set of new constant symbols for each sort s of $\tau_1 \cup \tau_2$. Let $C^* = \bigcup_s C^s$. Given a set S of sentences, let S_1 consists of all $\tau_1 \cup C^*$-sentences in S with only finitely many constant from C^*, and let S_2 consists of all $\tau_2 \cup C^*$-sentences in S with only finitely many constant from C^*. Let us say that θ *separates* S' and S'' if

1. $S' \models \theta$,
2. $S'' \models \neg\theta$,
3. $\mathrm{Un}'(\theta) \subseteq \mathrm{Un}(S')$,
4. $\mathrm{Ex}'(\theta) \subseteq \mathrm{Un}(S'')$,

where $\mathrm{Un}'(\theta)$ consists of sorts $s \in \mathrm{Un}(\theta)$ and sorts of constants $c \in C^*$ occurring in θ, and $\mathrm{Ex}'(\theta)$ consists of sorts $s \in \mathrm{Ex}(\theta)$ and sorts of constants $c \in C^*$ occurring in θ.

Let Δ consist of finite sets S of sentences of $L_{\omega_1\omega}$ such that $S = S_1 \cup S_2$ and:

($\star$) There is no $L \cup C^*$-sentence that separates S_1 and S_2.

Note that $\{\varphi, \neg\psi\} \in \Delta$. We show that Δ is a consistency property. This involves checking the nine conditions that a consistency property has to satisfy. Most are almost trivial. For example, let us look at condition 7. Consider $S \in \Delta$ and $\exists x^s \varphi(x^s) \in S_1$. Let $c_0 \in C^s$ be such that c_0 does not occur in S. Now the sets $S_1 \cup \{\varphi(c_0)\}$ and S_2 satisfy ($\star$). On the other hand, consider $S \in \Delta$ and $\exists x^s \varphi(x^s) \in S_2$. Let again $c_0 \in C^s$ be such that c_0 does not occur in S. Now the sets S_1 and $S_2 \cup \{\varphi(c_0)\}$ satisfy ($\star$). Theorem 22 is proved. ◀

If above $\varphi, \psi \in A$, where A is a countable admissible set, then also $\theta \in A$ [1].

Many sorted interpolation gives numerous preservation results of which we mention just one. It should be noted that we are dealing with infinitary logic, so we do not have a compactness theorem. Also, for example, it is not true that every formula has a prenex normal form ([14]). On the other hand, Theorem 22 gives also many-sorted interpolation for first order logic, so it can be used to prove preservation results for first order logic as well.

Here is an example:

▶ **Theorem 23** ([41]). *A formula φ of $L_{\omega_1\omega}$ is preserved by submodels if and only if it is logically equivalent to a universal formula.*

Proof. Let us assume the single sorted φ is written in sort 0 variables and has just one binary predicate symbol R. Let φ' be the same written in sort 1 variables and with R replaced by R'. Let EXT be the conjunction

$$\forall x^1 \exists x^0 (x^0 = x^1) \wedge \forall x^1 \forall y^1 (R'(x^1, y^1) \leftrightarrow R(x^1, y^1)). \tag{1}$$

Note that $(\{M_0, M_1\}, R, R') \models \forall x^1 \exists x^0 (x^0 = x^1)$ iff $M_1 \subseteq M_0$. The sentence (1) is true in $(\{M_0, M_1\}, R, R')$ iff $(M_1, R') \subseteq (M_0, R)$. By assumption, $\mathrm{EXT} \wedge \neg\varphi' \models \neg\varphi$. Let $\theta \in L_{\omega_1\omega}$ be given by Theorem 22. Thus $\mathrm{EXT} \wedge \neg\varphi' \models \theta$ and $\theta \models \neg\varphi$. The only common sort is 0,

so θ has only sort 0 symbols. To see that θ is existential we note that if a sort 0 variable was universally quantified in θ, then it is universally quantified in EXT $\wedge \neg\varphi'$, but there is no universally quantified sort 0 variable in EXT $\wedge \neg\varphi'$. Thus θ is existential. Moreover, $\models \neg\varphi \leftrightarrow \theta$. ◀

The infinitary logic $L_{\omega_1\omega}$ can be extended by new logical operations preserving its "good" properties such as interpolation: Let us call any function $\mathcal{P}(\omega) \to 2$ a *propositional connective*. If P is a propositional connective, then $L_{\omega_1\omega}(P)$ is the extension of $L_{\omega_1\omega}$ by the connective

$$\mathfrak{M} \models_P C(\langle\varphi_i : i < \omega\rangle)(\vec{a}) \iff P(\{i : \mathfrak{M} \models_P \varphi_i(\vec{a})\}) = 1.$$

Harrington [26] proved that there are 2^{2^ω} propositional connectives P such that $L_{\omega_1\omega}(P)$ satisfies the Craig Interpolation Theorem as well as a form of the Barwise Compactness Theorem. Unfortunately Harrington's new propositional connectives are just abstract functions with no intuitive or natural meaning. Still their existence demonstrates that $L_{\omega_1\omega}$ is by no means maximal with respect to the Craig Interpolation Theorem, even if a weak form of compactness is added.

The situation changes radically when we move to $L_{\omega_2\omega}$:

▶ **Theorem 24** ([42]). Craig$(L_{\omega_2\omega}, L_{\infty\omega})$ *fails.*

Proof. Let $\varphi \in L_{\omega_2\omega}$ say "$<$ is a linear order of order-type ω_1" [47]. Let $\psi \in L_{\omega_1\omega}$ say "$<'$ is a total linear order of order-type ω". Clearly $\varphi \models \neg\psi$. Suppose $\theta \in L_{\infty\omega}$ is such that $\varphi \models \theta$, $\theta \models \neg\psi$ and the vocabulary of θ is $\emptyset$. Then $(\omega_1, <) \models \varphi$, so $(\omega_1) \models \theta$. But $(\omega_1) \equiv_{\infty\omega} (\omega)$, as an easy application of the Ehrenfeucht-Fraïssè game of length ω shows. So $(\omega) \models \theta$, a contradiction. ◀

The method of proof of Theorem 20, i.e. essentially an undefinability of truth argument, can be used to prove:

▶ **Theorem 25** ([25]). **1.** WBeth$(L_{\omega_2\omega}, L_{\omega_2\omega_2})$ *fails.*
2. WBeth$(L_{\omega_1\omega_1}, L_{\infty\infty})$ *fails.*

However, with an appropriate modification of the method of the proof of Theorem 22 one can show Craig$(L_{\kappa\omega}, L_{(2^{<\kappa})^+\kappa})$, where κ is regular ([42]). By modifying the question of interpolation suitably, one can obtain a more balanced result, see [5]. The result Craig$(L_{\kappa\omega}, L_{(2^{<\kappa})^+\kappa})$ raises the question, whether or not there is a logic L such that $L_{\kappa\omega} \leq L \leq L_{(2^{<\kappa})^+\kappa}$ and L has interpolation? Shelah's new infinitary logic L^1_κ gives one answer to this question ([51]).

To see this, let us redefine $L_{\kappa\omega}$ as $\bigcup_{\lambda<\kappa} L_{\lambda^+\omega}$ and $L_{\kappa\kappa}$ as $\bigcup_{\lambda<\kappa} L_{\lambda^+\lambda^+}$ in the case that κ is a limit cardinal. For regular limits this agrees with the old notation for $L_{\kappa\omega}$ and $L_{\kappa\kappa}$, but for singular cardinals the new notation seems more canonical. Let $\kappa = \beth_\kappa$, where $\beth_0 = \omega$, $\beth_{\alpha+1} = 2^{\beth_\alpha}$ and $\beth_\nu = \sup\{\beth_\alpha : \alpha < \nu\}$ for limit ordinals ν. The new infinitary logic L^1_κ introduced in [51] satisfies $L_{\kappa\omega} \leq L^1_\kappa$, $L^1_\kappa \leq L_{\kappa\kappa}$ and Craig(L^1_κ). Moreover, L^1_κ has a

Lindström style model theoretic characterization in terms of a strong form of undefinability of well-order.

The main ingredient of L^1_κ is a new variant $G^\beta_\theta(A, B)$, where β is an ordinal and θ a cardinal, of the Ehrenfeucht-Fraïssè game for $L_{\kappa\kappa}$. In this variant player II gives only partial answers to moves of player I. She makes promises and fulfills the promises move by move. If the game lasted for ω moves, the partial answers of II would constitute full answers. But the game has a well-founded clock, so II never ends up fully completing her answers. Although L^1_κ cannot express well-ordering, it can express e.g. the property of a linear order of not having an uncountable descending chain.

Shelah's game $G^\beta_\theta(A, B)$ proceeds as follows:

- At first player I picks $\beta_0 < \beta$ and $\vec{a^0}$ from A with $len(\vec{a^0}) \leq \theta$. The move β_0 is a "clock" move which controls the (finite) length of the game. The move $\vec{a^0}$ is player I's challenge to player II. He wants to know what are the images of the $\leq \theta$ elements of the sequence $\vec{a^0}$.
- Next II picks $f_0 : \vec{a^0} \to \omega$ and $g_0 : A \to B$ a partial isomorphism such that $f_0^{-1}(0) \subseteq \mathrm{dom}(g_0)$. Here player II responds to the challenge made by player I. But now comes the catch. Player II divides the set $\vec{a^0}$ into ω pieces with f_0. She is not going to respond yet to the entire challenge $\vec{a^0}$, only to a piece, namely $f_0^{-1}(0)$. To the other pieces she is going to respond later, one by one. With good luck the game ends before she has to respond to $f_0^{-1}(10^{10})$!
- Next I picks $\beta_1 < \beta_0$ and $\vec{b^1}$ from B such that $len(\vec{b^1}) \leq \theta$. Here I challenges II to give the preimages of the $\leq \theta$ elements of $\vec{b^1}$.
- Next II picks $f_1 : \vec{b^1} \to \omega$ and $g_1 : A \to B$ a partial isomorphism, $g_1 \supseteq g_0$ such that $f_0^{-1}(1) \subseteq \mathrm{dom}(g_1)$ and $f_1^{-1}(0) \subseteq \mathrm{ran}(g_1)$. With f_1 player II splits the challenge $\vec{b^1}$ into ω pieces to which she is going to respond one by one while the game continues. With g_1 she starts responding to the challenge $\vec{b^1}$. In order for her responses to remain consistent, it is necessary that $g_1 \supseteq g_0$. By making sure that $f_1^{-1}(0) \subseteq \mathrm{ran}(g_1)$ she responds to the challenge $f_1^{-1}(0)$. Now she has to also continue responding to the challenge $\vec{a^0}$ by making sure $f_0^{-1}(1) \subseteq \mathrm{dom}(g_1)$.
- And so on until $\beta_n = 0$.

Player II wins if she can play all her moves, otherwise Player I wins. $A \sim^\beta_\theta B$ if Player II has a winning strategy in the game. $A \equiv^\beta_\theta B$ is defined as the transitive closure of $A \sim^\beta_\theta B$. A union of $\leq \beth_{\beta+1}(\theta)$ equivalence classes of $\equiv^\beta_\theta$ for some $\theta < \kappa$ and $\beta < \theta^+$ is called a sentence of L^1_κ.

How does this definition make L^1_κ an abstract logic? One has to work a bit to prove that all the closure properties that we require of a logic are satisfied. In the end, L^1_κ is an abstract logic in the sense of Definition 1.1.1 of [2]. But what is the syntax like? All we know about the sentences of L^1_κ is that they are classes of models. Still we can prove:

▶ **Theorem 26** (Shelah [51]). *Suppose $\kappa = \beth_\kappa$. Then L^1_κ satisfies the Craig Interpolation Theorem.*

The proof is based entirely on properties of the game $G^{\beta}_{\theta}(A, B)$ and is not totally unlike the proof of Theorem 17. It is somewhat surprising that many model theoretic results can be proved for L^1_κ although we are not able (yet) to answer the question:

Problem: What is the syntax of Shelah's logic?

Cartagena logic [31] is a syntactic fragment of Shelah's logic. Its Δ-extension is Shelah's logic L^1_κ. For more on L^1_κ, see [56].

5 Higher Order Logics

Higher order logic are the oldest extensions of first order logic. In *weak second order logic* L^2_{w} there are variables for individuals, as in first order logic, but also variables for finite sets of individuals. If X is a set-variable and t is a term denoting an individual, then we can form the atomic formula $X(t)$. Quantifiers range over individual and over set variables. The logic L^2_{w} is quite close to the logic $L(Q_0)$ (see section 3), in fact, $\Delta(L^2_{\mathrm{w}}) \equiv \Delta(L(Q_0))$.

It was shown in [45] that the Beth property fails for *weak* second order logic, and for *monadic* second order logic in which second order variables range over arbitrary subsets of the domain but vocabularies are canbe non-monadic. The former claim is proved by means of an undefinability of truth argument, almost verbatim as we did in the proof of Theorem 20, while the latter claim is proved by reduction to the decidable monadic second order theory of the successor function ([8]). Namely, in the standard model of the successor function, which is definable in monadic second order logic, addition and multiplication are implicitly definable by their familiar recursive definitions. If they were explicitly definable, arithmetic would be reducible to the monadic second order theory of the successor function. However, this is impossible because the former is non-arithmetic but the latter is decidable by [8].

In stationary logic $L(aa)$, introduced in [48] and [3], we have variables for individuals and also for countable sets of individuals. If s is a set-variable and t is a term denoting an individual, then we can again form the atomic formula $s(t)$. This time we do not have existential and universal quantifiers for set-variables. Instead we have a quantifier which can say that "most" countable sets have some property. What does "most" mean when we talk about countable sets? Here we use the concept of a club set. A set C of countable subsets of M is *unbounded* if for every countable $X \subseteq M$ there is $Y \in C$ such that $X \subseteq Y$. The set C is *closed* if it is closed under unions of increasing ω-chains. A *club* set means a closed unbounded set. The club (countable) subsets of a set M form a (normal) filter[14], which however is not an ultrafilter. Still, it is a useful measure of largeness. We adopt the following generalized second order quantifier:

[14] For the definitions of filter and ultrafilter, see [29].

$$\mathfrak{M} \models aas\varphi(s,\vec{a}) \iff \{A \subseteq M \ : \ |A| \leq \omega, (\mathfrak{M}, A) \models \varphi(A,\vec{a})\} \text{ contains a club of countable subsets of } M.$$

As proved in [48] and [3], the logic $L(aa)$ is countably compact and has a nice complete axiomatization. Of course, it does not have the Löwenheim property, but every consistent sentence has a model of cardinality at most $\aleph_1$. It extends both $L(Q_1)$ and $L(Q^{\text{cof}}_\omega)$. The logic $L(aa)$ does not have Beth property [39], but:

▶ **Theorem 27** ([49]). $\text{Craig}(L(Q^{cof}_\omega), L(aa))$.

Proof. (A rough sketch) We actually prove $\text{Rob}(L(Q^{\text{cof}}_\omega), L(aa))$. This involves proving that if T_1 and T_2 are countable complete $L(aa)$-theories, $T_0 = T_1 \cap T_2$ is complete, and $\tau(T_1) \cap \tau(T_2) = \tau(T_0)$, then $(T_1 \cap L(Q^{\text{cof}}_\omega)) \cup (T_2 \cap L(Q^{\text{cof}}_\omega))$ is consistent. Theories T_1 and T_2 are first enriched by new unary predicates P_α, $\alpha < \omega_1$. We add the axiom $\varphi(P_{\alpha_1}, \ldots, P_{\alpha_n})$ to T'_l, whenever $\alpha_1 < \ldots < \alpha_n < \omega_1$ and $aas_1 \ldots aas_n\varphi(s_1, \ldots, s_n) \in T_l$. These predicates build a canonical club in our final model giving us control of $L(aa)$-truth. To see how this works in a simple case, consider the set of

$$\forall x_1 \ldots \forall x_n((P(x_1) \wedge \ldots \wedge P(x_{n+1})) \to \varphi^{(P)}(x_1, \ldots, x_n)), \tag{2}$$

where

$$aas\forall x_1 \ldots \forall x_n((s(x_1) \wedge \ldots \wedge s(x_{n+1})) \to \varphi^{(s)}(x_1, \ldots, x_n)) \in T_1 \tag{3}$$

and $\varphi(x_1, \ldots, x_n)$ is first order. Since every model has a closed unbounded set of countable subsets that are domains of elementary submodels with respect to first order logic,[15] and the club sets form a filter, the collection of all sentences (2) essentially "says" in a model $\mathfrak{M}$ that $P^{\mathfrak{M}}$ is the domain of a countable elementary submodel of $\mathfrak{M}$.

Next T'_1 and T'_2 are reduced to complete first order theories T^*_1, T^*_2 by introducing new relation symbols, namely an n-ary symbol R_φ for each formula $\varphi(x_1, \ldots, x_n)$ of $L(aa)$, with the defining axiom

$$\forall x_1 \ldots x_n(R_\varphi(x_1, \ldots, x_n) \leftrightarrow \varphi(x_1, \ldots, x_n)),$$

and taking only the resulting first order formulas.

Then we use the Robinson property of first order logic to obtain a model $\mathfrak{M}$ of $T^*_1 \cup T^*_2$. W.l.o.g. $\mathfrak{M}$ is $\aleph_2$-saturated (see e.g. [27, p. 480] for the definition and basic properties of $\aleph_2$-saturation). Let $\mathfrak{N}$ be the union of the submodels of $\mathfrak{M}$ determined by the predicates P_α. Now we can show that if $\mathfrak{N}$ thinks the cofinality of a definable linear order is uncountable, it is in $\mathfrak{N}$ exactly $\aleph_1$. On the other hand, if $\mathfrak{N}$ thinks the cofinality of a definable linear order is ω,

[15] One first forms Skolem-functions and then takes submodels that are closed under the Skolem-functions. The collections of such submodels is always closed unbounded.

the real cofinality of the linear order in $\mathfrak{N}$ is actually $\aleph_2$. The final step is a chain argument, using a chain of length ω, to turn cofinality $\aleph_2$ to cofinality $\aleph_0$ without losing the cofinalities that are $\aleph_1$. ◀

Why the above result is remarkable is that it is the closest we have got so far in obtaining interpolation among countably compact proper extensions of $L_{\omega\omega}$. The situation raises the following intriguing question:

Problem: Is there a logic L such that $L(Q_\omega^{\text{cof}}) \leq L \leq L(aa)$ and L has the Craig interpolation property?

Note that if a logic L as in the above problem exists, then L is also countably compact.

In full single-sorted second order logic L^2, with variables for arbitrary relations, interpolation (even uniform[16]) holds trivially, because we can quantify over relations. The separation property, in particular, becomes trivially true as single-sorted $\text{PC}(L^2)$-classes are always $\text{EC}(L^2)$-classes. In particular, $\Delta_1^1(L^2) \equiv L^2$. In many-sorted second order logic interpolation (even the weak Beth property) fails because we can use new sorts to define truth implicitly, but not explicitly ([12], [25]), imitating the proof of Theorem 20. The difference between the single-sorted and the many-sorted case is that in the latter the predicates that one would like to quantify away may involve new sorts, i.e. elements outside the current model. There is no way L^2 can reach out to them. In particular, $\Delta(L^2) \not\equiv L^2$.

Problem: Is there any (set-theoretically definable) extension of second order logic that has many-sorted interpolation?

If we drop the requirement of set-theoretical definability, there is a solution: sort logic ([55]). If we just want a proper extension of first order logic with many-sorted interpolation, the logic $L_{\omega_1\omega}$ offers a solution, as we have seen.

Existential second order logic ESO (i.e. single-sorted $\text{PC}(L_{\omega\omega})$) is a perfectly nice abstract logic, only it is not closed under negation. It satisfies (even uniform) single-sorted Craig Interpolation Theorem, as we can quantify over relations, i.e. in the single-sorted context $\text{PC}(\text{ESO}) \equiv \text{ESO}$. For the same reason ESO satisfies the Souslin-Kleene Interpolation and Beth Theorems. In each case the claim is trivially true. The usual proof of the Robinson property works also for ESO. The logic ESO satisfies also compactness and has the Löwenheim property. This does not violate Lindström's Theorem, because ESO is not closed under negation. In [52] it is proved that there is no strongest abstract logic without negation which is compact and satisfies the Löwenheim property. Many-sorted $\text{PC}(L_{\omega\omega})$ is actually equivalent to single-sorted $\text{PC}(L_{\omega\omega})$ ([36]). Thus also many-sorted ESO satisfies interpolation and the related properties following from interpolation.

Dependence logic $\mathcal{D}$, based on the atom $=\!(\vec{x}, \vec{y})$, which says that $\vec{x}$ totally determines $\vec{y}$, was introduced in [53]. On the level of sentences $\mathcal{D}$ is equivalent to ESO, hence it satisfies

[16] See Chapter 2

(even uniform) Craig Interpolation, Souslin-Kleene Interpolation, Beth theorem, and the Robinson property, whether single- or many-sorted. The Craig Interpolation property holds for $\mathcal{D}$ formulas also in the following form: Suppose $\varphi(\vec{x},\vec{y}) \models \psi(\vec{x},\vec{z})$, where $\vec{y} \cap \vec{z} = \emptyset$. Then there is $\theta(\vec{x})$ such that $\varphi(\vec{x},\vec{y}) \models \theta(\vec{x})$ and $\theta(\vec{x}) \models \psi(\vec{x},\vec{z})$. Namely, we can take $\theta(\vec{x})$ to be $\exists\vec{y}\varphi(\vec{x},\vec{y})$. This works, because of locality[17] ([53, Lemma 3.27]).

Inclusion Logic $\mathcal{I}$ ([22]), based on atoms of the form $\vec{x} \subseteq \vec{y}$, which say that every value of $\vec{x}$ occurs as a value of $\vec{y}$, is not equal to ESO on the level of sentences. Rather, inclusion logic is equivalent to Positive Greatest Fixed Point Logic GFP^+, i.e. the fragment of Greatest Fixed Point Logic in which fixed point operators occur only positively ([23]). Thus we cannot use reduction to ESO to infer any interpolation properties for inclusion logic.

Problem: Does Inclusion Logic $\mathcal{I}$ ([22]) satisfy the Craig Interpolation property?

6 Conclusion

Interpolation in all its forms mostly fails in extensions of first order logic. It seems difficult to extend first order logic in a way which leads to the kind of balance required by interpolation and the Beth property. There is the artificial way of using the Δ-operation to obtain $\Delta(L)$, which always has the Souslin-Kleene Interpolation property and therefore the weak Beth property. But there is no general method to find a syntax for the semantically defined $\Delta(L)$. The infinitary logic $L_{\omega_1\omega}$ is remarkable in the richness of its model theory and, as we have seen, it has the interpolation property. In bigger infinitary logics interpolation systematically fails, signalling, perhaps, that there is something incomplete in their syntax. The new large infinitary logic L^1_κ enjoys interpolation, but lacks so far a satisfactory syntax. Overall, the area abounds with open problems, only some of which have been mentioned above.

Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 101020762), as well as from the Research Council of Finland (grants number 322795 and 368671).

References

1 Jon Barwise. *Admissible sets and structures.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin-New York, 1975. An approach to definability theory.

[17] Locality of a formula means that the truth of the formula depends only on values of assignments on variables which are free in the formula.

2 Jon Barwise and Solomon Feferman, editors. *Model-theoretic logics*. Perspectives in Mathematical Logic. Springer-Verlag, New York, 1985.

3 Jon Barwise, Matt Kaufmann, and Michael Makkai. Stationary logic. *Ann. Math. Logic*, 13(2):171–224, 1978. doi:10.1016/0003-4843(78)90003-7.

4 Jon Barwise and John Schlipf. An introduction to recursively saturated and resplendent models. *J. Symbolic Logic*, 41(2):531–536, 1976. doi:10.2307/2272253.

5 Jon Barwise and Johan van Benthem. Interpolation, preservation, and pebble games. *J. Symbolic Logic*, 64(2):881–903, 1999. doi:10.2307/2586507.

6 Johan van Benthem, Balder Ten Cate, and Jouko Väänänen. Lindström theorems for fragments of first-order logic. *Log. Methods Comput. Sci.*, 5(3):3:3, 27, 2009. doi:10.2168/LMCS-5(3:3)2009.

7 Evert W. Beth. On Padoa's method in the theory of definition. *Indag. Math.*, 15:330–339, 1953. Nederl. Akad. Wetensch. Proc. Ser. A **56**.

8 J. Richard Büchi. On a decision method in restricted second order arithmetic. In *Logic, Methodology and Philosophy of Science (Proc. 1960 Internat. Congr.)*, pages 1–11. Stanford Univ. Press, Stanford, CA, 1962.

9 Chen Chung Chang and H. Jerome Keisler. *Model theory*, volume 73 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, third edition, 1990.

10 William Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *J. Symbolic Logic*, 22:250–268, 1957. doi:10.2307/2963593.

11 William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. Symbolic Logic*, 22:269–285, 1957. doi:10.2307/2963594.

12 William Craig. Satisfaction for n-th order languages defined in n-th order languages. *J. Symbolic Logic*, 30:13–25, 1965. doi:10.2307/2270577.

13 William Craig. The road to two theorems of logic. *Synthese*, 164(3):333–339, 2008. doi:10.1007/s11229-008-9353-3.

14 Max A. Dickmann. *Large infinitary languages – Model Theory*, volume 83 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam-Oxford; American Elsevier Publishing Co., Inc., New York, 1975.

15 Mirna Džamonja and Jouko Väänänen. Chain logic and Shelah's infinitary logic. *Israel J. Math.*, 245(1):93–134, 2021. doi:10.1007/s11856-021-2207-0.

16 Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fund. Math.*, 49:129–141, 1960/61. doi:10.4064/fm-49-2-129-141.

17 Solomon Feferman. Lectures on proof theory. In *Proceedings of the Summer School in Logic (Leeds, 1967)*, volume No. 70 of *Lecture Notes in Math.*, pages 1–107. Springer, Berlin-New York, 1968.

18 Jörg Flum. Characterizing logics. In Barwise and Feferman [2], pages 77–120.

19 Jörg Flum and Martin Ziegler. *Topological model theory*, volume 769 of *Lecture Notes in Mathematics*. Springer, Berlin, 1980.

20 Roland Fraïssé. Sur quelques classifications des relations, basées sur des isomorphismes restreints. *Publ. Sci. Univ. Alger. Sér. A*, 2:15–60, 1955.

21 Harvey Friedman. Beth's theorem in cardinality logics. *Israel J. Math.*, 14:205–212, 1973. doi:10.1007/BF02762675.

22 Pietro Galliani. Inclusion and exclusion dependencies in team semantics—on some logics of imperfect information. *Ann. Pure Appl. Logic*, 163(1):68–84, 2012. doi:10.1016/j.apal.2011.08.005.

23 Pietro Galliani and Lauri Hella. Inclusion logic and fixed point logic. In *Computer science logic 2013*, volume 23 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 281–295. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2013.

24 Marta García-Matos and Jouko Väänänen. Abstract model theory as a framework for universal logic. In *Logica universalis*, pages 19–33. Birkhäuser, Basel, 2005. doi:10.1007/3-7643-7304-0_2.

25 Richard Gostanian and Karel Hrbáček. On the failure of the weak Beth property. *Proc. Amer. Math. Soc.*, 58:245–249, 1976. doi:10.2307/2041394.

26 Leo Harrington. Extensions of countable infinitary logic which preserve most of its nice properties. *Arch. Math. Logik Grundlag.*, 20(3-4):95–102, 1980. doi:10.1007/BF02021129.

27 Wilfrid Hodges. *Model theory*, volume 42 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1993. doi:10.1017/CBO9780511551574.

28 John E. Hutchinson. Model theory via set theory. *Israel J. Math.*, 24(3-4):286–304, 1976. doi:10.1007/BF02834760.

29 Thomas Jech. *Set theory*. Springer Monographs in Mathematics. Springer-Verlag, Berlin, millennium edition, 2003.

30 H. Jerome Keisler. *Model theory for infinitary logic. Logic with countable conjunctions and finite quantifiers*, volume 62 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam-London, 1971.

31 Siiri Kivimäki, Jouko Väänänen, and Andrés Villaveces. Cartagena logic. *CoRR*, abs/2108.13495, 2024. doi:10.48550/arXiv.2108.13495.

32 Per Lindström. First order predicate logic with generalized quantifiers. *Theoria*, 32:186–195, 1966. doi:10.1111/j.1755-2567.1966.tb00600.x.

33 Per Lindström. On extensions of elementary logic. *Theoria*, 35:1–11, 1969. doi:10.1111/j.1755-2567.1969.tb00356.x.

34 Per Lindström. Prologue. In *Quantifiers: logics, models and computation. Vol. 1. Surveys*, volume 248 of *Synthese Lib.*, pages 21–24. Kluwer Acad. Publ., Dordrecht, 1995.

35 Menachem Magidor and Jerome Malitz. Compact extensions of $L(Q)$. Ia. *Ann. Math. Logic*, 11(2):217–261, 1977. doi:10.1016/0003-4843(77)90019-5.

36 Mihály Makkai. On PC_Δ-classes in the theory of models. *Publ. Math. Inst. Hung. Acad. Sci., Ser. A*, 9:159–194, 1964.

37 J. A. Makowsky and Daniele Mundici. Abstract equivalence relations. In Barwise and Feferman [2], pages 717–746.

38 J. A. Makowsky and Saharon Shelah. The theorems of Beth and Craig in abstract model theory. I. The abstract setting. *Trans. Amer. Math. Soc.*, 256:215–239, 1979. doi:10.2307/1998109.

39 J. A. Makowsky and Saharon Shelah. The theorems of Beth and Craig in abstract model theory. II. Compact logics. *Arch. Math. Logik Grundlag.*, 21(1-2):13–35, 1981. doi:10.1007/BF02011631.

40 J. A. Makowsky, Saharon Shelah, and Jonathan Stavi. Δ-logics and generalized quantifiers. *Ann. Math. Logic*, 10(2):155–192, 1976. doi:10.1016/0003-4843(76)90021-8.

41 Jerome Malitz. Universal classes in infinitary languages. *Duke Math. J.*, 36:621–630, 1969. URL: http://projecteuclid.org/euclid.dmj/1077378481.

42 Jerome Malitz. Infinitary analogs of theorems from first order model theory. *J. Symbolic Logic*, 36:216–228, 1971. doi:10.2307/2270256.

43 Alan H. Mekler and Saharon Shelah. Stationary logic and its friends. I. *Notre Dame J. Formal Logic*, 26(2):129–138, 1985. doi:10.1305/ndjfl/1093870821.

44 Andrzej Mostowski. On a generalization of quantifiers. *Fund. Math.*, 44:12–36, 1957. doi:10.4064/fm-44-1-12-36.

45 Andrzej Mostowski. Craig's interpolation theorem in some extended systems of logic. In *Logic, Methodology and Philos. Sci. III (Proc. Third Internat. Congr., Amsterdam, 1967)*, pages 87–103. North-Holland, Amsterdam, 1968.

46 Abraham Robinson. A result on consistency and its application to the theory of definition. *Indag. Math.*, 18:47–58, 1956. Nederl. Akad. Wetensch. Proc. Ser. A **59**.

47 Dana Scott. Logic with denumerably long formulas and finite strings of quantifiers. In *Theory of Models (Proc. 1963 Internat. Sympos. Berkeley)*, pages 329–341. North-Holland, Amsterdam, 1965.

48 Saharon Shelah. Generalized quantifiers and compact logic. *Trans. Amer. Math. Soc.*, 204:342–364, 1975. doi:10.2307/1997362.

49 Saharon Shelah. Remarks in abstract model theory. *Ann. Pure Appl. Logic*, 29(3):255–288, 1985. doi:10.1016/0168-0072(85)90002-8.

50 Saharon Shelah. The theorems of Beth and Craig in abstract model theory. III. Δ-logics and infinitary logics. *Israel J. Math.*, 69(2):193–213, 1990. doi:10.1007/BF02937304.

51 Saharon Shelah. Nice infinitary logics. *J. Amer. Math. Soc.*, 25(2):395–427, 2012. doi:10.1090/S0894-0347-2011-00712-1.

52 Saharon Shelah and Jouko Väänänen. Positive logics. *Arch. Math. Logic*, 62(1-2):207–223, 2023. doi:10.1007/s00153-022-00837-3.

53 Jouko Väänänen. *Dependence logic*, volume 70 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 2007. A new approach to independence friendly logic. doi:10.1017/CBO9780511611193.

54 Jouko Väänänen. *Models and games*, volume 132 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2011. doi:10.1017/CBO9780511974885.

55 Jouko Väänänen. Sort logic and foundations of mathematics. In *Infinity and truth*, volume 25 of *Lect. Notes Ser. Inst. Math. Sci. Natl. Univ. Singap.*, pages 171–186. World Sci. Publ., Hackensack, NJ, 2014. doi:10.1142/9789814571043_0005.

56 Jouko Väänänen and Boban Velickovic. On some infinitary logics. *CoRR*, abs/2402.13344, 2024. doi:10.48550/arXiv.2402.13344.

Chapter 8

Interpolation and Amalgamation

George Metcalfe
Mathematical Institute, University of Bern, Switzerland

Abstract

This chapter presents a state-of-the-art survey of relationships, traditionally referred to as 'bridges', between interpolation properties for propositional logics — including superintuitionistic, modal, and substructural logics — and amalgamation properties for corresponding classes of algebraic structures. These bridges are developed in the framework of universal algebra and illustrated with a broad range of examples from logic and algebra, demonstrating their use in establishing properties for both fields.

Contents

1 Introduction **273**
2 Logic and Algebra **275**
3 Consequence and Congruence **279**
4 Local Deduction Theorems and the Congruence Extension Property . . **281**
5 Amalgamation and the Robinson Property **286**
6 Deductive Interpolation Properties **290**
7 The Craig Interpolation Property **293**
Acknowledgments **296**
References **296**

1 Introduction

The aim of this chapter is to establish and explore some remarkable relationships or 'bridges' existing between various forms of interpolation for propositional logics and amalgamation for classes of algebraic structures. Such bridges have appeared regularly in the literature, both for specific families of logics — notably, superintuitionistic, modal, and substructural logics (see, e.g., [54, 28, 29, 45]) — and within the broader settings of abstract algebraic logic [51, 11] and model theory [2]. Here we follow [66, 57] in constructing these bridges in the framework



© George Metcalfe;
licensed under Creative Commons License CC BY 4.0
In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 8, pages 273–299. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.h

of universal algebra, which supplies the tools required for a uniform presentation, while still covering the vast majority of propositional logics. To keep our presentation self-contained, we introduce the requisite elementary notions from universal algebra along the way, giving proofs of key theorems and providing pointers to the relevant literature where appropriate. Throughout, we incorporate a range of examples from logic and algebra that illustrate the usefulness of the 'bridges' in both directions, and give references to more recent developments.

To get our bearings, let us consider first the historically pivotal case of intuitionistic propositional logic IPC, understood as a consequence relation $\vdash_{\mathrm{IPC}}$, defined syntactically by an axiom system or sequent calculus, or semantically via Heyting algebras or Kripke models. Maksimova proved in 1977 that precisely eight superintuitionistic logics (i.e., axiomatic extensions of IPC) have the Craig interpolation property (CIP) or equivalently, in this setting, the deductive interpolation property (DIP) [52]. That is, for each such logic L and formulas α and β satisfying $\alpha \vdash_{\mathrm{L}} \beta$, there exists a formula γ satisfying $\alpha \vdash_{\mathrm{L}} \gamma$ and $\gamma \vdash_{\mathrm{L}} \beta$, whose variables occur in both α and β. Maksimova's proof made use of Kripke semantics for superintuitionistic logics and was essentially algebraic. First, she proved that a superintuitionistic logic has the DIP if, and only if, an associated variety (equational class) of Heyting algebras has the amalgamation property (AP), and, second, that there are precisely eight such varieties (see Chapter 3 for further details). This result was later strengthened by Ghilardi and Zawadowski [30], who, building on Pitts' theorem for IPC [68], proved that these eight logics have the stronger property of uniform interpolation, and that the first-order theories of the eight varieties of Heyting algebras with the AP have a model completion (see Chapter 9).

Similar results have been established for families of normal modal logics (see, e.g., [8, 28]). In particular, Maksimova used appropriate amalgamation properties to prove that between forty-three and forty-nine axiomatic extensions of S4 have the DIP, and between thirty-one and thirty-seven such extensions have the CIP [53, 54], whereas continuum-many axiomatic extensions of Gödel-Löb logic GL have both properties [54]. Interpolation and amalgamation properties (and their failures) have also been established for diverse families of substructural logics and varieties of residuated algebras, respectively (see, e.g., [60, 29, 45, 55, 57, 32, 58, 37]). For example, precisely nine varieties of Sugihara monoids have the AP, and hence precisely nine axiomatic extensions of the relevant logic $\mathrm{RM_t}$ (R-Mingle with unit) have the DIP or equivalently, in this case, the CIP [56]. Moreover, a variety of MV-algebras has the AP if, and only if, it is generated by a totally ordered MV-algebra, so countably infinitely many axiomatic extensions of Łukasiewicz logic Ł have the DIP [16], despite the fact that classical propositional logic CPC is the only consistent axiomatic extension of Ł that has the CIP. Conversely, syntactic proofs of the CIP for substructural logics — in particular, extensions of the Full Lambek Calculus with exchange — have been used to establish the AP for varieties of residuated algebras where no algebraic proof is known (see, e.g., [29, 58] for details).

Amalgamation emerged as a central concept in Schreier's work on amalgamated free products of groups in the 1920s [69]. The AP was formulated in full generality by Fraïssé in his 1954 paper [21] and subsequently studied intensively by Jónsson in the setting of universal

algebra [38, 39, 40, 41, 42]. The relationship between amalgamation and interpolation was first considered by Daigneault in the context of polyadic algebras [12], and then, in a more general algebraic setting, by Jónsson [42], but is credited in the latter to unpublished work of Keisler, and, as observed by Pigozzi in [66], essential ideas underlying the proof may be traced back to Magnus' work in group theory. The basis for this relationship for a variety of algebras (or corresponding logic) was identified explicitly in [66]: equational consequence in the variety can be interpreted in terms of congruences of its free algebras. This interpretation provides the basis for a wealth of other bridge theorems between 'algebraic' and 'logical' properties studied in the literature (see, e.g., [48, 11, 57, 23] for further examples and references). In particular, such a bridge theorem has been established between the Beth definability property for a propositional logic — closely related to the CIP in some contexts (see Chapter 1) — and surjectivity of epimorphisms in the corresponding variety [62, 4].

Let us provide a brief overview of the contents of this chapter. In Section 2, we recall some basic notions of universal algebra and define equational consequence for a class of algebras, explaining in Section 3 how this definition can be recast for a variety in terms of congruences of its free algebras. In Section 4, we establish a bridge between the congruence extension property for a variety and the existence of an equational 'local deduction theorem' known as the extension property, describing also how a distinguished subclass may suffice for checking these properties. Similarly, in Section 5, we establish a bridge between the AP for a variety (or, in some cases, a distinguished subclass) and a property of equational consequence known as the Robinson property. In Section 6, we use these bridge theorems to relate algebraic properties to (variants of) the DIP. Finally, in Section 7, we turn our attention to the CIP and corresponding superamalgamation property for varieties of algebras with a lattice reduct.

2 Logic and Algebra

In this section, we recall some basic tools of universal algebra (referring the reader to [6] for further details) and introduce the key notion of equational consequence for a class of algebras. We also explain, pointing to some familiar examples, how equational consequence provides an appropriate setting for relating propositional logics to a suitable 'algebraic semantics'.

Let us begin by fixing an algebraic language $\mathcal{L}$ — that is, a first-order language with no relation symbols — and let $\mathcal{L}_n$ denote the set of operation (function) symbols of arity $n \in \mathbb{N}$. An *$\mathcal{L}$-algebra* $\mathbf{A}$ consists of a non-empty set A equipped with an operation $f^{\mathbf{A}} \colon A^n \to A$ for each $f \in \mathcal{L}_n$, typically written as $\langle A, f_1^{\mathbf{A}}, \dots, f_k^{\mathbf{A}} \rangle$ when $\mathcal{L}$ has operation symbols $f_1, \dots, f_k$. An $\mathcal{L}$-algebra $\mathbf{B}$ is a *subalgebra* of an $\mathcal{L}$-algebra $\mathbf{A}$ if $B \subseteq A$ and $f^{\mathbf{B}}(b_1, \dots, b_n) = f^{\mathbf{A}}(b_1, \dots, b_n)$ for all $f \in \mathcal{L}_n$ and $b_1, \dots, b_n \in B$. The *direct product* $\mathbf{A} = \prod_{i \in I} \mathbf{A}_i$ of a family $\{\mathbf{A}_i\}_{i \in I}$ of $\mathcal{L}$-algebras consists of the Cartesian product $\prod_{i \in I} A_i$ equipped with an operation $f^{\mathbf{A}}$ for each $f \in \mathcal{L}_n$ satisfying $f^{\mathbf{A}}(a_1, \dots, a_n)(i) = f^{\mathbf{A}_i}(a_1(i), \dots, a_n(i))$ for all $i \in I$.

Below we recall some classes of algebras that play a prominent role in the study of propositional logics, omitting sub- and superscripts when these are clear from the context.

▶ **Example 1.** A *lattice* is defined order-theoretically as a partially ordered set $\langle L, \leq \rangle$ such that any two elements $a, b \in L$ have a greatest lower bound $a \wedge b$ and least upper bound $a \vee b$. It is said to be *bounded* if it has a greatest element $\top$ and least element $\bot$, and *complete* if every subset $A \subseteq L$ has a greatest lower bound $\bigwedge A$ and least upper bound $\bigvee A$. A lattice may also be defined algebraically, however, in a language with binary operation symbols $\wedge$ and $\vee$ as an algebra $\mathbf{L} = \langle L, \wedge, \vee \rangle$ such that defining $a \leq b :\iff a \wedge b = a$ yields a lattice $\langle L, \leq \rangle$ in the order-theoretic sense. If $\langle L, \leq \rangle$ is bounded, then constant symbols $\bot, \top$ can be added to the language to obtain an algebra $\langle L, \wedge, \vee, \bot, \top \rangle$. A (bounded) lattice $\mathbf{L}$ is called *distributive* if $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ and $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ for all $a, b, c \in L$. ⌟

▶ **Example 2.** A *Boolean algebra*, defined in the language of bounded lattices extended with a unary operation symbol $\neg$, is an algebra $\mathbf{B} = \langle B, \wedge, \vee, \neg, \bot, \top \rangle$ such that $\langle B, \wedge, \vee, \bot, \top \rangle$ is a distributive bounded lattice and $\neg a$ is the (necessarily unique) complement of $a \in B$, i.e., $a \wedge \neg a = \bot$ and $a \vee \neg a = \top$. A *modal algebra* is defined in this language extended with a unary operation symbol $\Box$ as an algebra $\langle M, \wedge, \vee, \neg, \bot, \top, \Box \rangle$ such that $\langle M, \wedge, \vee, \neg, \bot, \top \rangle$ is a Boolean algebra, $\Box \top = \top$, and $\Box(a \wedge b) = \Box a \wedge \Box b$ for all $a, b \in M$. ⌟

▶ **Example 3.** A *Heyting algebra*, defined in the language of bounded lattices extended with a binary operation symbol $\to$, is an algebra $\mathbf{H} = \langle H, \wedge, \vee, \to, \bot, \top \rangle$ such that $\langle H, \wedge, \vee, \bot, \top \rangle$ is a bounded distributive lattice and $\to$ is the residual of $\wedge$, i.e., $a \wedge b \leq c \iff a \leq b \to c$ for all $a, b, c \in H$. Boolean algebras are term-equivalent to Heyting algebras satisfying $\neg\neg a = a$ for all $a \in H$, where $\neg a \coloneqq a \to \bot$; that is, $\langle H, \wedge, \vee, \neg, \bot, \top \rangle$ is a Boolean algebra and, conversely, for any Boolean algebra $\mathbf{B}$, defining $a \to b \coloneqq \neg a \vee b$ yields a Heyting algebra $\langle B, \wedge, \vee, \to, \bot, \top \rangle$ satisfying $\neg\neg a = a$ for all $a \in B$. Heyting algebras and Boolean algebras provide algebraic semantics for intuitionistic propositional logic IPC and classical propositional logic CPC, respectively, and other classes of Heyting algebras play this role for superintuitionistic logics. In particular, Heyting algebras satisfying $(a \to b) \vee (b \to a) = \top$ for all $a, b \in H$ provide algebraic semantics for Gödel logic G and are called *Gödel algebras* (see, e.g., [8]). ⌟

▶ **Example 4.** An *FL-algebra* (or *pointed residuated lattice*), defined in a language with binary operation symbols $\wedge$, $\vee$, $\cdot$, $\backslash$, $/$, and constant symbols f, e, is an algebra $\mathbf{L} = \langle L, \wedge, \vee, \cdot, \backslash, /, \mathrm{f}, \mathrm{e} \rangle$ such that $\langle L, \wedge, \vee \rangle$ is a lattice, $\langle L, \cdot, \mathrm{e} \rangle$ is a monoid, and $\backslash$ and $/$ are residuals of $\cdot$, i.e., $b \leq a\backslash c \iff ab \leq c \iff a \leq c/b$ for all $a, b, c \in L$. If $ab = ba$ for all $a, b \in L$, then $\mathbf{L}$ is called an *FL_e-algebra* and we define $a \to b \coloneqq a\backslash b = b/a$. FL-algebras and $\mathrm{FL_e}$-algebras provide algebraic semantics for the full Lambek calculus FL and full Lambek calculus with exchange $\mathrm{FL_e}$, respectively, and their subclasses play this role for a wide range of substructural logics (see, e.g., [29, 58]). For example, Heyting algebras are term-equivalent to FL-algebras satisfying $a \wedge b = ab$ and $\mathrm{f} \leq a$ for all $a, b \in L$, and *MV-algebras* — algebraic semantics for Łukasiewicz logic Ł — are term-equivalent to $\mathrm{FL_e}$-algebras satisfying $a \vee b = (a \to b) \to b$ and $\mathrm{f} \leq a$ for all $a, b \in L$. Let us note also that FL-algebras satisfying $\mathrm{e} = \mathrm{f}$ and $a(a\backslash \mathrm{e}) = \mathrm{e}$ for all $a \in L$ are term-equivalent to *ℓ-groups*, algebras with their own extensive literature (see, e.g., [1]) that play a key role in the structure theory of residuated algebras (see [58]). ⌟

An important ingredient of the interplay between logic and algebra is the notion of a formula algebra over a set of variables. Let us denote arbitrary sets of variables by $\overline{x}, \overline{y}, \overline{z}$, assuming without further comment that these are disjoint, and denote unions by writing $\overline{x}, \overline{y}$. We also assume for a more streamlined presentation that $\mathcal{L}$ has at least one constant symbol.[1] The *$\mathcal{L}$-formula algebra* $\mathbf{Fm}_{\mathcal{L}}(\overline{x})$ *over* $\overline{x}$ (often referred to as a *term algebra*) consists of the set $\mathrm{Fm}_{\mathcal{L}}(\overline{x})$ of *$\mathcal{L}$-formulas over* $\overline{x}$, built inductively using $\overline{x}$ and the operation symbols of $\mathcal{L}$, with an operation $f^{\mathbf{Fm}_{\mathcal{L}}(\overline{x})}$ for each $f \in \mathcal{L}_n$ that maps $\alpha_1, \ldots, \alpha_n \in \mathrm{Fm}_{\mathcal{L}}(\overline{x})$ to $f(\alpha_1, \ldots, \alpha_n)$. Note that since, by assumption, $\mathcal{L}$ has at least one constant symbol, $\mathbf{Fm}_{\mathcal{L}}(\emptyset)$ always exists.

Although consequence in a propositional logic is typically defined over formulas, it is convenient when relating logic to algebra to also consider consequences between equations. Formally, an *$\mathcal{L}$-equation over* $\overline{x}$ is an ordered pair of $\mathcal{L}$-formulas $\alpha, \beta \in \mathrm{Fm}_{\mathcal{L}}(\overline{x})$, denoted by $\alpha \approx \beta$, and the set of $\mathcal{L}$-equations over $\overline{x}$ is identified with the set $\mathrm{Eq}_{\mathcal{L}}(\overline{x}) \coloneqq (\mathrm{Fm}_{\mathcal{L}}(\overline{x}))^2$. When the language is clear from the context, we drop the subscript $\mathcal{L}$.

A map $\varphi\colon A \to B$ between $\mathcal{L}$-algebras $\mathbf{A}$ and $\mathbf{B}$ is a *homomorphism*, denoted by writing $\varphi\colon \mathbf{A} \to \mathbf{B}$, if for each $f \in \mathcal{L}_n$ and all $a_1, \ldots, a_n \in A$,

$$\varphi(f^{\mathbf{A}}(a_1, \ldots, a_n)) = f^{\mathbf{B}}(\varphi(a_1), \ldots, \varphi(a_n)),$$

and its *kernel* is defined as $\ker(\varphi) \coloneqq \{\langle a_1, a_2\rangle \in A^2 \mid \varphi(a_1) = \varphi(a_2)\}$.

An injective homomorphism $\varphi\colon \mathbf{A} \to \mathbf{B}$ is called an *embedding* of $\mathbf{A}$ into $\mathbf{B}$, and if φ is bijective, it is called an *isomorphism* between $\mathbf{A}$ and $\mathbf{B}$. If there exists a surjective homomorphism from $\mathbf{A}$ to $\mathbf{B}$, then $\mathbf{B}$ is said to be a *homomorphic image* of $\mathbf{A}$, and if there exists an isomorphism between $\mathbf{A}$ and $\mathbf{B}$, then $\mathbf{A}$ is said to be *isomorphic* to $\mathbf{B}$, denoted by writing $\mathbf{A} \cong \mathbf{B}$. For convenience, we also extend a homomorphism $\varphi\colon \mathbf{A} \to \mathbf{B}$ to the homomorphism $\varphi\colon \mathbf{A} \times \mathbf{A} \to \mathbf{B} \times \mathbf{B}$; $\langle a_1, a_2\rangle \mapsto \langle \varphi(a_1), \varphi(a_2)\rangle$.

We now have the ingredients required to define equational consequence with respect to some given class $\mathcal{K}$ of $\mathcal{L}$-algebras. For any set $\overline{x}$ and $\Sigma \cup \{\varepsilon\} \subseteq \mathrm{Eq}(\overline{x})$, let

$$\begin{aligned}\Sigma \vdash_{\mathcal{K}} \varepsilon \;:\Longleftrightarrow\; & \text{for any } \mathbf{A} \in \mathcal{K} \text{ and homomorphism } \varphi\colon \mathbf{Fm}(\overline{x}) \to \mathbf{A},\\ & \Sigma \subseteq \ker(\varphi) \implies \varepsilon \in \ker(\varphi),\end{aligned}$$

and for any $\Sigma \cup \Delta \subseteq \mathrm{Eq}(\overline{x})$,

$$\Sigma \vdash_{\mathcal{K}} \Delta :\Longleftrightarrow \Sigma \vdash_{\mathcal{K}} \varepsilon \text{ for each } \varepsilon \in \Delta.$$[2]

---

[1] All definitions and theorems presented here are easily adjusted to accommodate languages with no constant symbols by adding assumptions that certain sets or their intersections are non-empty.

[2] To confirm that $\Sigma \vdash_{\mathcal{K}} \varepsilon$ is well-defined, we should check that the defining condition is independent of the set $\overline{x}$ for which $\Sigma \cup \{\varepsilon\} \subseteq \mathrm{Eq}(\overline{x})$. To this end, it suffices to observe that every homomorphism $\varphi\colon \mathbf{Fm}(\overline{x}) \to \mathbf{A}$ extends to a homomorphism $\hat{\varphi}\colon \mathbf{Fm}(\overline{x}, \overline{y}) \to \mathbf{A}$ with $\ker(\hat{\varphi}) \cap \mathrm{Fm}(\overline{x})^2 = \ker(\varphi)$, and every homomorphism $\psi\colon \mathbf{Fm}(\overline{x}, \overline{y}) \to \mathbf{A}$ restricts to a homomorphism $\psi'\colon \mathbf{Fm}(\overline{x}) \to \mathbf{A}$ with $\ker(\psi) \cap \mathrm{Fm}(\overline{x})^2 = \ker(\psi')$.

It is easily checked that this notion of consequence, when restricted to a fixed set $\overline{x}$, yields an (abstract) consequence relation over $\mathrm{Eq}(\overline{x})$; that is, for any $\Sigma \cup \Pi \cup \{\varepsilon\} \subseteq \mathrm{Eq}(\overline{x})$,

- if $\varepsilon \in \Sigma$, then $\Sigma \vdash_{\mathcal{K}} \varepsilon$ *(reflexivity)*;
- if $\Sigma \vdash_{\mathcal{K}} \varepsilon$ and $\Sigma \subseteq \Pi$, then $\Pi \vdash_{\mathcal{K}} \varepsilon$ *(monotonicity)*;
- if $\Sigma \vdash_{\mathcal{K}} \varepsilon$ and $\Pi \vdash_{\mathcal{K}} \Sigma$, then $\Pi \vdash_{\mathcal{K}} \varepsilon$ *(transitivity)*.

These consequence relations are also *substitution-invariant*; that is, for any $\Sigma \cup \{\varepsilon\} \subseteq \mathrm{Eq}(\overline{x})$ and homomorphism (substitution) $\sigma\colon \mathbf{Fm}(\overline{x}) \to \mathbf{Fm}(\overline{x})$,

- if $\Sigma \vdash_{\mathcal{K}} \varepsilon$, then $\sigma[\Sigma] \vdash_{\mathcal{K}} \sigma(\varepsilon)$ *(substitution-invariance)*.

Certain classes of $\mathcal{L}$-algebras enjoy further useful properties and play an important role in universal algebra and (algebraic) logic. In particular, a class of $\mathcal{L}$-algebras is called a *variety* if it is closed under taking homomorphic images, subalgebras, and direct products. Equivalently, by a famous theorem of Birkhoff, a class of $\mathcal{L}$-algebras is a variety if, and only if, it is an *equational class*, i.e., a class of $\mathcal{L}$-algebras that satisfy some given set of $\mathcal{L}$-equations. Notably, equational consequence in a variety $\mathcal{V}$ is *finitary*; that is, for any $\Sigma \cup \{\varepsilon\} \subseteq \mathrm{Eq}(\overline{x})$,

- if $\Sigma \vdash_{\mathcal{V}} \varepsilon$, then $\Sigma' \vdash_{\mathcal{V}} \varepsilon$ for some finite $\Sigma' \subseteq \Sigma$ *(finitarity)*.

This statement can be deduced using either the compactness theorem of first-order logic or Lemma 9 below and the fact that the congruences of any algebra form an algebraic lattice.

A variety $\mathcal{V}$ of $\mathcal{L}$-algebras may serve as an algebraic semantics for a propositional logic L, viewed as a substitution-invariant consequence relation $\vdash_{\mathrm{L}}$ defined over $\mathbf{Fm}(\overline{x})$ for a countably infinite set $\overline{x}$. That is, there may exist *transformers* τ and ρ that map formulas to finite sets of equations, and equations to finite sets of formulas, respectively, and satisfy for any $T \cup \{\alpha\} \subseteq \mathrm{Fm}(\overline{x})$ and $\Sigma \cup \{\varepsilon\} \subseteq \mathrm{Eq}(\overline{x})$,

- $T \vdash_{\mathrm{L}} \alpha \iff \bigcup \tau[T] \vdash_{\mathcal{V}} \tau(\alpha)$;
- $\Sigma \vdash_{\mathcal{V}} \varepsilon \iff \bigcup \rho[\Sigma] \vdash_{\mathrm{L}} \rho(\varepsilon)$;
- $\{\alpha\} \vdash_{\mathrm{L}} \bigcup \rho[\tau(\alpha)]$ and $\bigcup \rho[\tau(\alpha)] \vdash_{\mathrm{L}} \alpha$;
- $\{\varepsilon\} \vdash_{\mathcal{V}} \bigcup \tau[\rho(\varepsilon)]$ and $\bigcup \tau[\rho(\varepsilon)] \vdash_{\mathcal{V}} \varepsilon$,

and, for any $\alpha \in \mathrm{Fm}(\overline{x})$, $\varepsilon \in \mathrm{Eq}(\overline{x})$, and homomorphism (substitution) $\sigma\colon \mathbf{Fm}(\overline{x}) \to \mathbf{Fm}(\overline{x})$,

- $\sigma[\tau(\alpha)] = \tau(\sigma(\alpha))$ and $\sigma[\rho(\varepsilon)] = \rho(\sigma(\varepsilon))$.

When such transformers exist, they allow us to translate equational consequences into consequences between formulas, and vice versa. In particular, deductive interpolation and other syntactic properties may be interpreted as concerning either equations or formulas, depending on context and convenience. For further details, we refer the reader to the vast literature on abstract algebraic logic (see, e.g., [5, 20]). Let us just note that we restrict our account here to varieties rather than quasivarieties (or more general classes of algebras), partly to avoid additional complexity and partly because varieties already provide algebraic semantics for the most well-studied non-classical logics.

▶ **Example 5.** Heyting algebras form a variety that provides algebraic semantics for IPC via transformers τ, mapping a formula α to the set of equations $\{\alpha \approx \top\}$, and ρ, mapping an equation $\alpha \approx \beta$ to the set of formulas $\{\alpha \to \beta, \beta \to \alpha\}$. By general results of abstract algebraic logic, each axiomatic extension of IPC then has an algebraic semantics (via the same transformers) provided by the variety of Heyting algebras satisfying equations corresponding to the additional axioms. In particular, Boolean algebras and Gödel algebras provide algebraic semantics for CPC and G, respectively. These transformers can also be used to show that varieties of modal algebras serve as algebraic semantics for axiomatic extensions of K. ⌟

▶ **Example 6.** FL-algebras form a variety that provides algebraic semantics for the full Lambek calculus FL via transformers τ, mapping a formula α to the set of equations $\{\alpha \wedge \mathrm{e} \approx \mathrm{e}\}$, and ρ, mapping an equation $\alpha \approx \beta$ to the set of formulas $\{\alpha\backslash\beta, \beta\backslash\alpha\}$ (see, e.g., [29, 58]). Algebraic semantics for other well-known substructural logics are provided by various varieties of FL-algebras (see [29, 58]). In particular, $\mathrm{FL_e}$-algebras and MV-algebras (see Example 4), provide algebraic semantics for $\mathrm{FL_e}$ and Łukasiewicz logic Ł, respectively, and, although a logic of ℓ-groups has not been considered, Abelian ℓ-groups (i.e., ℓ-groups satisfying $xy \approx yx$) provide algebraic semantics for Abelian logic A (see, e.g., [58]). ⌟

▶ Remark 7. It is not hard to see that transformers restricted to single formulas and equations are available for the previous examples; just replace $\{\alpha \to \beta, \beta \to \alpha\}$ and $\{\alpha\backslash\beta, \beta\backslash\alpha\}$ with $\{(\alpha \to \beta) \wedge (\beta \to \alpha)\}$ and $\{(\alpha\backslash\beta) \wedge (\beta\backslash\alpha)\}$, respectively. However, for many logics — including the implicational fragment of IPC with an algebraic semantics provided by Hilbert algebras — more than one formula or equation is necessary (see, e.g., [20] for details). ⌟

3 Consequence and Congruence

In this section, we recall some further elementary notions of universal algebra and present an interpretation of equational consequence in a variety in terms of the congruences of its free algebras (Lemma 9). This result, which provides the scaffolding for proofs of the bridge theorems in subsequent sections, was first established explicitly in [66].

Consider any $\mathcal{L}$-algebra $\mathbf{A}$, recalling that $\mathcal{L}$ is a fixed arbitrary algebraic language with at least one constant symbol. A *congruence* of $\mathbf{A}$ is an equivalence relation on A satisfying for each $f \in \mathcal{L}_n$ and all $a_1, b_1, \ldots, a_n, b_n \in A$,

$$\langle a_1, b_1\rangle, \ldots, \langle a_n, b_n\rangle \in \Theta \implies \langle f^{\mathbf{A}}(a_1, \ldots, a_n), f^{\mathbf{A}}(b_1, \ldots, b_n)\rangle \in \Theta.$$

Notably, the kernel of any homomorphism $\varphi\colon \mathbf{A} \to \mathbf{B}$ is a congruence of $\mathbf{A}$.

The set $\operatorname{Con}\mathbf{A}$ of congruences of $\mathbf{A}$ is closed under taking arbitrary intersections and hence forms a complete lattice $\langle \operatorname{Con}\mathbf{A}, \subseteq\rangle$ with least element $\Delta_A := \{\langle a, a\rangle \mid a \in A\}$ and greatest element $A \times A$. Clearly, $\bigwedge S = \bigcap S$ for any $S \subseteq \operatorname{Con}\mathbf{A}$, but $\bigcup S$ may not be a congruence, so $\bigvee S$ is the congruence of $\mathbf{A}$ generated by $\bigcup S$, i.e., the smallest congruence

of $\mathbf{A}$ containing $\bigcup S$. More generally, the *congruence of* $\mathbf{A}$ *generated by* $R \subseteq A \times A$ is

$$\mathrm{Cg}^{\mathbf{A}}(R) \coloneqq \bigcap\{\Theta \in \mathrm{Con}\,\mathbf{A} \mid R \subseteq \Theta\}.$$

A congruence $\Theta \in \mathrm{Con}\,\mathbf{A}$ is called *finitely generated* if $\Theta = \mathrm{Cg}^{\mathbf{A}}(R)$ for some finite $R \subseteq A \times A$.[3]

Just as normal subgroups and ideals are used to construct quotient groups and rings, so congruences are used to construct quotients of arbitrary algebras. Given any $\mathcal{L}$-algebra $\mathbf{A}$ and $\Theta \in \mathrm{Con}\,\mathbf{A}$, the $\mathcal{L}$-algebra $\mathbf{A}/\Theta$ consists of the set A/Θ of Θ-equivalence classes of A with a well-defined (since Θ is a congruence) operation $f^{\mathbf{A}/\Theta}$ for each $f \in \mathcal{L}_n$ satisfying $f^{\mathbf{A}/\Theta}(a_1/\Theta, \ldots, a_n/\Theta) = f(a_1, \ldots, a_n)/\Theta$ for all $a_1, \ldots, a_n \in A$. Observe also that the *canonical map* $\pi_\Theta \colon A \to A/\Theta;\ a \mapsto a/\Theta$ is a surjective homomorphism from $\mathbf{A}$ onto $\mathbf{A}/\Theta$ with $\ker(\pi_\Theta) = \Theta$, so $\mathrm{Con}\,\mathbf{A}$ consists of precisely the kernels of homomorphisms $\varphi \colon \mathbf{A} \to \mathbf{B}$.

Congruences and quotients are used to formulate generalizations of the usual isomorphism theorems for groups and rings, including the following *general homomorphism theorem*.

▶ **Theorem 8** (cf. [58, Theorem B.2])**.** *For any surjective homomorphism* $\pi \colon \mathbf{A} \to \mathbf{B}$ *and homomorphism* $\varphi \colon \mathbf{A} \to \mathbf{C}$ *with* $\ker(\pi) \subseteq \ker(\varphi)$*, there exists a unique homomorphism* $\psi \colon \mathbf{B} \to \mathbf{C}$ *satisfying* $\psi\pi = \varphi$*; moreover,* ψ *is injective if, and only if,* $\ker(\pi) = \ker(\varphi)$*.*

Now let $\mathcal{V}$ be any variety of $\mathcal{L}$-algebras and let $\overline{x}$ be any set. We obtain a congruence $\Phi_{\mathcal{V}}(\overline{x})$ of the formula algebra $\mathbf{Fm}(\overline{x})$ by defining for $\alpha, \beta \in \mathrm{Fm}(\overline{x})$,

$$\langle \alpha, \beta \rangle \in \Phi_{\mathcal{V}}(\overline{x}) \mathrel{:\Longleftrightarrow} \vdash_{\mathcal{V}} \alpha \approx \beta.$$

Equivalently, $\Phi_{\mathcal{V}}(\overline{x}) = \bigcap I_{\mathcal{V}}(\overline{x})$, where $I_{\mathcal{V}}(\overline{x})$ is the set $\{\Theta \in \mathrm{Con}\,\mathbf{Fm}(\overline{x}) \mid \mathbf{Fm}(\overline{x})/\Theta \in \mathcal{V}\}$. The *free algebra of* $\mathcal{V}$ *over* $\overline{x}$ may then be defined as the quotient algebra

$$\mathbf{F}_{\mathcal{V}}(\overline{x}) \coloneqq \mathbf{Fm}(\overline{x})/\Phi_{\mathcal{V}}(\overline{x}).$$

Let us drop the subscript $\mathcal{V}$ when the variety is clear from the context, and write α to denote both a formula α in $\mathbf{Fm}(\overline{x})$ and its image $\alpha/\Phi(\overline{x})$ under $\pi_{\Phi(\overline{x})}$ in $\mathbf{F}(\overline{x})$. Then $\mathbf{F}(\overline{x})$ is generated by $\overline{x}$ and enjoys the universal mapping property for $\mathcal{V}$: every map from $\overline{x}$ to $\mathbf{A} \in \mathcal{V}$ extends to a unique homomorphism from $\mathbf{F}(\overline{x})$ to $\mathbf{A}$. Observe also that the homomorphism φ from $\mathbf{Fm}(\overline{x})$ to the direct product $\prod\{\mathbf{Fm}(\overline{x})/\Theta \mid \Theta \in I(\overline{x})\}$ satisfying $\varphi(\alpha)(\Theta) = \pi_\Theta(\alpha)$ for each $\alpha \in \mathrm{Fm}(\overline{x})$ and all $\Theta \in I(\overline{x})$, has kernel $\Phi(\overline{x})$. From this observation and Theorem 8, it follows that $\mathbf{F}(\overline{x})$ embeds into a direct product of members of $\mathcal{V}$, and hence, since $\mathcal{V}$ is a variety, $\mathbf{F}(\overline{x}) \in \mathcal{V}$. Let us also assume for convenience, and without loss of generality, that $\mathbf{F}(\overline{x})$ is a subalgebra of $\mathbf{F}(\overline{x}, \overline{y})$ for any disjoint sets of variables $\overline{x}$, $\overline{y}$.

We now have all the ingredients necessary to state and prove the key lemma relating equational consequence in a variety to congruences of its free algebras.

---

[3] The map $\mathrm{Cg}^{\mathbf{A}}$ on $\mathcal{P}(A^2)$ is an *algebraic closure operator* on $A \times A$, corresponding to the fact that $\langle \mathrm{Con}\,\mathbf{A}, \subseteq \rangle$ is an *algebraic lattice* whose compact elements are the finitely generated congruences of $\mathbf{A}$.

▶ **Lemma 9.** *For any variety* $\mathcal{V}$ *and* $\Sigma \cup \{\varepsilon\} \subseteq \mathrm{Eq}(\overline{x})$,

$$\Sigma \vdash_{\mathcal{V}} \varepsilon \iff \varepsilon \in \mathrm{Cg}^{\mathbf{F}(\overline{x})}(\Sigma).$$

Proof. Let π denote the canonical map from $\mathbf{Fm}(\overline{x})$ onto $\mathbf{F}(\overline{x})$ and define $\Theta := \mathrm{Cg}^{\mathbf{F}(\overline{x})}(\pi[\Sigma])$, recalling that $\varepsilon \in \mathrm{Cg}^{\mathbf{F}(\overline{x})}(\Sigma)$ is notational shorthand for $\pi(\varepsilon) \in \Theta$.

Suppose first that $\Sigma \vdash_{\mathcal{V}} \varepsilon$ and consider the homomorphism $\varphi\colon \mathbf{F}(\overline{x}) \to \mathbf{F}(\overline{x})/\Theta$; $\alpha \mapsto \alpha/\Theta$. Since $\mathbf{F}(\overline{x})/\Theta \in \mathcal{V}$ and $\Sigma \subseteq \ker(\varphi\pi)$, by assumption, $\varepsilon \in \ker(\varphi\pi)$. Hence $\pi(\varepsilon) \in \ker(\varphi) = \Theta$.

For the converse, suppose that $\pi(\varepsilon) \in \Theta$ and consider any $\mathbf{A} \in \mathcal{V}$ and homomorphism $\varphi\colon \mathbf{Fm}(\overline{x}) \to \mathbf{A}$ such that $\Sigma \subseteq \ker(\varphi)$. Since $\ker(\pi) = \Phi_{\mathcal{V}}(\overline{x}) \subseteq \ker(\varphi)$, there exists, by Theorem 8, a homomorphism $\psi\colon \mathbf{F}(\overline{x}) \to \mathbf{A}$ satisfying $\psi\pi = \varphi$. It follows that $\pi[\Sigma] \subseteq \ker(\psi)$ and, by assumption, $\pi(\varepsilon) \in \Theta \subseteq \ker(\psi)$. So $\varepsilon \in \ker(\psi\pi) = \ker(\varphi)$. Hence $\Sigma \vdash_{\mathcal{V}} \varepsilon$. ◀

4 Local Deduction Theorems and the Congruence Extension Property

In this section, we illustrate the usefulness of Lemma 9 by relating the well-known congruence extension property to a general 'local deduction theorem' for equational consequence known as the extension property. This property was studied in [63] as the 'limited GINT' and Theorem 13 below may be viewed as a refinement of Theorem 8 from this paper; it also appears in an abstract algebraic logic setting as the 'extension interpolation property' in [11], and as one of the model-theoretic properties considered in [2].

Many propositional logics admit a 'local deduction theorem', which allows a formula occurring as a premise in a consequence to be combined with the conclusion and vice versa. The following property provides a general formulation of such relationships in the setting of equational consequence for varieties.

A variety $\mathcal{V}$ has the *extension property* (EP) if for any $\Sigma \subseteq \mathrm{Eq}(\overline{x}, \overline{y})$ and $\Pi \cup \{\varepsilon\} \subseteq \mathrm{Eq}(\overline{y})$ satisfying $\Sigma \cup \Pi \vdash_{\mathcal{V}} \varepsilon$, there exists a $\Delta \subseteq \mathrm{Eq}(\overline{y})$ satisfying $\Sigma \vdash_{\mathcal{V}} \Delta$ and $\Delta \cup \Pi \vdash_{\mathcal{V}} \varepsilon$.

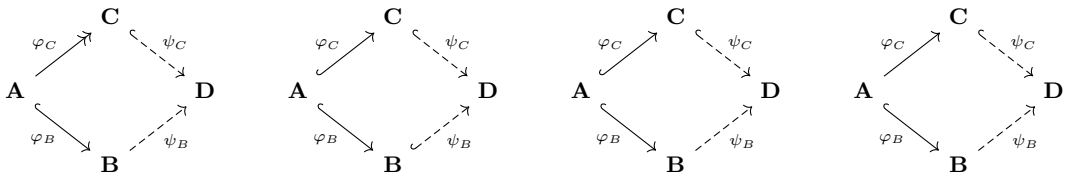
▶ **Example 10.** The fact that every variety of Heyting algebras has the EP is a direct consequence of the deduction theorem for superintuitionistic logics; that is, for any such logic L and set of formulas $T \cup \{\alpha, \beta\}$,

$$T \cup \{\alpha\} \vdash_{\mathrm{L}} \beta \iff T \vdash_{\mathrm{L}} \alpha \to \beta.$$

Similarly, every variety of modal algebras has the EP by virtue of the local deduction theorem for normal modal logics; that is, for any such logic L and set of formulas $T \cup \{\alpha, \beta\}$,

$$T \cup \{\alpha\} \vdash_{\mathrm{L}} \beta \iff T \vdash_{\mathrm{L}} (\alpha \wedge \Box\alpha \wedge \cdots \wedge \Box^n\alpha) \to \beta \text{ for some } n \in \mathbb{N}.$$

If the variety of modal algebras satisfies $\Box x \leq \Box\Box x$ (i.e., L is an axiomatic extension of K4), then the right-hand side of the above equivalence can be simplified to $T \vdash_{\mathrm{L}} (\alpha \wedge \Box\alpha) \to \beta$. ⌟



(i) CEP (ii) AP (iii) 1AP (iv) TIP

Figure 1 Commutative diagrams for algebraic properties.

Let us show now, using the translation schema provided by Lemma 9, that a variety $\mathcal{V}$ has the EP if, and only if, for any $\Theta \in \operatorname{Con}\mathbf{F}(\overline{x},\overline{y})$ and $\Psi \in \operatorname{Con}\mathbf{F}(\overline{y})$,

$$(\operatorname{Cg}^{\mathbf{F}(\overline{x},\overline{y})}(\Psi) \vee \Theta) \cap \mathrm{F}(\overline{y})^2 = \Psi \vee (\Theta \cap \mathrm{F}(\overline{y})^2). \tag{1}$$

Suppose first that $\mathcal{V}$ has the EP and consider any $\Theta \in \operatorname{Con}\mathbf{F}(\overline{x},\overline{y})$ and $\Psi \in \operatorname{Con}\mathbf{F}(\overline{y})$. For the non-trivial inclusion of (1), let $\varepsilon \in (\operatorname{Cg}^{\mathbf{F}(\overline{x},\overline{y})}(\Psi) \vee \Theta) \cap \mathrm{F}(\overline{y})^2$. Then $\Theta \cup \Psi \vdash_{\mathcal{V}} \varepsilon$ and, by the EP, there exists a $\Delta \subseteq \mathrm{Eq}(\overline{y})$ such that $\Theta \vdash_{\mathcal{V}} \Delta$ and $\Delta \cup \Psi \vdash_{\mathcal{V}} \varepsilon$. Hence $\Delta \subseteq \Theta \cap \mathrm{F}(\overline{y})^2$ and, as required, $\varepsilon \in \Psi \vee (\Theta \cap \mathrm{F}(\overline{y})^2)$. For the converse, given $\Sigma \subseteq \mathrm{Eq}(\overline{x},\overline{y})$ and $\Pi \cup \{\varepsilon\} \subseteq \mathrm{Eq}(\overline{y})$ satisfying $\Sigma \cup \Pi \vdash_{\mathcal{V}} \varepsilon$, let $\Theta := \operatorname{Cg}^{\mathbf{F}(\overline{x},\overline{y})}(\Sigma)$ and $\Psi := \operatorname{Cg}^{\mathbf{F}(\overline{y})}(\Pi)$. Then applying (1) and defining $\Delta := \Theta \cap \mathrm{F}(\overline{y})^2$ yields $\Sigma \vdash_{\mathcal{V}} \Delta$ and $\Delta \cup \Pi \vdash_{\mathcal{V}} \varepsilon$.

We use this recasting of the EP to relate it to the following well-known algebraic property.

A class $\mathcal{K}$ of $\mathcal{L}$-algebras has the *congruence extension property* (CEP) if for any $\mathbf{B} \in \mathcal{K}$, subalgebra $\mathbf{A}$ of $\mathbf{B}$, and $\Theta \in \operatorname{Con}\mathbf{A}$, there exists a $\Phi \in \operatorname{Con}\mathbf{B}$ such that $\Phi \cap A^2 = \Theta$.

It is often convenient to use a slight reformulation of this property, observing that $\mathcal{K}$ has the CEP if, and only if, $\operatorname{Cg}^{\mathbf{B}}(\Theta) \cap A^2 = \Theta$ for any $\mathbf{B} \in \mathcal{K}$, subalgebra $\mathbf{A}$ of $\mathbf{B}$, and $\Theta \in \operatorname{Con}\mathbf{A}$.

Like many of the algebraic properties considered in this chapter, the CEP admits an elegant presentation via commutative diagrams. A *span* in a class of $\mathcal{L}$-algebras $\mathcal{K}$ is a 5-tuple $\langle \mathbf{A},\mathbf{B},\mathbf{C},\varphi_B,\varphi_C\rangle$ consisting of algebras $\mathbf{A},\mathbf{B},\mathbf{C} \in \mathcal{K}$ and homomorphisms $\varphi_B\colon \mathbf{A} \to \mathbf{B}$, $\varphi_C\colon \mathbf{A} \to \mathbf{C}$. We call this span *injective* if φ_B is an embedding, *doubly injective* if both φ_B and φ_C are embeddings, and *injective-surjective* if φ_B is an embedding and φ_C is surjective.

We claim that a variety $\mathcal{V}$ has the CEP if, and only if, for any injective-surjective span $\langle \mathbf{A},\mathbf{B},\mathbf{C},\varphi_B,\varphi_C\rangle$ in $\mathcal{V}$, there exist a $\mathbf{D} \in \mathcal{V}$, a homomorphism $\psi_B\colon \mathbf{B} \to \mathbf{D}$, and an embedding $\psi_C\colon \mathbf{C} \to \mathbf{D}$ such that $\psi_B\varphi_B = \psi_C\varphi_C$, that is, the diagram in Figure 1(i) is commutative. Suppose first that $\mathcal{V}$ has the CEP. Let $\langle \mathbf{A},\mathbf{B},\mathbf{C},\varphi_B,\varphi_C\rangle$ be any injective-surjective span in $\mathcal{V}$, assuming without loss of generality that φ_B is the inclusion map. Then $\Theta := \ker(\varphi_C) \in \operatorname{Con}\mathbf{A}$ and Theorem 8 yields an isomorphism $\chi\colon \mathbf{A}/\Theta \to \mathbf{C}$; $a/\Theta \mapsto \varphi_C(a)$. Let $\Phi := \operatorname{Cg}^{\mathbf{B}}(\Theta)$ and $\mathbf{D} := \mathbf{B}/\Phi$, noting that $\Phi \cap A^2 = \Theta$, by the CEP. It follows that $\psi'_C\colon \mathbf{A}/\Theta \to \mathbf{D}$; $a/\Theta \mapsto a/\Phi$ is an embedding. Hence, defining $\psi_B := \pi_\Phi$ and $\psi_C = \psi'_C\chi^{-1}$, for any $a \in A$,

$$\psi_B\varphi_B(a) = a/\Phi = \psi'_C(a/\Theta) = \psi'_C\chi^{-1}\chi(a/\Theta) = \psi_C\varphi_C(a).$$

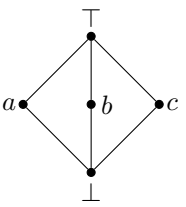
For the converse, suppose that the property holds, and consider any subalgebra $\mathbf{A}$ of $\mathbf{B} \in \mathcal{V}$ and congruence $\Theta \in \operatorname{Con}\mathbf{A}$. We obtain an injective-surjective span $\langle \mathbf{A}, \mathbf{B}, \mathbf{A}/\Theta, \varphi_B, \varphi_C \rangle$, where φ_B is the inclusion map and $\varphi_C \coloneqq \pi_\Theta$. By assumption, there exist a $\mathbf{D} \in \mathcal{V}$, a homomorphism $\psi_B \colon \mathbf{B} \to \mathbf{D}$, and an embedding $\psi_C \colon \mathbf{A}/\Theta \to \mathbf{D}$ such that $\psi_B\varphi_B = \psi_C\varphi_C$. Let $\Phi \coloneqq \ker(\psi_B) \in \operatorname{Con}\mathbf{B}$. Then for any $\langle a_1, a_2 \rangle \in \Phi \cap A^2$,

$$\psi_C(a_1/\Theta) = \psi_C\varphi_C(a_1) = \psi_B(a_1) = \psi_B(a_2) = \psi_C\varphi_C(a_2) = \psi_C(a_2/\Theta),$$

and hence, by injectivity, $a_1/\Theta = a_2/\Theta$, i.e., $\langle a_1, a_2 \rangle \in \Theta$. So $\Phi \cap A^2 = \Theta$ as required.

▶ **Example 11.** Since the congruence lattice of a group $\mathbf{G} = \langle G, \cdot, {}^{-1}, \mathrm{e} \rangle$ is isomorphic to the lattice of its normal subgroups, $\mathbf{G}$ has the CEP if, and only if, for any subgroup $\mathbf{H}$ of $\mathbf{G}$ and normal subgroup $\mathbf{N}$ of $\mathbf{H}$, there exists a normal subgroup $\mathbf{K}$ of $\mathbf{G}$ satisfying $K \cap H = N$. Clearly, the variety of Abelian groups has the CEP, since in this case every subgroup of $\mathbf{G}$ is normal, and any subgroup of a subgroup of $\mathbf{G}$ is also a subgroup of $\mathbf{G}$. However, the variety of groups does not have the CEP, since, for example, the alternating group $\mathbf{A}_5$ is simple, but has subgroups that are not simple. ⌟

▶ **Example 12.** The variety $\mathcal{BL}at$ of bounded lattices does not have the CEP. Consider, for example, the bounded lattice $\mathbf{M}_5 = \langle \{\bot, a, b, c, \top\}, \wedge, \vee, \bot, \top \rangle$ depicted by the Hasse diagram:



Note that $\operatorname{Con}\mathbf{M}_5 = \{\Delta_{M_5}, (M_5)^2\}$, i.e., $\mathbf{M}_5$ is simple. Let $\mathbf{A}$ be the sublattice of $\mathbf{M}_5$ with $A = \{\bot, a, c, \top\}$ and let Θ be the congruence of $\mathbf{A}$ with congruence classes $\{a, \bot\}$ and $\{c, \top\}$. Then $\Delta_{M_5} \cap A^2 = \Delta_A \neq \Theta$ and $(M_5)^2 \cap A^2 = A^2 \neq \Theta$, so $\mathbf{M}_5$ does not have the CEP. ⌟

We now establish the promised bridge between the CEP and the EP, noting that one direction implies that varieties of algebras such as Abelian groups enjoy a 'local deduction theorem', while the other direction implies that varieties corresponding to propositional logics, such as Heyting algebras and modal algebras, possess a fundamental algebraic property.

▶ **Theorem 13.** *A variety has the congruence extension property if, and only if, it has the extension property.*

Proof. It suffices to prove that a variety $\mathcal{V}$ has the CEP if, and only if, it satisfies (1). For both directions, we will use the fact — a consequence of the correspondence theorem for universal algebra (see, e.g., [58, Theorem B.4]) — that for any $\mathcal{L}$-algebras $\mathbf{B}$ and $\mathbf{C}$, surjective homomorphism $\pi \colon \mathbf{C} \to \mathbf{B}$, and $R \subseteq C \times C$,

$$\pi^{-1}[\mathrm{Cg}^{\mathbf{B}}(\pi[R])] = \mathrm{Cg}^{\mathbf{C}}(R) \vee \ker(\pi). \tag{2}$$

Suppose first that $\mathcal{V}$ has the CEP and consider any $\Theta \in \operatorname{Con}\mathbf{F}(\overline{x},\overline{y})$ and $\Phi \in \operatorname{Con}\mathbf{F}(\overline{y})$. Let $\pi\colon \mathbf{F}(\overline{x},\overline{y}) \to \mathbf{F}(\overline{x},\overline{y})/\Theta$ be the canonical map with $\ker(\pi) = \Theta$ and let φ be the restriction of π to $\mathbf{F}(\overline{y})$ with $\ker(\varphi) = \Theta \cap \mathrm{F}(\overline{y})^2$. Define $\mathbf{B} \coloneqq \pi(\mathbf{F}(\overline{x},\overline{y})) = \mathbf{F}(\overline{x},\overline{y})/\Theta$ and $\mathbf{A} \coloneqq \varphi(\mathbf{F}(\overline{y}))$. Using (2) in the first and fifth steps and the CEP for the third,

$$\begin{aligned}
(\mathrm{Cg}^{\mathbf{F}(\overline{x},\overline{y})}(\Phi) \vee \Theta) \cap \mathrm{F}(\overline{y})^2 &= \pi^{-1}[\mathrm{Cg}^{\mathbf{B}}(\pi[\Phi])] \cap \mathrm{F}(\overline{y})^2 \\
&\subseteq \pi^{-1}[\mathrm{Cg}^{\mathbf{B}}(\varphi[\Phi]) \cap A^2] \\
&= \pi^{-1}[\mathrm{Cg}^{\mathbf{A}}(\varphi[\Phi])] \\
&= \varphi^{-1}[\mathrm{Cg}^{\mathbf{A}}(\varphi[\Phi])] \\
&= \Phi \vee (\Theta \cap \mathrm{F}(\overline{y})^2) \\
&\subseteq (\mathrm{Cg}^{\mathbf{F}(\overline{x},\overline{y})}(\Phi) \vee \Theta) \cap \mathrm{F}(\overline{y})^2.
\end{aligned}$$

For the converse, suppose that $\mathcal{V}$ satisfies (1) and consider any congruence Θ of a subalgebra $\mathbf{A}$ of some $\mathbf{B} \in \mathcal{V}$. Let $\overline{y} \coloneqq A$ and $\overline{x} \coloneqq B \backslash A$, noting that $\Theta \subseteq A^2 \subseteq \mathrm{F}(\overline{y})^2 \subseteq \mathrm{F}(\overline{x},\overline{y})^2$. Consider the surjective homomorphisms $\varphi\colon \mathbf{F}(\overline{y}) \to \mathbf{A}$ and $\pi\colon \mathbf{F}(\overline{x},\overline{y}) \to \mathbf{B}$ extending the identity maps on $\overline{y}$ and $\overline{x},\overline{y}$, respectively. Define $\Phi \coloneqq \ker(\pi)$, noting that $\ker(\varphi) = \Phi \cap \mathrm{F}(\overline{y})^2$. Using (2) in the first and fifth steps and (1) in the third,

$$\begin{aligned}
\mathrm{Cg}^{\mathbf{B}}(\Theta) \cap A^2 &= \pi[(\mathrm{Cg}^{\mathbf{F}(\overline{x},\overline{y})}(\Theta) \vee \Phi] \cap A^2 \\
&\subseteq \pi[(\mathrm{Cg}^{\mathbf{F}(\overline{x},\overline{y})}(\Theta) \vee \Phi) \cap \mathrm{F}(\overline{y})^2] \\
&= \pi[\mathrm{Cg}^{\mathbf{F}(\overline{y})}(\Theta) \vee (\Phi \cap \mathrm{F}(\overline{y})^2)] \\
&= \varphi[\mathrm{Cg}^{\mathbf{F}(\overline{y})}(\Theta) \vee (\Phi \cap \mathrm{F}(\overline{y})^2)] \\
&= \mathrm{Cg}^{\mathbf{A}}(\varphi[\Theta]) \\
&= \Theta \\
&\subseteq \mathrm{Cg}^{\mathbf{B}}(\Theta) \cap A^2.
\end{aligned}$$

◀

▶ **Example 14.** A description of the generation of congruences in a variety can be used to establish an explicit version of the EP, typically described as a 'local deduction theorem'. In particular, every variety of $\mathrm{FL_e}$-algebras $\mathcal{V}$ has the CEP and therefore the EP, but also, more concretely (see [58] for details), for any $\Sigma \subseteq \mathrm{Eq}(\overline{x})$ and $\alpha, \beta \in \mathrm{Fm}(\overline{x})$,

$$\Sigma \cup \{\mathrm{e} \leq \alpha\} \vdash_{\mathcal{V}} \mathrm{e} \leq \beta \iff \Sigma \vdash_{\mathcal{V}} (\alpha \wedge \mathrm{e})^n \leq \beta \text{ for some } n \in \mathbb{N}.$$

Equivalently, if L is an axiomatic extension of $\mathrm{FL_e}$, then for any $T \cup \{\alpha, \beta\} \subseteq \mathrm{Fm}(\overline{x})$,

$$T \cup \{\alpha\} \vdash_{\mathrm{L}} \beta \iff T \vdash_{\mathcal{V}} \alpha^n \to \beta \text{ for some } n \in \mathbb{N},$$

which for axiomatic extensions of IPC simplifies to the familiar deduction theorem with $n = 1$. Note, however, that the variety of FL-algebras does not have the CEP (see, e.g., [29, p. 217]), so FL does not admit a local deduction theorem of this form. ⌟

A significant obstacle to establishing an algebraic property such as the CEP for a variety is the fact that in principle it should be established for *all* its members. There exist, however, 'transfer' results in the literature, however, that reduce such problems to more manageable subclasses, specifically the (finitely) subdirectly irreducible members that serve as 'building blocks' for all members of the variety.

A *subdirect product* of a family of $\mathcal{L}$-algebras $\{\mathbf{B}_i\}_{i\in I}$ is a subalgebra $\mathbf{A}$ of $\prod_{i\in I}\mathbf{B}_i$ such that the projection map $\pi_i\colon \mathbf{A}\to\mathbf{B}_i$; $a\mapsto a(i)$ is surjective for each $i\in I$. An $\mathcal{L}$-algebra $\mathbf{A}$ is *(finitely) subdirectly irreducible* if for any isomorphism φ between $\mathbf{A}$ and a subdirect product of a (non-empty finite) family of $\mathcal{L}$-algebras $\{\mathbf{B}_i\}_{i\in I}$, there is an $i\in I$ such that $\pi_i\varphi$ is an isomorphism. Equivalently, an $\mathcal{L}$-algebra $\mathbf{A}$ is subdirectly irreducible if Δ_A is completely meet-irreducible in $\operatorname{Con}\mathbf{A}$ and finitely subdirectly irreducible if Δ_A is meet-irreducible in $\operatorname{Con}\mathbf{A}$.[4] By Birkhoff's subdirect representation theorem (see, e.g., [58, Theorem B.7]), every $\mathcal{L}$-algebra is isomorphic to a subdirect product of subdirectly irreducible $\mathcal{L}$-algebras.

Let $\mathcal{V}_{\text{SI}}$ and $\mathcal{V}_{\text{FSI}}$ denote the classes of subdirectly irreducible and finitely subdirectly irreducible members of $\mathcal{V}$, respectively. Under certain conditions, properties such as the CEP transfer from $\mathcal{V}_{\text{SI}}$ or $\mathcal{V}_{\text{FSI}}$ to $\mathcal{V}$ and, in some cases, back again. Often it is easier to consider the larger class $\mathcal{V}_{\text{FSI}}$. In particular, if $\mathcal{V}$ has equationally definable principal congruence meets (a common property for the algebraic semantics of a propositional logic that corresponds to having a suitable disjunction connective), then $\mathcal{V}_{\text{FSI}}$ is a universal class [10, Theorem 2.3].

To obtain a transfer theorem for the CEP, we require that $\mathcal{V}$ be *congruence-distributive*, that is, $\operatorname{Con}\mathbf{A}$ should be distributive for every $\mathbf{A}\in\mathcal{V}$. Since any $\mathcal{L}$-algebra with a lattice reduct is congruence-distributive, this requirement is fulfilled by the algebraic semantics of broad families of propositional logics.

▶ **Theorem 15** ([23, Corollary 2.4])**.** *Let $\mathcal{V}$ be any congruence-distributive variety. Then $\mathcal{V}$ has the congruence extension property if, and only if, $\mathcal{V}_{\text{FSI}}$ has the congruence extension property.*

▶ **Example 16.** Theorem 15 can drastically reduce the amount of work needed to check if a variety has the CEP. For example, the variety $\mathcal{BDL}at$ of bounded distributive lattices is congruence-distributive and $\mathcal{BDL}at_{\text{FSI}}$ contains, up to isomorphism, only the trivial and two-element bounded lattices. Since $\mathcal{BDL}at_{\text{FSI}}$ clearly has the CEP, so does $\mathcal{BDL}at$. ⌟

▶ Remark 17. For a congruence-distributive variety $\mathcal{V}$, each member of $\mathcal{V}_{\text{FSI}}$ embeds into an ultraproduct of members of $\mathcal{V}_{\text{SI}}$ [10, Lemma 1.5]. Theorem 15 therefore implies that if $\mathcal{V}$ is a congruence-distributive variety and $\mathcal{V}_{\text{SI}}$ is an elementary class, then $\mathcal{V}$ has the CEP if, and only if, $\mathcal{V}_{\text{SI}}$ has the CEP. The latter was first proved in [13, Theorem 3.3] and follows also from a similar, but seemingly distinct, result for congruence-modular varieties [47, Theorem 2.3]. ⌟

[4] An element a of a lattice $\mathbf{L}$ is *meet-irreducible* if $a=b\wedge c$ implies $a=b$ or $a=c$, and this is true of the greatest element $\top$ of $\mathbf{L}$ if it has one; however, a is *completely meet-irreducible* if $a=\bigwedge B$ implies $a\in B$ for any $B\subseteq L$, which is not the case for $\top=\bigwedge\emptyset$. In particular, we assume here that trivial algebras are finitely subdirectly irreducible (following, e.g., [10]) but not subdirectly irreducible.

5 Amalgamation and the Robinson Property

In this section, we construct a bridge for varieties between the amalgamation property and a property of equational consequence known as the Robinson property, first established in [66].

Let us fix again an algebraic language $\mathcal{L}$ that has at least one constant symbol and let $\mathcal{K}$ and $\mathcal{K}'$ be any classes of $\mathcal{L}$-algebras. An *amalgam* in $\mathcal{K}'$ of a doubly injective span $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C \rangle$ in $\mathcal{K}$ is a triple $\langle \mathbf{D}, \psi_B, \psi_C \rangle$ consisting of an algebra $\mathbf{D} \in \mathcal{K}'$ and embeddings $\psi_B \colon \mathbf{B} \to \mathbf{D}$ and $\psi_C \colon \mathbf{C} \to \mathbf{D}$ such that $\psi_B \varphi_B = \psi_C \varphi_C$.

A class $\mathcal{K}$ of $\mathcal{L}$-algebras has the *amalgamation property* (AP) if every doubly injective span in $\mathcal{K}$ has an amalgam in $\mathcal{K}$ (see Figure 1(ii)).

Let us note also in passing that a class $\mathcal{K}$ of $\mathcal{L}$-algebras has the *strong amalgamation property* (strong AP) if every doubly injective span $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C \rangle$ in $\mathcal{K}$ has an amalgam $\langle \mathbf{D}, \psi_B, \psi_C \rangle$ in $\mathcal{K}$ satisfying $\psi_B \varphi_B[A] = \psi_B[B] \cap \psi_C[C] = \psi_C \varphi_C[A]$.

▶ **Example 18.** The variety $\mathcal{BL}at$ of bounded lattices has the strong AP [38]. Consider any doubly injective span $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C \rangle$ in $\mathcal{BL}at$, assuming without loss of generality that φ_B and φ_C are inclusion maps and $A = B \cap C$. Let $R \coloneqq \leq^{\mathbf{B}} \cup \leq^{\mathbf{C}}$ and define for $x, y \in B \cup C$,

$$x \preceq y \mathrel{:\Longleftrightarrow} Rxy \text{ or } (Rxz \text{ and } Rzy, \text{ for some } z \in B \cap C).$$

Then $\preceq$ is the smallest partial order on $B \cup C$ extending $\leq^{\mathbf{B}}$ and $\leq^{\mathbf{C}}$. Now let $\mathbf{D}$ be the Dedekind-MacNeille completion of the poset $\langle B \cup C, \preceq \rangle$, i.e., the set of subsets X of $B \cup C$ satisfying $(X^u)^l = X$, ordered by set-inclusion, where Y^l and Y^u denote the sets of lower bounds and upper bounds of $Y \subseteq B \cup C$, respectively. Then the maps ψ_B and ψ_C sending an element x to $\{x\}^l$ in $\langle B \cup C, \preceq \rangle$ are embeddings of $\mathbf{B}$ and $\mathbf{C}$, respectively, into $\mathbf{D}$, and satisfy $\psi_B(x) = \psi_C(x)$, for each $x \in A$. Moreover, since any element in $\psi_B[B] \cap \psi_C[C]$ is of the form $\{b\}^l = \psi_B(b) = \psi_C(c) = \{c\}^l$ for some $b \in B$ and $c \in C$, we obtain $b = c \in A$. Hence $\psi_B \varphi_B[A] = \psi_B[B] \cap \psi_C[C] = \psi_C \varphi_C[A]$. ⌟

▶ **Example 19.** The method described in Example 18 is easily adapted to establish that the variety of bounded semilattices has the strong AP [18], and, with considerably more effort, can then be used to prove that the varieties of implicative semilattices and Heyting algebras have this property [19]. The fact that the variety of Heyting algebras has the strong AP was first proved in [14] and an alternative categorical proof may be found in [67]. ⌟

▶ **Example 20.** Schreier's work on free amalgamated products implies that the variety of groups has the strong AP [69], and it is not hard to see that the variety of Abelian groups also has this property. On the other hand, the varieties of monoids and commutative monoids do not even have the AP. For a counterexample (adapted from [46]), consider the commutative monoids $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ with $A = \{u, v, w, 0, \mathrm{e}\}$, $B = A \cup \{b\}$, and $C = A \cup \{c\}$, where e is the neutral element, $bu = ub = v$, $cv = vc = w$, and all other products are 0. If the doubly injective

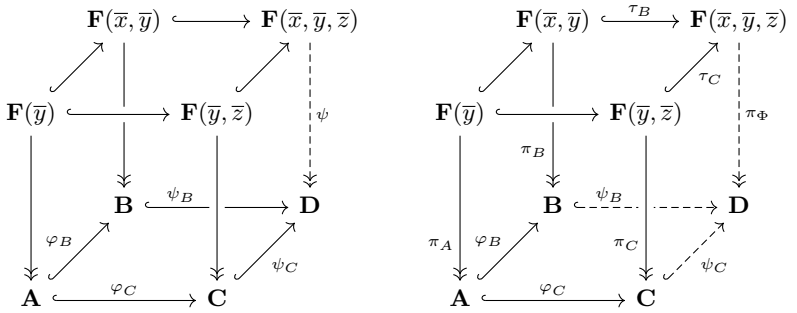


Figure 2 Commutative diagrams for the proof of Theorem 21.

span $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C \rangle$, with inclusion maps φ_B, φ_C, were to have an amalgam $\langle \mathbf{D}, \psi_B, \psi_C \rangle$ in the variety of monoids, then $\psi_C(w) = \psi_C(v)\psi_C(c) = \psi_B(v)\psi_C(c) = \psi_B(b)\psi_B(u)\psi_C(c) = \psi_B(b)\psi_C(u)\psi_C(c) = \psi_B(b)\psi_C(0) = \psi_B(b)\psi_B(0) = \psi_B(0) = \psi_C(0)$, contradicting $w \neq 0$. ⌟

We now introduce the relevant property of equational consequence for varieties.

A variety $\mathcal{V}$ has the *Robinson property* (RP) if for any $\Sigma \subseteq \mathrm{Eq}(\overline{x}, \overline{y})$ and $\Pi \subseteq \mathrm{Eq}(\overline{y}, \overline{z})$ satisfying $\Sigma \vdash_\nu \delta \iff \Pi \vdash_\nu \delta$ for all $\delta \in \mathrm{Eq}(\overline{y})$, it follows that $\Sigma \cup \Pi \vdash_\nu \varepsilon \iff \Pi \vdash_\nu \varepsilon$ for any $\varepsilon \in \mathrm{Eq}(\overline{y}, \overline{z})$.

▶ **Theorem 21.** *A variety $\mathcal{V}$ has the amalgamation property if, and only if, it has the Robinson property.*

Proof. Observe first, using Lemma 9 to translate between sets of equations and congruences of free algebras, that $\mathcal{V}$ has the RP if, and only if, for any $\Theta \in \mathrm{Con}\,\mathbf{F}(\overline{x}, \overline{y})$ and $\Psi \in \mathrm{Con}\,\mathbf{F}(\overline{y}, \overline{z})$ satisfying $\Theta \cap \mathrm{F}(\overline{y})^2 = \Psi \cap \mathrm{F}(\overline{y})^2$, there exists a $\Phi \in \mathrm{Con}\,\mathbf{F}(\overline{x}, \overline{y}, \overline{z})$ satisfying $\Theta = \Phi \cap \mathrm{F}(\overline{x}, \overline{y})^2$ and $\Psi = \Phi \cap \mathrm{F}(\overline{y}, \overline{z})^2$.

For the left-to-right direction, suppose that $\mathcal{V}$ has the AP and consider any $\Theta \in \mathrm{Con}\,\mathbf{F}(\overline{x}, \overline{y})$ and $\Psi \in \mathrm{Con}\,\mathbf{F}(\overline{y}, \overline{z})$ satisfying $\Theta_0 := \Theta \cap \mathrm{F}(\overline{y})^2 = \Psi \cap \mathrm{F}(\overline{y})^2$. Define $\mathbf{A} := \mathbf{F}(\overline{y})/\Theta_0$, $\mathbf{B} := \mathbf{F}(\overline{x}, \overline{y})/\Theta$, and $\mathbf{C} := \mathbf{F}(\overline{y}, \overline{z})/\Psi$. It follows that $\varphi_B \colon \mathbf{A} \to \mathbf{B}$; $[\alpha]_{\Theta_0} \mapsto [\alpha]_\Theta$ and $\varphi_C \colon \mathbf{A} \to \mathbf{C}$; $[\alpha]_{\Theta_0} \mapsto [\alpha]_\Psi$ are embeddings, and hence, by assumption, $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C \rangle$ has an amalgam $\langle \mathbf{D}, \psi_B, \psi_C \rangle$ in $\mathcal{V}$. Moreover, we may assume without loss of generality that $\mathbf{D}$ is generated by $\psi_B[B] \cup \psi_C[C]$.

Let $\psi \colon \mathbf{F}(\overline{x}, \overline{y}, \overline{z}) \to \mathbf{D}$ be the unique surjective homomorphism that maps each $x \in \overline{x}$ to $\psi_B([x]_\Theta)$, each $y \in \overline{y}$ to $\psi_B([y]_\Theta) = \psi_C([y]_\Psi)$, and each $z \in \overline{z}$ to $\psi_C([z]_\Psi)$, as illustrated in the leftmost diagram of Figure 2. Let $\Phi := \ker(\psi)$. We claim that $\Theta = \Phi \cap \mathrm{F}(\overline{x}, \overline{y})^2$ and $\Psi = \Phi \cap \mathrm{F}(\overline{y}, \overline{z})^2$, proving just the non-trivial inclusion of the first equality. Let $\langle \alpha, \beta \rangle \in \Phi \cap \mathrm{F}(\overline{x}, \overline{y})^2$. Since ψ is determined by the prescribed values for the generators

of $\mathbf{F}(\overline{x},\overline{y},\overline{z})$, clearly $\psi(\alpha) = \psi_B([\alpha]_\Theta)$ and $\psi(\beta) = \psi_B([\beta]_\Theta)$. But $\langle \alpha, \beta\rangle \in \Phi = \ker(\psi)$, so $\psi_B([\alpha]_\Theta) = \psi(\alpha) = \psi(\beta) = \psi_B([\beta]_\Theta)$, and, since ψ_B is injective, $[\alpha]_\Theta = [\beta]_\Theta$ and $\langle \alpha, \beta\rangle \in \Theta$.

For the right-to-left direction, suppose that $\mathcal{V}$ satisfies the above condition and consider any doubly injective span $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C\rangle$ in $\mathcal{V}$, assuming without loss of generality that φ_B and φ_C are inclusion maps with $B = \overline{x},\overline{y}$, $C = \overline{y},\overline{z}$, and $A = B \cap C = \overline{y}$. Extending identity maps, we obtain surjective homomorphisms $\pi_A\colon \mathbf{F}(\overline{y}) \to \mathbf{A}$, $\pi_B\colon \mathbf{F}(\overline{x},\overline{y}) \to \mathbf{B}$, and $\pi_C\colon \mathbf{F}(\overline{y},\overline{z}) \to \mathbf{C}$. Define $\Theta_0 := \ker(\pi_A)$, $\Theta := \ker(\pi_B)$, and $\Psi := \ker(\pi_C)$, observing that $\Theta_0 = \Theta \cap \mathrm{F}(\overline{y})^2 = \Psi \cap \mathrm{F}(\overline{y})^2$. By assumption, there exists a $\Phi \in \operatorname{Con}\mathbf{F}(\overline{x},\overline{y},\overline{z})$ such that $\Theta = \Phi \cap \mathrm{F}(\overline{x},\overline{y})^2$ and $\Psi = \Phi \cap \mathrm{F}(\overline{y},\overline{z})^2$. Define $\mathbf{D} := \mathbf{F}(\overline{x},\overline{y},\overline{z})/\Phi$ and let $\pi_\Phi\colon \mathbf{F}(\overline{x},\overline{y},\overline{z}) \to \mathbf{D}$; $\alpha \mapsto [\alpha]_\Phi$ be the canonical map and $\tau_B\colon \mathbf{F}(\overline{x},\overline{y}) \to \mathbf{F}(\overline{x},\overline{y},\overline{z})$ and $\tau_C\colon \mathbf{F}(\overline{y},\overline{z}) \to \mathbf{F}(\overline{x},\overline{y},\overline{z})$ be inclusion maps. Then $\ker(\pi_\Phi\tau_B) = \ker(\pi_B)$ and $\ker(\pi_\Phi\tau_C) = \ker(\pi_C)$. Hence, by Theorem 8, there exist embeddings $\psi_B\colon \mathbf{B} \to \mathbf{D}$ and $\psi_C\colon \mathbf{C} \to \mathbf{D}$ such that $\psi_B\pi_B = \pi_\Phi\tau_B$ and $\psi_C\pi_C = \pi_\Phi\tau_C$, as illustrated by the rightmost diagram of Figure 2. So $\psi_B\varphi_B = \psi_C\varphi_C$ and $\langle \mathbf{D}, \psi_B, \psi_C\rangle$ is an amalgam of $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C\rangle$ in $\mathcal{V}$. ◀

In order to obtain a precise match between amalgamation in a variety $\mathcal{V}$ and the subclass $\mathcal{V}_{\mathrm{FSI}}$, we consider a property that is, in some contexts at least, weaker than the AP.

A class $\mathcal{K}$ of $\mathcal{L}$-algebras has the *one-sided amalgamation property* (1AP) if for any doubly injective span $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C\rangle$ in $\mathcal{K}$, there exist a $\mathbf{D} \in \mathcal{K}$, a homomorphism $\psi_B\colon \mathbf{B} \to \mathbf{D}$, and an embedding $\psi_C\colon \mathbf{C} \to \mathbf{D}$ such that $\psi_B\varphi_B = \psi_C\varphi_C$ (see Figure 1(iii)).

In fact, a variety $\mathcal{V}$ has the 1AP if, and only, if it has the AP. For the non-trivial direction, we apply the 1AP to a doubly injective span $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C\rangle$ and then again to the doubly injective span $\langle \mathbf{A}, \mathbf{C}, \mathbf{B}, \varphi_C, \varphi_B\rangle$ to obtain $\mathbf{D}_1, \mathbf{D}_2 \in \mathcal{V}$ with appropriate homomorphisms and embeddings, and obtain an amalgam $\mathbf{D}_1 \times \mathbf{D}_2 \in \mathcal{V}$ equipped with the induced embeddings.

We now state a useful transfer theorem for the AP and 1AP, recalling that many varieties that serve as algebraic semantics for propositional logics have both the CEP — e.g., via a local deduction theorem — and a class of finitely subdirectly irreducible algebras that is closed under taking subalgebras — often by virtue of having a suitable disjunction connective.

▶ **Theorem 22** ([23, Corollary 3.5]). *Let $\mathcal{V}$ be any variety with the congruence extension property such that $\mathcal{V}_{\mathrm{FSI}}$ is closed under taking subalgebras. Then $\mathcal{V}$ has the amalgamation property if, and only if, $\mathcal{V}_{\mathrm{FSI}}$ has the one-sided amalgamation property.*

This transfer theorem, which extends results in [36, 57] relating amalgamation in $\mathcal{V}$ to amalgamation in $\mathcal{V}_{\mathrm{SI}}$, is particularly useful when investigating amalgamation in a variety $\mathcal{V}$ where the members of $\mathcal{V}_{\mathrm{FSI}}$ have some simple structural features.

▶ **Example 23.** Recall from Example 16 that $\mathcal{BDL}at$ has the CEP and that $\mathcal{BDL}at_{\mathrm{FSI}}$ contains, up to isomorphism, only the trivial and two-element bounded lattices. Clearly, $\mathcal{BDL}at_{\mathrm{FSI}}$ has the 1AP, so $\mathcal{BDL}at$ has the AP. Indeed, $\mathcal{BL}at$ and $\mathcal{BDL}at$ are the only non-trivial varieties of

bounded lattices that have the AP [15]. On the other hand, $\mathcal{BDL}at$, unlike $\mathcal{BL}at$, does not have the strong AP. For a counterexample, consider the sublattices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathcal{BDL}at$ of $\mathbf{M}_5$ (see Example 12) with $A = \{\bot, a, \top\}$, $B = \{\bot, a, b, \top\}$, and $C = \{\bot, a, c, \top\}$, and suppose that the doubly injective span $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C \rangle$ in $\mathcal{BDL}at$, with inclusion maps φ_B, φ_C, has an amalgam $\langle \mathbf{D}, \psi_B, \psi_C \rangle$ in $\mathcal{BDL}at$. Then $\psi_B(b)$ and $\psi_C(c)$ are both complements of $\psi_B(a) = \psi_C(a)$ in $\mathbf{D}$ and, by the uniqueness of complements in bounded distributive lattices, $\psi_B(b) = \psi_C(c)$. So $\psi_B[A] \neq \psi_B[B] \cap \psi_C[C] \neq \psi_C[A]$. ⌟

▶ **Example 24.** A variety $\mathcal{V}$ of Heyting algebras generated by a finite totally ordered Heyting algebra $\mathbf{H}$ has the AP if, and only if, $n := |H| \leq 3$. Using Jónsson's Lemma for congruence-distributive varieties [43], the class $\mathcal{V}_{\text{FSI}}$ contains exactly n algebras up to isomorphism. For $n \leq 3$, it is clear that $\mathcal{V}_{\text{FSI}}$ has the 1AP and hence $\mathcal{V}$ has the AP. For $n > 3$, we obtain a counterexample by considering $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathcal{V}$ such that $A = \{\bot, a_1, \ldots, a_{n-3}, \top\}$, $B = \{\bot, b, a_1, \ldots, a_{n-3}, \top\}$, and $C = \{\bot, a_1, \ldots, a_{n-3}, c, \top\}$ with $\bot < b < a_1 < \cdots < a_{n-3} < \top$ and $\bot < a_1 < \cdots < a_{n-3} < c < \top$. If the doubly injective span $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C \rangle$, with inclusion maps φ_B, φ_C, were to have an amalgam $\langle \mathbf{D}, \psi_B, \psi_C \rangle$ in $\mathcal{V}$, then $\mathcal{V}$ would contain the $n+1$-element totally ordered Heyting algebra generated by $\psi_B[B] \cup \psi_C[C]$, contradicting the fact that $\mathcal{V} \models \top \approx x_1 \vee (x_1 \to x_2) \vee (x_2 \to x_3) \vee \cdots \vee (x_{n-1} \to x_n)$. ⌟

▶ **Example 25.** An FL-algebra is said to be *semilinear* if it is isomorphic to a subdirect product of totally ordered FL-algebras. Such algebras provide algebraic semantics for broad families of many-valued logics (see, e.g., [58]). In particular, *BL-algebras* — algebraic semantics for Hájek's basic fuzzy logic BL — are term-equivalent to semilinear FL_e-algebras satisfying $\text{f} \leq x$ and $x(x \to y) \approx x \wedge y$, while Gödel algebras and MV-algebras are term-equivalent to BL-algebras satisfying $xx \approx x$ and $(x \to \text{f}) \to \text{f} \approx x$, respectively.

If $\mathcal{V}$ is a variety of semilinear FL-algebras, then $\mathcal{V}_{\text{FSI}}$ consists of its totally ordered members, and if $\mathcal{V}$ also has the CEP, then it has the AP if, and only if, $\mathcal{V}_{\text{FSI}}$ has the 1AP. This correspondence has been used to establish the AP or its failure for a wide range of varieties of semilinear FL-algebras (see, e.g., [55, 56, 57, 31, 24, 22, 27, 26, 33]). For example, continuum-many varieties of semilinear FL-algebras satisfying $xx \approx x$ have the AP, but only finitely many of these satisfy $xy \approx yx$ [24]. A full description of the varieties of BL-algebras that have the AP has been given in [27]; these include all varieties (and no more) of MV-algebras generated by a single totally-ordered algebra [16], exactly three non-trivial varieties of Gödel algebras (Boolean algebras, Gödel algebras, and the variety generated by the three-element totally ordered Heyting algebra), and the variety of all BL-algebras [60]. For further details, as well as proofs that the varieties of semilinear FL-algebras and semilinear FL_e-algebras do not have the AP, we refer the reader to the survey article [26]. ⌟

▶ Remark 26. Suppose that $\mathcal{V}$ is a finitely generated variety — i.e., $\mathcal{L}$ is finite and $\mathcal{V}$ is generated as a variety by a given finite set of finite $\mathcal{L}$-algebras — that is congruence-distributive and such that $\mathcal{V}_{\text{FSI}}$ is closed under taking subalgebras. Then there are effective algorithms to decide if $\mathcal{V}$ has the CEP or AP. Using Jónsson's Lemma for congruence-distributive varieties [43], a finite

set $\mathcal{V}^*_{\mathrm{FSI}} \subseteq \mathcal{V}_{\mathrm{FSI}}$ of finite algebras can be constructed such that each $\mathbf{A} \in \mathcal{V}_{\mathrm{FSI}}$ is isomorphic to some $\mathbf{A}^* \in \mathcal{V}^*_{\mathrm{FSI}}$. Hence, by Theorem 15, it can be decided if $\mathcal{V}$ has the CEP by checking if each member of $\mathcal{V}^*_{\mathrm{FSI}}$ has the CEP. Moreover, every finitely generated congruence-distributive variety that has the AP has the CEP [44, Corollary 2.11]. Hence, by Theorem 22, it can also be decided if $\mathcal{V}$ has the AP by checking if $\mathcal{V}^*_{\mathrm{FSI}}$ has the 1AP. ⌟

6 Deductive Interpolation Properties

In this section, we consider a range of (deductive) interpolation properties that may or may not be possessed by a variety $\mathcal{V}$. These properties are all expressed in terms of equational consequence, or, equivalently, via congruences of the free algebras of $\mathcal{V}$, and, like the Robinson property considered in Section 5, transfer to various amalgamation properties of $\mathcal{V}$.

A variety $\mathcal{V}$ has the *deductive interpolation property* (DIP) if for any $\Sigma \subseteq \mathrm{Eq}(\overline{x}, \overline{y})$ and $\varepsilon \in \mathrm{Eq}(\overline{y}, \overline{z})$ satisfying $\Sigma \vdash_{\mathcal{V}} \varepsilon$, there exists a $\Pi \subseteq \mathrm{Eq}(\overline{y})$ satisfying $\Sigma \vdash_{\mathcal{V}} \Pi$ and $\Pi \vdash_{\mathcal{V}} \varepsilon$.

Note that, by the finitarity of equational consequence in a variety $\mathcal{V}$, we may assume that the 'interpolant' Π in this definition is finite. Observe also that $\mathcal{V}$ has the DIP if, and only if, for any $\Sigma \subseteq \mathrm{Eq}(\overline{x}, \overline{y})$, there exists a $\Gamma \subseteq \mathrm{Eq}(\overline{y})$ (which cannot be assumed to be finite) satisfying $\Sigma \vdash_{\mathcal{V}} \varepsilon \iff \Gamma \vdash_{\mathcal{V}} \varepsilon$ for any $\varepsilon \in \mathrm{Eq}(\overline{y}, \overline{z})$. The right-to-left direction is immediate. For the left-to-right direction, given $\Sigma \subseteq \mathrm{Eq}(\overline{x}, \overline{y})$, define $\Gamma \coloneqq \{\delta \in \mathrm{Eq}(\overline{y}) \mid \Sigma \vdash_{\mathcal{V}} \delta\}$ and consider any $\varepsilon \in \mathrm{Eq}(\overline{y}, \overline{z})$. If $\Gamma \vdash_{\mathcal{V}} \varepsilon$, then clearly $\Sigma \vdash_{\mathcal{V}} \varepsilon$. Conversely, if $\Sigma \vdash_{\mathcal{V}} \varepsilon$, then, by the DIP, there exists a $\Pi \subseteq \mathrm{Eq}(\overline{y})$ satisfying $\Sigma \vdash_{\mathcal{V}} \Pi$ and $\Pi \vdash_{\mathcal{V}} \varepsilon$. By definition, $\Pi \subseteq \Gamma$, so $\Gamma \vdash_{\mathcal{V}} \varepsilon$.

The DIP can also be formulated as a property of embeddings between congruence lattices of free algebras. Observe first that the inclusion map $\tau \colon \mathbf{F}(\overline{y}) \to \mathbf{F}(\overline{x}, \overline{y})$ 'lifts' to the maps

$$\begin{aligned} \tau^* &\colon \mathrm{Con}\,\mathbf{F}(\overline{y}) \to \mathrm{Con}\,\mathbf{F}(\overline{x}, \overline{y}); \quad \Theta \mapsto \mathrm{Cg}^{\mathbf{F}(\overline{x}, \overline{y})}(\tau[\Theta]) \\ \tau^{-1} &\colon \mathrm{Con}\,\mathbf{F}(\overline{x}, \overline{y}) \to \mathrm{Con}\,\mathbf{F}(\overline{y}); \quad \Psi \mapsto \tau^{-1}[\Psi] = \Psi \cap \mathrm{F}(\overline{y})^2, \end{aligned}$$

yielding an *adjunction* $\langle \tau^*, \tau^{-1} \rangle$; that is, for any $\Theta \in \mathrm{Con}\,\mathbf{F}(\overline{y})$ and $\Psi \in \mathrm{Con}\,\mathbf{F}(\overline{x}, \overline{y})$,

$$\tau^*(\Theta) \subseteq \Psi \iff \Theta \subseteq \tau^{-1}(\Psi).$$

It follows, using Lemma 9, that $\mathcal{V}$ has the DIP if, and only if, for any sets $\overline{x}, \overline{y}, \overline{z}$ with inclusion maps $\tau_1, \tau_2, \tau_3, \tau_4$ between free algebras of $\mathcal{V}$, the following diagram commutes:

$$\begin{array}{ccc} \mathrm{Con}\,\mathbf{F}(\overline{x}, \overline{y}) & \xrightarrow{\tau_1^{-1}} & \mathrm{Con}\,\mathbf{F}(\overline{y}) \\ \Big\downarrow{\scriptstyle \tau_2^*} & & \Big\downarrow{\scriptstyle \tau_4^*} \\ \mathrm{Con}\,\mathbf{F}(\overline{x}, \overline{y}, \overline{z}) & \xrightarrow[\tau_3^{-1}]{} & \mathrm{Con}\,\mathbf{F}(\overline{y}, \overline{z}) \end{array}$$

That is, the DIP for $\mathcal{V}$ is equivalent to the 'Beck-Chevalley-like' condition $\tau_4^* \tau_1^{-1} = \tau_3^{-1} \tau_2^*$ for appropriate inclusion maps $\tau_1, \tau_2, \tau_3, \tau_4$ between free algebras of $\mathcal{V}$. For the relationship of this condition to a version of the interpolation property formulated in categorical logic, we refer the reader to [67, 64, 30].

We now establish a bridge theorem relating the AP and DIP, leaning heavily on the correspondence between the AP and RP provided by Theorem 21.

▶ **Theorem 27.** *Let $\mathcal{V}$ be any variety.*

(a) *If $\mathcal{V}$ has the amalgamation property, then it has the deductive interpolation property.*

(b) *If $\mathcal{V}$ has the deductive interpolation property and the extension property, then it has the amalgamation property.*

Proof. (a) Suppose that $\mathcal{V}$ has the AP and hence, by Theorem 21, the RP. Consider any $\Sigma \subseteq \mathrm{Eq}(\overline{x}, \overline{y})$ and $\varepsilon \in \mathrm{Eq}(\overline{y}, \overline{z})$ such that $\Sigma \vdash_{\mathcal{V}} \varepsilon$, and define $\Pi := \{\delta \in \mathrm{Eq}(\overline{y}) \mid \Sigma \vdash_{\mathcal{V}} \delta\}$. Clearly, $\Sigma \vdash_{\mathcal{V}} \Pi$ and, since $\Sigma \cup \Pi \vdash_{\mathcal{V}} \varepsilon$, the RP yields $\Pi \vdash_{\mathcal{V}} \varepsilon$.

(b) Suppose that $\mathcal{V}$ has the DIP and the EP. By Theorem 21, it suffices to show that $\mathcal{V}$ has the RP, so consider any $\Sigma \subseteq \mathrm{Eq}(\overline{x}, \overline{y})$ and $\Pi \subseteq \mathrm{Eq}(\overline{y}, \overline{z})$ satisfying $\Sigma \vdash_{\mathcal{V}} \delta \iff \Pi \vdash_{\mathcal{V}} \delta$ for all $\delta \in \mathrm{Eq}(\overline{y})$, and any $\varepsilon \in \mathrm{Eq}(\overline{y}, \overline{z})$ such that $\Sigma \cup \Pi \vdash_{\mathcal{V}} \varepsilon$. By the EP, there exists a $\Delta \subseteq \mathrm{Eq}(\overline{y}, \overline{z})$ such that $\Sigma \vdash_{\mathcal{V}} \Delta$ and $\Delta \cup \Pi \vdash_{\mathcal{V}} \varepsilon$. By the DIP, there exists a $\Gamma \subseteq \mathrm{Eq}(\overline{y})$ satisfying $\Sigma \vdash_{\mathcal{V}} \delta \iff \Gamma \vdash_{\mathcal{V}} \delta$ for all $\delta \in \mathrm{Eq}(\overline{y}, \overline{z})$. In particular, $\Sigma \vdash_{\mathcal{V}} \Gamma$, so, by assumption, $\Pi \vdash_{\mathcal{V}} \Gamma$. But also $\Gamma \vdash_{\mathcal{V}} \Delta$, so $\Pi \vdash_{\mathcal{V}} \Delta$. Hence $\Pi \vdash_{\mathcal{V}} \varepsilon$ as required. ◀

▶ **Example 28.** A variety of FL-algebras that has the CEP (e.g., any variety of $\mathrm{FL_e}$-algebras) has the AP if, and only if, it has the DIP, by Theorem 27. This bridge has been traversed successfully in both directions. For example, it was proved in [25] that continuum-many varieties of $\mathrm{FL_e}$-algebras have the AP and hence that continuum-many axiomatic extensions of $\mathrm{FL_e}$ have the DIP. The AP was established for the varieties of Abelian ℓ-groups in [65] and MV-algebras in [61], yielding the DIP for Abelian logic A and Łukasiewicz logic Ł, respectively, but in [57] the DIP was proved directly, yielding alternative proofs of the AP for these varieties. Note also that the failure of the AP for the variety of ℓ-groups, established in [65], has been used to prove that many other varieties of FL-algebras lack this property [32, 58]. ⌟

The conjunction of the DIP and EP for a variety $\mathcal{V}$ yields a strictly stronger property.

A variety $\mathcal{V}$ has the *Maehara interpolation property* (MIP) if for any $\Sigma \subseteq \mathrm{Eq}(\overline{x}, \overline{y})$ and $\Pi \cup \{\varepsilon\} \subseteq \mathrm{Eq}(\overline{y}, \overline{z})$ satisfying $\Sigma \cup \Pi \vdash_{\mathcal{V}} \varepsilon$, there exists a $\Delta \subseteq \mathrm{Eq}(\overline{y})$ satisfying $\Sigma \vdash_{\mathcal{V}} \Delta$ and $\Delta \cup \Pi \vdash_{\mathcal{V}} \varepsilon$.

The algebraic analogue of the MIP is a well-known categorical property.

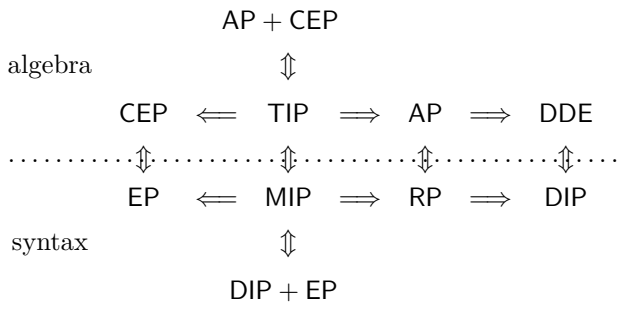


Figure 3 Bridges between algebraic and syntactic properties.

A class $\mathcal{K}$ of $\mathcal{L}$-algebras has the *transferable injections property* (TIP) if for any injective span $\langle \mathbf{A}, \mathbf{B}, \mathbf{C}, \varphi_B, \varphi_C \rangle$ in $\mathcal{K}$, there exist a $\mathbf{D} \in \mathcal{K}$, a homomorphism $\psi_B \colon \mathbf{B} \to \mathbf{D}$, and an embedding $\psi_C \colon \mathbf{C} \to \mathbf{D}$ such that $\psi_B \varphi_B = \psi_C \varphi_C$ (see Figure 1(iv)).

The equivalence of the MIP and TIP for varieties was proved in [3] (see also [72]), and their equivalence to the conjunctions of the DIP and CEP, and the RP and EP, were established in [9] and [63], respectively.

▶ **Theorem 29.** *The following statements are equivalent for any variety $\mathcal{V}$:*
(1) *$\mathcal{V}$ has the Maehara interpolation property.*
(2) *$\mathcal{V}$ has the deductive interpolation property and the extension property.*
(3) *$\mathcal{V}$ has the amalgamation property and the congruence extension property.*
(4) *$\mathcal{V}$ has the transferable injections property.*

This leaves open the problem of describing an algebraic property that corresponds directly to the DIP, to which (at least) two solutions may be found in the literature. A variety that admits free products has the DIP if, and only if, it has the 'flat amalgamation property' [2]. More generally, a variety has the DIP if, and only if, it has 'diamond diagrams for embeddings' (DDE), studied in [45] as the 'injective generalized amalgamation property' for substructural logics and considered in a more general algebraic setting in [7].

▶ **Example 30.** It was observed in [63] that the variety of monoids has the flat amalgamation property, and it is easy to see that the same is true of the variety of commutative monoids, yielding examples of varieties that have the DIP but not the AP (see Example 20) or CEP. ⌟

For the reader's convenience, some of the bridges between algebraic and syntactic properties presented (so far) in this chapter are displayed in Figure 3.

We conclude this section by considering a deductive version of uniform interpolation that is closely related to the DIP, referring the reader to Chapter 9 for a more nuanced account.

A variety $\mathcal{V}$ has the *right uniform deductive interpolation property* (RUDIP) if for any finite $\Sigma \subseteq \mathrm{Eq}(\overline{x}, \overline{y})$, there exists a finite $\Gamma \subseteq \mathrm{Eq}(\overline{y})$ satisfying $\Sigma \vdash_{\mathcal{V}} \varepsilon \iff \Gamma \vdash_{\mathcal{V}} \varepsilon$ for any $\varepsilon \in \mathrm{Eq}(\overline{y}, \overline{z})$.

Omitting the requirement that Γ be finite in this property yields the DIP. Conversely, the RUDIP may be viewed as the conjunction of the DIP together with a further well-studied algebraic property. Recall that an algebra $\mathbf{A}$ belonging to a variety $\mathcal{V}$ is *finitely presented in* $\mathcal{V}$ if it is isomorphic to $\mathbf{F}(\overline{x})/\Theta$ for some finite set $\overline{x}$ and finitely generated $\Theta \in \mathrm{Con}\,\mathbf{F}(\overline{x})$.

A variety $\mathcal{V}$ is *coherent* if every finitely generated subalgebra of a finitely presented member of $\mathcal{V}$ is finitely presented.

The notion of coherence originated in sheaf theory and has been studied broadly in the setting of groups, rings, modules, monoids, and other algebras (see, e.g., [17, 35, 49, 50]). It has also been considered from a model-theoretic perspective by Wheeler [70, 71], who proved that the first-order theory of a variety has a model completion if, and only if, it is coherent and has the AP and a further (rather complicated) property (see also [30, 34, 59]). The connection to the RUDIP is clarified by the following result.

▶ **Theorem 31** ([34, 50]). *The following statements are equivalent for any variety $\mathcal{V}$:*

(1) *$\mathcal{V}$ is coherent.*

(2) *For any finite sets $\overline{x}, \overline{y}$ and finitely generated congruence Θ of $\mathbf{F}(\overline{x}, \overline{y})$, the congruence $\Theta \cap \mathrm{F}(\overline{y})^2$ of $\mathbf{F}(\overline{y})$ is finitely generated.*

(3) *For any finite $\Sigma \subseteq \mathrm{Eq}(\overline{x}, \overline{y})$, there exists a finite $\Gamma \subseteq \mathrm{Eq}(\overline{y})$ satisfying $\Sigma \vdash_{\mathcal{V}} \varepsilon \iff \Gamma \vdash_{\mathcal{V}} \varepsilon$ for any $\varepsilon \in \mathrm{Eq}(\overline{y})$.*

▶ **Corollary 32.** *A variety $\mathcal{V}$ has the right uniform deductive interpolation property if, and only if, it is coherent and has the deductive interpolation property.*

The RUDIP and a left uniform deductive interpolation property were established for the variety of Heyting algebras in [68] and subsequently for many other varieties serving as algebraic semantics for propositional logics (see Chapter 9 for further details). Note, however, that although the variety of modal algebras has an implication-based uniform interpolation property, it is not coherent and therefore lacks the deductive version. Indeed, as shown in [49, 50] using a general criterion, broad families of varieties providing algebraic semantics for modal and substructural logics fail to be coherent, and hence do not have the RUDIP.

7 The Craig Interpolation Property

In this final section, we turn our attention to Craig interpolation, providing an equational formulation of this property for varieties of algebras with a lattice reduct and exploring its relationship to the DIP and various forms of amalgamation.

Let us assume that $\mathcal{L}$ is an algebraic language with binary operation symbols $\wedge$ and $\vee$ and, as before, at least one constant symbol, and write $s \leq t$ as shorthand for $s \wedge t \approx s$. We call an $\mathcal{L}$-algebra $\mathbf{A}$ *lattice-ordered* if its reduct $\langle A, \wedge, \vee \rangle$ is a lattice.

A variety $\mathcal{V}$ of lattice-ordered $\mathcal{L}$-algebras has the *Craig interpolation property* (CIP) if for any $\alpha \in \mathrm{Fm}(\overline{x}, \overline{y})$ and $\beta \in \mathrm{Fm}(\overline{y}, \overline{z})$ satisfying $\vdash_{\mathcal{V}} \alpha \leq \beta$, there exists a $\gamma \in \mathrm{Fm}(\overline{y})$ satisfying $\vdash_{\mathcal{V}} \alpha \leq \gamma$ and $\vdash_{\mathcal{V}} \gamma \leq \beta$.

Translating between consequence in propositional logics and their algebraic semantics as described in Section 2, this equational formulation of the CIP corresponds directly for superintuitionistic, modal, and substructural logics to familiar formulations of the CIP using an 'implication' connective $\rightarrow$ or $\backslash$ (see Chapter 3).

Relationships between the CIP and DIP and various amalgamation properties depend heavily on the existence of suitable local deduction theorems. It follows easily from the deduction theorem for superintuitionistic logics that the CIP and DIP, and hence also the AP, coincide for any variety of Heyting algebras. As remarked in the introduction, there are precisely eight such varieties, corresponding to eight superintuitionistic logics, that have these properties [52]. In general, however, neither the DIP nor the AP implies the CIP. For example, the variety of MV-algebras has the DIP and AP, but lacks the CIP; e.g., $x \wedge \neg x \leq y \vee \neg y$ is satisfied by all MV-algebras, but every variable-free formula is equivalent to $\bot$ or $\top$ in this variety, and both $x \wedge \neg x \leq \bot$ and $\top \leq y \vee \neg y$ fail in totally ordered MV-algebras with at least three elements. The failure of the CIP for many other varieties of semilinear FL-algebras can be established similarly [56]. In particular, only three varieties of BL-algebras have this property: Boolean algebras, Gödel algebras, and the variety generated by the three-element totally ordered Heyting algebra [60].

The following two propositions identify varieties providing algebraic semantics for broad families of substructural logics and modal logics for which the CIP implies the DIP and AP.

▶ **Proposition 33.** *If a variety of FL_e-algebras has the Craig interpolation property, then it has the deductive interpolation property and amalgamation property.*

Proof. Let $\mathcal{V}$ be a variety of $\mathrm{FL_e}$-algebras that has the CIP. It suffices to show that $\mathcal{V}$ has the DIP (see Example 28), so consider without loss of generality any $\alpha \in \mathrm{Fm}(\overline{x}, \overline{y})$ and $\beta \in \mathrm{Fm}(\overline{y}, \overline{z})$ satisfying $\{\mathrm{e} \leq \alpha\} \vdash_{\mathcal{V}} \mathrm{e} \leq \beta$. By the local deduction theorem for varieties of $\mathrm{FL_e}$-algebras (see Example 14), $\vdash_{\mathcal{V}} (\alpha \wedge \mathrm{e})^n \leq \beta$, for some $n \in \mathbb{N}$. Hence, by the CIP, there exists a $\gamma \in \mathrm{Fm}(\overline{y})$ satisfying $\vdash_{\mathcal{V}} (\alpha \wedge \mathrm{e})^n \leq \gamma$ and $\vdash_{\mathcal{V}} \gamma \leq \beta$. Using the local deduction theorem again twice, $\{\mathrm{e} \leq \alpha\} \vdash_{\mathcal{V}} \mathrm{e} \leq \gamma$ and $\{\mathrm{e} \leq \gamma\} \vdash_{\mathcal{V}} \mathrm{e} \leq b$. So $\mathcal{V}$ has the DIP. ◀

▶ **Proposition 34.** *If a variety of modal algebras has the Craig interpolation property, then it has the deductive interpolation property and amalgamation property.*

Proof. Analogous to the proof of Proposition 33. ◀

Establishing the CIP — e.g., using the proof-theoretic methods explained in Chapter 5 — is therefore one way of proving that a variety of $\mathrm{FL_e}$-algebras or modal algebras has the AP. Indeed, the only known proof that the variety of $\mathrm{FL_e}$-algebras has the AP takes this approach (see [58] or Chapter 5). Conversely, to prove the CIP for varieties of $\mathrm{FL_e}$-algebras or modal algebras using algebraic methods, we require a strengthening of the AP.

A class $\mathcal{K}$ of lattice-ordered $\mathcal{L}$-algebras has the *superamalgamation property* (super AP) if every doubly injective span $\langle \mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \varphi_1, \varphi_2 \rangle$ in $\mathcal{K}$ has an amalgam $\langle \mathbf{C}, \psi_1, \psi_2 \rangle$ in $\mathcal{K}$ satisfying for any $b_1 \in B_1$, $b_2 \in B_2$ and distinct $i, j \in \{1, 2\}$,

$$\psi_i(b_i) \leq \psi_j(b_j) \implies \psi_i(b_i) \leq \psi_i\varphi_i(a) = \psi_j\varphi_j(a) \leq \psi_j(b_j) \text{ for some } a \in A.$$

It is not hard to see that if a variety of lattice-ordered $\mathcal{L}$-algebras has the super AP, then it must also have the strong AP.

Bridge theorems between the CIP and super AP for varieties of modal algebras and $\mathrm{FL_e}$-algebras are established similarly to Theorem 27 relating the DIP and AP.

▶ **Theorem 35** ([53]). *A variety of modal algebras has the Craig interpolation property if, and only if, it has the superamalgamation property.*

▶ **Example 36.** Chapter 4 presents six quite different proofs that the variety of modal algebras — the algebraic semantics of the modal logic K — has the CIP and hence the super AP, DIP, and AP, including one that establishes the CIP via an analytic sequent calculus for K and another that establishes the super AP by constructing super-amalgams of spans of modal algebras. More generally, interpolation properties have been established for modal logics using a wide range of methods, both syntactic, using sometimes quite complex proof systems (see Chapter 5), and semantic (see, e.g., [54, 8, 28]). ⌟

▶ **Theorem 37** ([29]). *A variety of FL_e-algebras has the Craig interpolation property if, and only if, it has the superamalgamation property.*

▶ **Example 38.** A proof of the CIP for the variety of FL-algebras — the algebraic semantics of the full Lambek calculus FL — is obtained using an analytic sequent calculus (see, e.g., [29, 58]). This variety does not have the AP (see [37], also for failures of the AP for related varieties), although the question of whether it has the DIP is still open. The varieties of ℓ-groups and semilinear residuated lattices do not have the CIP or the AP (see Example 28), but the status of the DIP is open also for these cases. ⌟

Table 1 displays the status of the CEP, CIP, DIP, and AP for some of the varieties featured in this chapter (where 'n/a' stands for 'not applicable' and '?' stands for 'open problem').[5]

[5] Note that although Abelian groups lack a lattice reduct, they satisfy the Craig interpolation property with respect to an implication defined by $a \to b := a^{-1}b$.

Table 1 Interpolation and amalgamation properties for a selection of varieties.

Variety	CEP	CIP	DIP	AP
Boolean algebras	yes	yes	yes	yes
modal algebras	yes	yes	yes	yes
Heyting algebras	yes	yes	yes	yes
Gödel algebras	yes	yes	yes	yes
MV-algebras	yes	no	yes	yes
BL-algebras	yes	no	yes	yes
FL-algebras	no	yes	?	no
$\mathrm{FL_e}$-algebras	yes	yes	yes	yes
semilinear FL-algebras	no	no	?	no
semilinear $\mathrm{FL_e}$-algebras	yes	no	no	no
ℓ-groups	no	no	?	no
Abelian ℓ-groups	yes	no	yes	yes
bounded lattices	no	yes	yes	yes
bounded distributive lattices	yes	yes	yes	yes
commutative monoids	no	n/a	yes	no
Abelian groups	yes	n/a	yes	yes

Acknowledgments

I would like to thank Wesley Fussner, Sam van Gool, Isabel Hortelano-Martin, and Simon Santschi for their careful reading of previous versions of this chapter and helpful feedback. I also acknowledge support from the Swiss National Science Foundation (SNSF) grant no. 200021_215157.

References

1 M.E. Anderson and T.H. Feil. *Lattice-Ordered Groups: An Introduction*. Springer, 1988.
2 P.D. Bacsich. Amalgamation properties and interpolation theorems for equational theories. *Algebra Universalis*, 5:45–55, 1975.
3 B. Banaschewski. Injectivity and essential extensions in equational classes of algebras. In *Proc. Conf. on Universal Algebra*, pages 131–147, 1969.
4 W.J. Blok and E. Hoogland. The Beth property in algebraic logic. *Studia Logica*, 83:49–90, 2006.
5 W.J. Blok and D. Pigozzi. *Algebraizable Logics*, volume 396 of *Memoirs of the American Mathematical Society*. American Mathematical Society, 1989.
6 S. Burris and H.P. Sankappanavar. *A Course in Universal Algebra*. Springer, 1981.
7 L.M. Cabrer and J. Gil-Férez. Leibniz interpolation properties. *Ann. Pure Appl. Logic*, 165(4):933–962, 2014.
8 A. Chagrov and M. Zakharyaschev. *Modal Logic*. Oxford University Press, 1996.

9 J. Czelakowski. Sentential logics and Maehara interpolation property. *Studia Logica*, 44(3):265–283, 1985.

10 J. Czelakowski and W. Dziobiak. Congruence distributive quasivarieties whose finitely subdirectly irreducible members form a universal class. *Algebra Universalis*, 27:128–149, 1990.

11 J. Czelakowski and D. Pigozzi. Amalgamation and interpolation in abstract algebraic logic. In *Models, Algebras, and Proofs*, volume 203 of *Lecture Notes in Pure and Applied Mathematics*, pages 187–265. Marcel Dekker, Inc., 1999.

12 A. Daigneault. Freedom in polyadic algebras and two theorems of Beth and Craig. *Mich. Math. J.*, 11:129–135, 1964.

13 B.A. Davey. Weak injectivity and congruence extension in congruence-distributive equational classes. *Canad. J. Math.*, 29(3):449–459, 1977.

14 A. Day. Varieties of Heyting algebras II (amalgamation and injectivity). Unpublished manuscript, undated.

15 A. Day and J. Ježek. The amalgamation property for varieties of lattices. *Trans. Amer. Math. Soc.*, 286:251–256, 1984.

16 A. Di Nola and A. Lettieri. One chain generated varieties of MV-algebras. *J. Algebra*, 225(2):667–697, 2000.

17 P. Eklof and G. Sabbagh. Model-completions and modules. *Ann. Math. Logic*, 2(3):251–295, 1970.

18 I. Fleischer. Amalgamation for semilattices. *Algebra Universalis*, 6:411–412, 1976.

19 I. Fleischer. Relatively pseudocomplemented semilattices amalgamate strongly. *Algebra Universalis*, 11:130–132, 1980.

20 J.M. Font. *Abstract algebraic logic: An introductory textbook*. College Publications, 2016.

21 R. Fraïssé. Sur l'extension aux relations de quelques propriétés des ordres. *Annales Scientifiques de l'École Normale Supérieure*, 71:363–388, 1954.

22 W. Fussner and N. Galatos. Semiconic idempotent logic II: Beth definability and deductive interpolation. *Ann. Pure Appl. Logic*, 176(3):103528, 2025.

23 W. Fussner and G. Metcalfe. Transfer theorems for finitely subdirectly irreducible algebras. *J. Algebra*, 640:1–20, 2024.

24 W. Fussner, G. Metcalfe, and S. Santschi. Interpolation and the exchange rule. *J. Symbolic Logic*. Published online 2026:1–29. doi:10.1017/jsl.2026.10198.

25 W. Fussner and S. Santschi. Interpolation in linear logic and related systems. *ACM Trans. Comput. Logic*, 25(4):1–19, 2024.

26 W. Fussner and S. Santschi. Amalgamation in semilinear residuated lattices. *Studia Logica*, 114: 103–144, 2026.

27 W. Fussner and S. Santschi. Interpolation in Hájek's basic logic. *Ann. Pure Appl. Logic*, 176(9):103615, 2025.

28 D. Gabbay and L. Maksimova. *Interpolation and Definability: Modal and Intuitionistic Logic*, volume 46 of *Oxford Logic Guides*. Oxford University Press, 2005.

29 N. Galatos, P. Jipsen, T. Kowalski, and H. Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Elsevier, 2007.

30 S. Ghilardi and M. Zawadowski. *Sheaves, Games, and Model Completions: A Categorical Approach to Nonclassical Propositional Logics*. Springer, 2002.

31 J. Gil-Férez, P. Jipsen, and G. Metcalfe. Structure theorems for idempotent residuated lattices. *Algebra Universalis*, 81(2), 2020.

32 J. Gil-Férez, A. Ledda, and C. Tsinakis. The failure of the amalgamation property for semilinear varieties of residuated lattices. *Math. Slovaca*, 65(4):817–828, 2015.

33 V. Giustarini and S. Ugolini. Blockwise gluings and amalgamation failures in integral residuated lattices. Manuscript, 2025. URL: `https://arxiv.org/abs/2408.17400`, `arXiv:2408.17400`.

34 S. van Gool, G. Metcalfe, and C. Tsinakis. Uniform interpolation and compact congruences. *Ann. Pure Appl. Logic*, 168:1827–1948, 2017.

35 V. Gould. Coherent monoids. *J. Austral. Math. Soc. Ser. A*, 53(2):166–182, 1992.

36 G. Grätzer and H. Lakser. The structure of pseudocomplemented distributive lattices II: Congruence extension and amalgamation. *Trans. Amer. Math. Soc.*, 156:343–358, 1971.

37 P. Jipsen and S. Santschi. Residuated lattices do not have the amalgamation property, 2025. Under review. URL: `https://arxiv.org/abs/2504.16111`, `arXiv:2504.16111`.

38 B. Jónsson. Universal relational structures. *Math. Scand.*, 4:193–208, 1956.

39 B. Jónsson. Homogeneous universal relational structures. *Math. Scand.*, 8:137–142, 1960.

40 B. Jónsson. Sublattices of a free lattice. *Canad. J. Math.*, 13:146–157, 1961.

41 B. Jónsson. Algebraic extensions of relational systems. *Math. Scand.*, 11:179–205, 1962.

42 B. Jónsson. Extensions of relational structures. In *Proc. International Symposium on the Theory of Models*, pages 146–157. North Holland, 1965.

43 B. Jónsson. Algebras whose congruence lattices are distributive. *Math. Scand.*, 21:110–121 (1968), 1967.

44 K.A. Kearnes. On the relationship between AP, RS and CEP. *Proc. Amer. Math. Soc.*, 105(4):827–839, 1989.

45 H. Kihara and H. Ono. Interpolation properties, Beth definability properties and amalgamation properties for substructural logics. *J. Logic Comput.*, 20(4):823–875, 2010.

46 N. Kimura. *On Semigroups*. PhD thesis, Tulane University, 1957.

47 E.W. Kiss. Injectivity and related concepts in modular varieties II: The congruence extension property. *Bull. Aust. Math. Soc.*, 32:45–53, 1985.

48 E.W. Kiss, L. Márki, P. Pröhle, and W. Tholen. Categorical algebraic properties. A compendium of amalgamation, congruence extension, epimorphisms, residual smallness, and injectivity. *Stud. Sci. Math. Hung.*, 18:79–141, 1983.

49 T. Kowalski and G. Metcalfe. Coherence in modal logic. In *Proc. AiML'18*, pages 451–465. College Publications, 2018.

50 T. Kowalski and G. Metcalfe. Uniform interpolation and coherence. *Ann. Pure Appl. Logic*, 170(7):825–841, 2019.

51 J. Madarász. Interpolation and amalgamation: Pushing the limits. Part I. *Studia Logica*, 61:311–345, 1998.

52 L.L. Maksimova. Craig's theorem in superintuitionistic logics and amalgamable varieties of pseudo-Boolean algebras. *Algebra i Logika*, 16:643–681, 1977.

53 L.L. Maksimova. Interpolation theorems in modal logics and amalgamable varieties of topological Boolean algebras. *Algebra i Logika*, 18(5):556–586, 1979.

54 L.L. Maksimova. Amalgamation and interpolation in normal modal logics. *Studia Logica*, 50:457–471, 1991.

55 E. Marchioni. Amalgamation through quantifier elimination for varieties of commutative residuated lattices. *Arch. Math. Logic*, 51(1–2):15–34, 2012.

56 E. Marchioni and G. Metcalfe. Craig interpolation for semilinear substructural logics. *Math. Log. Quart.*, 58(6):468–481, 2012.

57 G. Metcalfe, F. Montagna, and C. Tsinakis. Amalgamation and interpolation in ordered algebras. *J. Algebra*, 402:21–82, 2014.

58 G. Metcalfe, F. Paoli, and C. Tsinakis. *Residuated Structures in Algebra and Logic*, volume 277 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2023.

59 G. Metcalfe and L. Reggio. Model completions for universal classes of algebras: necessary and sufficient conditions. *J. Symbolic Logic*, 88(1):381–417, 2023.

60 F. Montagna. Interpolation and Beth's property in many-valued logic: a semantic investigation. *Ann. Pure Appl. Logic*, 141:148–179, 2006.

61 D. Mundici. Free products in the category of abelian ℓ-groups with strong unit. *J. Algebra*, 113:89–109, 1988.

62 I. Németi. Beth definability is equivalent with surjectiveness of epis in general algebraic logic. Technical report, Math. Inst. Hungar. Acad. Sci., Budapest, 1984.

63 H. Ono. Interpolation and the Robinson property for logics not closed under the Boolean operations. *Algebra Universalis*, 23:111–122, 1986.

64 D. Pavlović. Categorical interpolation: Descent and the Beck-Chevalley condition without direct images. In *Category Theory*, volume 1488 of *Lecture Notes in Math.*, pages 306–325. Springer, 1991.

65 K.R. Pierce. Amalgamations of lattice-ordered groups. *Trans. Amer. Math. Soc.*, 172:249–260, 1972.

66 D. Pigozzi. Amalgamation, congruence-extension, and interpolation properties in algebras. *Algebra Universalis*, 1:269–349, 1972.

67 A.M. Pitts. Amalgamation and interpolation in the category of Heyting algebras. *J. Pure Appl. Alg.*, 29:155–165, 1983.

68 A.M. Pitts. On an interpretation of second-order quantification in first-order intuitionistic propositional logic. *J. Symbolic Logic*, 57:33–52, 1992.

69 O. Schreier. Die Untergruppen der freien gruppen. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 5(1):161–183, 1927.

70 W.H. Wheeler. Model-companions and definability in existentially complete structures. *Israel J. Math.*, 25(3):305–330, 1976.

71 W.H. Wheeler. A characterization of companionable, universal theories. *J. Symbolic Logic*, 43(3):402–429, 1978.

72 A. Wroński. On a form of equational interpolation property. In G. Dorn and P. Weingartner, editors, *Foundations of Logic and Linguistics: Problems and their Solutions*, pages 23–29. Springer, 1985.

Chapter 9

Uniform Interpolation

Sam van Gool
Université Paris-Saclay, ENS Paris-Saclay, CNRS, LMF, France

Abstract

Uniform interpolation is a strengthening of interpolation that holds for certain propositional logics. The starting point of this chapter is a theorem of A. Pitts, which shows that uniform interpolation holds for intuitionistic propositional logic. We outline how this theorem may be proved semantically via the definability of bisimulation quantifiers, and how it generalizes to an open mapping theorem between Esakia spaces. We also discuss connections between uniform interpolation and research in categorical logic, algebra, and model theory.

Contents

1 Introduction 302
2 Uniform Interpolants via Bisimulation Quantifiers 305
2.1 Models and Bisimilarity 305
2.2 Bisimulation Quantifiers 306
2.3 Definability via Bounded Bisimulations 307
2.4 Extensions of the Semantic Method to Other Logics 310
3 Uniform Interpolants via Adjunction and Topology 312
3.1 Uniform Interpolants as Adjoints 312
3.2 Esakia Duality and Open Mappings 315
3.3 Connections to Categorical Logic 316
4 Model-complete Theories from Uniform Interpolants 319
5 Syntactic Methods, Further Results and Connections 322
References 324



© Sam van Gool;
licensed under Creative Commons License CC BY 4.0
In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 9, pages 301–328. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.i

1 Introduction

Let $\psi(\overline{p},\overline{q})$ and $\theta(\overline{q},\overline{r})$ be propositional formulas and suppose that $\psi \to \theta$ is a tautology of classical logic. Define the formula

$$\phi(\overline{q}) \stackrel{\text{def}}{=} \bigvee \{\psi(\overline{p}/\overline{b},\overline{q}) \mid \overline{b} \in \{\bot,\top\}^{\overline{p}}\}, \tag{1}$$

that is, ϕ is the disjunction of all possible variants of ψ obtained by substituting some vector of truth values for the propositional variables $\overline{p}$. As also explained in Chapter 1, Section 7, observe that ϕ is an interpolant for the entailment $\psi \vdash \theta$. Indeed, $\psi \vdash \phi$ follows from the definition of ϕ, and $\phi \vdash \theta$ uses the assumption that $\psi \vdash \theta$. Notice, moreover, that the interpolant ϕ defined here does not at all depend on the consequent formula θ, but only on the antecedent formula ϕ and on the subset $\overline{p}$ of variables 'to be eliminated' from ψ. In other words, ϕ is *uniformly* an interpolant on the right of ψ: it interpolates any consequence θ' of ψ that does not contain any of the variables in $\overline{p}$. The formula ϕ of (1) is therefore called a *right uniform interpolant* for ψ with respect to the set of eliminated variables $\overline{p}$.[1] Symmetrically, the expression $\bigwedge\{\theta(\overline{q}/\overline{b},\overline{r}) \mid \overline{b} \in \{\bot,\top\}^{\overline{q}}\}$ defines a *left uniform interpolant* for θ with respect to $\overline{p}$ in classical logic.

What other propositional logics have uniform interpolants? One may show that an entailment relation $\vdash$ has right uniform interpolants if it has interpolants and is locally finite, by which we mean that there are finitely many equivalence classes of formulas in a finite set of variables. However, the local finiteness is by no means a *necessary* condition for a propositional logic to have (right) uniform interpolants. Indeed, the following surprising theorem, which inspired much of the research surveyed in this chapter, is an instance of a non-locally-finite logic with uniform interpolation.

▶ **Theorem 1** (Pitts, [53]). *In intuitionistic logic, any propositional formula has both left and right uniform interpolants, with respect to any set of propositional variables.*

To give the general definition of uniform interpolation, we place ourselves in the context of a set $\mathcal{L}$ of formulas in a propositional logic, and we assume given a binary *entailment relation* $\vdash$ between formulas in $\mathcal{L}$, which we will always assume is reflexive and transitive.

▶ **Definition 2.** *Let $\vdash$ be an entailment relation, ϕ a formula, and p a propositional variable. We call a formula $\mathsf{E}_p(\phi)$ a* right uniform interpolant *for ϕ with respect to p if the following hold:*

1. *the formula $\mathsf{E}_p(\phi)$ only contains variables that occur in ϕ, but not p,*
2. *the entailment $\phi \vdash \mathsf{E}_p(\phi)$ holds, and,*

[1] We warn the reader of a possible source of confusion: Chapter 1, following part of the literature, calls this a right uniform $\overline{q}$-interpolant, thus referring in the name to the *remaining* variables, rather than to the eliminated ones, as is done here, and also, for instance, in Chapter 5.

3. *for any p-free formula θ, if $\phi \vdash \theta$, then $\mathsf{E}_p(\phi) \vdash \theta$.*

Similarly, we call a formula $\mathsf{A}_p(\phi)$ a left uniform interpolant *for ϕ with respect to p if the following hold:*

1. *the formula $\mathsf{A}_p(\phi)$ only contains variables that occur in ϕ, but not p,*

2. *the entailment $\mathsf{A}_p(\phi) \vdash \phi$ holds, and,*

3. *for any p-free formula ψ, if $\psi \vdash \phi$, then $\psi \vdash \mathsf{A}_p(\phi)$.*

We say that the relation $\vdash$ has (left and right) uniform interpolants *if such formulas $\mathsf{E}_p(\phi)$ and $\mathsf{A}_p(\phi)$ exist for all formulas ϕ and propositional variables p.*

We immediately make a few remarks about Definition 2. First, we have only given the definition with respect to a single variable, but the uniform interpolants with respect to sets of variables can be obtained by iterating the single-variable constructs. As a consequence, if an entailment relation has uniform interpolants with respect to single variables, then it has uniform interpolants with respect to any set of variables.

Second, any formula ϕ has, up to equivalence, at most one right uniform interpolant, and at most one left uniform interpolant. Here, two formulas ϕ and ϕ' are *equivalent* in an entailment relation $\vdash$ if both $\phi \vdash \phi'$ and $\phi' \vdash \phi$ hold. With a slight abuse of notation, we therefore often speak of 'the' right uniform interpolant and 'the' left uniform interpolant of ϕ.

Third, we have defined uniform interpolants with respect to an entailment relation $\vdash$, but there exists a subtly different definition in the literature. Indeed, let $\to$ be a binary symbol in the language under consideration. One may then also reasonably define[2] *implication-based uniform interpolants* by replacing, in Definition 2, all assertions of the form '$\alpha \vdash \beta$' by '$\top \vdash \alpha \to \beta$', where $\top$ is a fixed constant symbol in the language, suitably connected to the operation $\to$. Let us say that the entailment relation $\vdash$ has the *deduction property* if, for all formulas α and β, $\alpha \vdash \beta$ is equivalent to $\top \vdash \alpha \to \beta$. If the entailment relation $\vdash$ does *not* have this deduction property, then the notions of uniform interpolant in the sense of Definition 2 and the implication-based uniform interpolant are not the same in general. Indeed, when $\vdash$ is the global consequence relation for the modal logic **K**, uniform interpolants in the sense of Definition 2 do not always exist, while implication-based uniform interpolants do exist [20, Sec. 6]; we come back to this in Remark 11 below. In light of this, the uniform interpolants of Definition 2 are sometimes called *deductive* uniform interpolants for emphasis.

▶ **Convention.** In this chapter we adopt the convention that the phrase 'uniform interpolant' always means 'uniform interpolant in the sense of Definition 2', and we explicitly mention 'implication-based uniform interpolant' when needed.

Finally, note that the three conditions (1)–(3) defining the right and left uniform interpolants, respectively, correspond to rules for second order existential and universal quantifiers;

[2] For instance, this is how uniform interpolation is defined in Definition 24 in Section 3.3 of Chapter 5.

this explains the suggestive notations $\mathsf{E}_p(\phi)$ and $\mathsf{A}_p(\phi)$. The connection between uniform interpolants and higher order quantifiers was central to Pitts' original paper [53, §4], and was further elaborated in the algebraic, categorical, and model-theoretic approaches to uniform interpolation, in particular by S. Ghilardi & M. Zawadowski [27], as we will discuss in depth in Sections 3 and 4 below. Note immediately, however, that the interpretation of second order quantifiers using the uniform interpolants E_p and A_p is not *conservative* [53, Exa. 10]; in other words, uniform interpolants provide *a* possible interpretation of second order quantifiers, but moreover satisfy specific properties that generic second order quantifiers do not.

Proof methods for uniform interpolation can be divided, roughly, into two strands: one is syntactic and relies on the existence of a well-behaved sequent calculus for the entailment relation $\vdash$. This is how uniform interpolation was established for intuitionistic entailment in Pitts' original proof [53]. Pitts' proof relies on a strongly terminating proof calculus for intuitionistic propositional logic, and has been the subject of intensive proof-theoretic investigations and extensions to other logics; see, for instance, Section 3.3 in Chapter 5 and the references therein; we will also make some more remarks on this method in Section 5 of this chapter. The other proof method, which will be our focus in this chapter, starts from a *semantic* characterization of the entailment relation $\vdash$, typically in terms of a consequence relation on Kripke models. One then shows that *bisimulation quantifiers* are definable in the propositional language, and deduces from this the existence of implication-based uniform interpolants. This approach to proving implication-based uniform interpolation was developed soon after [53] as an alternative proof for Theorem 1 [24], and was also shown to apply to modal logics such as Gödel-Löb logic **GL** [61, Cor. 2.13], modal logic **K** [20, Thm 6.1] and Grzegorczyk logic **S4Grz** [65, Sec. 8].

The rest of this chapter is structured as follows. In Section 2, we outline the proof of uniform interpolation for intuitionistic logic via bisimulation quantifiers. We then make a link with adjoints between free algebras, and with Esakia duality, in Section 3. In the same section, we explain the relationship between uniform interpolation and an open question on elementary toposes and Heyting algebras, which was Pitts' original motivation for proving his theorem. In Section 4, we expose the relationship between uniform interpolants and quantifier elimination in first order theories of classes of algebras associated to propositional logics. In Section 5, we point out some further connections, in particular to syntactic methods, and give a table, Table 1, summarizing what is (not) known about uniform interpolation for some of the most commonly considered propositional logics.

Note that our focus in this chapter is on uniform interpolation for *propositional* logics. In the classical setting, first order logic does not have uniform interpolants, see for instance Theorem 2.17 in Chapter 2. There are substructural predicate logics with uniform interpolation [3]. We do not know at present whether or not intuitionistic predicate logic has uniform interpolation. Further open problems about uniform interpolation can also be found by looking at the open cells of Table 1.

2 Uniform Interpolants via Bisimulation Quantifiers

In this section, we explain how to prove Theorem 1 using the idea of *bisimulation quantifiers*. This proof of Theorem 1 is essentially due to [24], which was inspired by a similar technique developed for **GL** in [61]. While [24] phrases the argument in terms of sheaf representations of finitely presented Heyting algebras, we here present the ideas in more elementary terms, following [65]; for more about the connection, see Subsection 2.4 below.

We focus on the case of intuitionistic propositional logic for our presentation here, but similar ideas apply for proving implication-based uniform interpolation in modal logics; also see Subsection 2.4 for more about this.

2.1 Models and Bisimilarity

For $\overline{p}$ a set of propositional variables, an (intuitionistic) *Kripke model over* $\overline{p}$, or $\overline{p}$*-model*, is a tuple $M = (W, \preceq, \pi)$, where W is a set of *nodes*, $\preceq$ is a reflexive and transitive binary relation on W, and π is a function that assigns to every variable $p \in \overline{p}$ an upwards closed subset $\pi(p)$ of W; such a model is called *finite* if its set of nodes is finite. A *pointed* $\overline{p}$*-model* is a tuple (M, w), where $M = (W, \preceq, \pi)$ is a $\overline{p}$-model and $w \in W$. For any pointed $\overline{p}$-model (M, w) and any formula $\phi(\overline{p})$, we write $M, w \Vdash \phi$ if ϕ is *forced* at node w. This forcing relation $\Vdash$ is defined inductively, as usual, for any pointed $\overline{p}$-model (M, w), by the following five clauses, where p is any variable in $\overline{p}$ and ϕ_1, ϕ_2 are any formulas with variables in $\overline{p}$:

1. $M, w \Vdash p \overset{\text{def}}{\iff} w \in \pi(p)$;
2. $M, w \Vdash \phi_1 \wedge \phi_2 \overset{\text{def}}{\iff} M, w \Vdash \phi_1$ and $M, w \Vdash \phi_2$;
3. $M, w \Vdash \phi_1 \vee \phi_2 \overset{\text{def}}{\iff} M, w \Vdash \phi_1$ or $M, w \Vdash \phi_2$;
4. $M, w \Vdash \phi_1 \to \phi_2 \overset{\text{def}}{\iff}$ for any node v in M such that $w \preceq v$, if $M, v \Vdash \phi_1$ then $M, v \Vdash \phi_2$;
5. $M, w \nVdash \bot$.

Intuitionistic entailment is *sound and complete* with respect to finite Kripke models. More precisely, if $\phi(\overline{p})$ and $\psi(\overline{p})$ are formulas, then the intuitionistic entailment $\phi \vdash \psi$ holds if, and only if, for any finite pointed $\overline{p}$-model (M, w), if $M, w \Vdash \phi$ then $M, w \Vdash \psi$. For a proof of this foundational equivalence, see, e.g., [9, Thm. 2.43]. In light of this equivalence, when $\phi(\overline{p})$ is a formula and $\overline{q}$ is any set of variables with $\overline{p} \subseteq \overline{q}$, we define

$$[\![\phi]\!]_{\overline{q}} := \{(M, w) \text{ a finite pointed } \overline{q}\text{-model} \mid (M, w) \Vdash \phi\}$$

and we note that soundness and completeness means that $\phi \vdash \psi$ is equivalent to $[\![\phi]\!]_{\overline{q}} \subseteq [\![\psi]\!]_{\overline{q}}$.

We will make use of the key semantic notion of *bisimilarity*, a combinatorial method for determining when two nodes in Kripke models have the same 'behavior'. In our setting, having the same behavior means that the two nodes force the same intuitionistic propositional formulas. In order to prove the existence of uniform interpolants with respect to a propositional variable p, we will in particular be interested in nodes that have the same behavior *for a given set of variables*. The following notion of $\overline{q}$-bisimilarity captures this idea.

▶ **Definition 3.** *Let $M = (W, \preceq, \pi)$ and $M' = (W', \preceq', \pi')$ be $\overline{p}$-models, and let $\overline{q} \subseteq \overline{p}$. A* $\overline{q}$-bisimulation *between M and M' is a relation $Z \subseteq W \times W'$ satisfying the following three properties, for any $(w, w') \in Z$:*

1. *for any $q \in \overline{q}$, we have $w \in \pi(q)$ if, and only if, $w' \in \pi'(q)$;*
2. *for any $v \in W$, if $w \preceq v$, then there exists $v' \in W'$ such that $(v, v') \in Z$ and $w' \preceq' v'$;*
3. *for any $v' \in W'$, if $w' \preceq' v'$, then there exists $v \in W$ such that $(v, v') \in Z$ and $w \preceq v$.*

We say that a pair of nodes $(w, w') \in W \times W'$ is $\overline{q}$-bisimilar *if there exists a $\overline{q}$-bisimulation between M and M' which contains the pair (w, w'). In this case, we write $(M, w) \sim_{\overline{q}} (M', w')$.*[3]

A fundamental fact is that $\overline{q}$-bisimilar nodes force exactly the same formulas in variables $\overline{q}$:

▶ **Lemma 4.** *Let (M, w) and (M', w') be pointed $\overline{p}$-models, let $\overline{q} \subseteq \overline{p}$, and suppose that $(M, w) \sim_{\overline{q}} (M', w')$. For any formula $\phi(\overline{q})$, we have $(M, w) \Vdash \phi$ if, and only if, $(M', w') \Vdash \phi$.*

Proof. By induction on the construction of the formula ϕ. ◀

2.2 Bisimulation Quantifiers

The *existential bisimulation quantifier* is given by the following semantic construction. Let $p, \overline{q}$ be a set of variables and let $\mathcal{K}$ be a class of finite pointed $(p, \overline{q})$-models. We define the class of finite pointed $(p, \overline{q})$-models

$$\mathcal{E}_p(\mathcal{K}) := \{(M, w) \mid \text{there exists } (M', w') \text{ such that } (M, w) \sim_{\overline{q}} (M', w') \text{ and } (M', w') \in \mathcal{K}\}.$$

Similarly, the *universal bisimulation quantifier* is defined as the class of finite pointed $(p, \overline{q})$-models

$$\mathcal{A}_p(\mathcal{K}) := \{(M, w) \mid \text{for all } (M', w') \text{ such that } (M, w) \sim_{\overline{q}} (M', w'), \text{ we have } (M', w') \in \mathcal{K}\}.$$

The following proposition shows that *definability* of bisimulation quantifiers implies the existence of uniform interpolants. We give a simple proof in which we already assume the usual, non-uniform Craig interpolation theorem for intuitionistic propositional logic. It is possible to avoid the appeal to this theorem, as we will explain after the proof.

▶ **Proposition 5.** *Let $\phi(p, \overline{q})$, $\varepsilon(\overline{q})$, and $\alpha(\overline{q})$ be formulas.*

1. *If $[\![\varepsilon]\!]_{p,\overline{q}} = \mathcal{E}_p([\![\phi]\!]_{p,\overline{q}})$, then ε is a right uniform interpolant for ϕ with respect to p.*
2. *If $[\![\alpha]\!]_{p,\overline{q}} = \mathcal{A}_p([\![\phi]\!]_{p,\overline{q}})$, then α is a left uniform interpolant for ϕ with respect to p.*

[3] We warn the reader that, in some sources, the relation that we denote by $\sim_{\overline{q}}$ would instead be denoted by $\sim_{\overline{r}}$, where $\overline{r} = \overline{p} \setminus \overline{q}$, thus emphasizing the set of propositional variables that have been 'forgotten', rather than the set of variables that have been 'kept'.

Proof. (1) Assume that $[\![\varepsilon]\!]_{p,\overline{q}} = \mathcal{E}_p([\![\phi]\!]_{p,\overline{q}})$. Note that $[\![\phi]\!]_{p,\overline{q}} \subseteq \mathcal{E}_p([\![\phi]\!]_{p,\overline{q}})$, since any model is p-bisimilar to itself. Thus, $[\![\phi]\!]_{p,q} \subseteq [\![\varepsilon]\!]_{p,\overline{q}}$, and therefore $\phi \vdash \varepsilon$ by completeness. Now let $\theta(\overline{q},\overline{r})$ be any formula such that $\phi \vdash \theta$. By the Craig interpolation theorem for intuitionistic logic, pick $\theta'(\overline{q})$ such that $\phi \vdash \theta'$ and $\theta' \vdash \theta$. We will prove that $\varepsilon \vdash \theta'$, from which $\varepsilon \vdash \theta$ follows by transitivity of $\vdash$. To establish $\varepsilon \vdash \theta'$, by completeness it suffices to show that $[\![\varepsilon]\!]_{p,\overline{q}} \subseteq [\![\theta']\!]_{p,\overline{q}}$. Let $(M,w) \in [\![\varepsilon]\!]_{p,\overline{q}}$ be arbitrary. Since $[\![\varepsilon]\!]_{p,\overline{q}} = \mathcal{E}_p([\![\phi]\!]_{p,\overline{q}})$, pick $(M',w') \sim_{\overline{q}} (M,w)$ such that $M',w' \Vdash \phi$. Since $\phi \vdash \theta'$, by soundness we have $M',w' \Vdash \theta'$. By Lemma 4, we have $M,w \Vdash \theta'$, as required. The proof of (2) is analogous. ◀

The converse statements of (1) and (2) in Proposition 5 are also true, as we will prove in Corollary 10 below. We also remark that the appeal to Craig interpolation theorem in the proof of Proposition 5 could be avoided, by modifying the statement in item (1) to: if, for any finite set of variables $\overline{r}$, $[\![\varepsilon]\!]_{p,\overline{q},\overline{r}} = \mathcal{E}_p([\![\phi]\!]_{p,\overline{q},\overline{r}})$, then ε is a right uniform interpolant for ϕ with respect to p. The proof of this modified statement is very similar to the proof of Proposition 5(1), but does not require the Craig interpolation theorem. A similar remark holds for item (2). Using this modified version of Proposition 5, one may slightly strengthen the proof outlined below and give a proof of the uniform interpolation theorem that does not already rely on the Craig interpolation theorem.

A more abstract point of view on bisimulation quantifiers $\mathcal{E}_p$ and $\mathcal{A}_p$ is that they are semantic realizations of left and right *adjoints*, respectively, of certain homomorphisms between free Heyting algebras. Proposition 5 then says that if these adjoints are definable in the logic, then they give uniform interpolants. We will expand upon this idea in Section 3 below.

2.3 Definability via Bounded Bisimulations

In light of Proposition 5, we now seek to prove that the existential and universal bisimulation quantifiers are definable in intuitionistic propositional logic. To do so, the main technical tool is that of *bounded bisimulation* [65] or, equivalently, the *bisimulation games* of [24].

▶ **Definition 6.** *Let $M = (W, \preceq, \pi)$ and $M' = (W', \preceq', \pi')$ be $\overline{q}$-models. Let $Z_0, \ldots, Z_n \subseteq W \times W'$ be a finite sequence of binary relations. We call $(Z_i)_{i=0}^n$ a* bounded bisimulation *of depth n if it satisfies the following three properties:*

1. *for any $(w,w') \in \bigcup_{i=0}^n Z_i$ and $q \in \overline{q}$, $w \in \pi(q)$ if, and only if, $w' \in \pi'(q)$,*

and, for each $0 \leq i < n$ and $(w,w') \in Z_{i+1}$,

2. *for any $v \in W$, if $w \preceq v$, then there exists $v' \in W'$ such that $(v,v') \in Z_i$ and $w' \preceq' v'$;*

3. *for any $v' \in W'$, if $w' \preceq' v'$, then there exists $w \in W$ such that $(w,w') \in Z_i$ and $w \preceq v$.*

*We say that M,w and M',w' are n-*bisimilar *if there exists a bounded bisimulation of depth n which relates w to w'. In this case, we write $M,w \sim_{n,\overline{q}} M',w'$.*

Note that if Z is a bisimulation, then the constant sequence $Z, \ldots, Z$ gives a bounded bisimulation of arbitrary depth. Thus, $M,w \sim_{\overline{q}} M',w'$ implies that, for every $n \geq 0$,

$M, w \sim_{n,\overline{q}} M', w'$. The converse does not hold in general, but it does hold for finite models.[4]

We also introduce the following convenient notation:

- $M, w \preceq_{0,\overline{q}} M', w'$ means that, for every $q \in \overline{q}$, if $w \in \pi(q)$, then $w' \in \pi'(q)$;
- for any $n \geq 1$, $M, w \preceq_{n,\overline{q}} M', w'$ means that, for every v' in M' such that $v' \succeq' w'$, there exists v in M with $v \succeq w$ and $M, v \sim_{n-1,\overline{q}} M', v'$.

With these notations, observe that, for any pointed $\overline{q}$-models $(M, w), (M', w')$ and any $n \geq 0$, we have $M, w \sim_{n,\overline{q}} M', w'$ if, and only if, both $(M, w) \preceq_{n,\overline{q}} (M', w')$ and $M', w' \preceq_{n,\overline{q}} M, w$ hold.

For readers familiar with the bisimulation games of [24, 27], we note that pointed models M, w and M', w' are n-bisimilar precisely when Player II has a winning strategy for the first n moves in the bisimulation game played on these models.

The interest of bounded bisimulation is its close relationship to definability of classes of pointed models. We formalize this in Propositions 7 and 8 below. Define the *depth* of a propositional formula ϕ to be the maximum nesting depth of the implication connective $\rightarrow$. For any finite set of variables $\overline{p}$ and $n \geq 0$, write $F_{n,\overline{p}}$ for the set of intuitionistic equivalence classes of formulas in the variables $\overline{p}$ of depth $\leq n$. Note that, for any n and $\overline{p}$, the set $F_{n,\overline{p}}$ is finite, as may be shown by a simple induction on n using disjunctive normal forms. Fix, for each class $C \in F_{n,\overline{p}}$, a representing formula $\phi_C(\overline{p})$ of depth $\leq n$. We will often confuse a class with its representing formula. In particular, when τ is a subset of $F_{n,\overline{p}}$, we will write $\bigwedge \tau$ to denote the formula $\bigwedge_{C \in \tau} \phi_C$.

For any pointed $\overline{q}$-model (M, w) and $\overline{p} \subseteq \overline{q}$, define

$$\mathrm{Th}_{n,\overline{p}}(M, w) := \{C \in F_{n,\overline{p}} \mid M, w \Vdash \phi_C\},$$

where we note that whether or not a class C belongs to $\mathrm{Th}_{n,\overline{p}}(M, w)$ does not depend on the particular choice of representing formula ϕ_C.

▶ **Proposition 7.** *Let (M, w) and (M', w') be finite pointed $\overline{q}$-models. For any $n \geq 0$, we have $M, w \preceq_{n,\overline{q}} M', w'$ if, and only if, $\mathrm{Th}_{n,\overline{q}}(M, w) \subseteq \mathrm{Th}_{n,\overline{q}}(M', w')$.*

Proof. See, e.g., [65, Sec. 4], compare also [24, Sec. 2]. ◀

Note that Proposition 7 and the preceding remarks in particular entail that for any $n, \overline{q}$, the relation $\sim_{n,\overline{q}}$ is an equivalence relation on the class of finite pointed $\overline{q}$-models. Moreover, this equivalence relation has only finitely many equivalence classes, because each $\sim_{n,\overline{q}}$-class uniquely determines a subset of $F_{n,\overline{q}}$, of which there are finitely many.

[4] Indeed, we will see below that 'for all $n \geq 0$, $M, w \sim_{n,\overline{q}} M', w'$' means precisely that (M, w) and (M', w') satisfy the same intuitionistic formulas. [52, Cor. 35] shows that for finite models, this is equivalent to bisimilarity, while an example in [52, Prop. 27] shows that it is not equivalent in general. When logical equivalence implies bisimilarity, this is called a *Hennessy-Milner* property; for more about such properties in the context of intuitionistic logic, see [14] and the references therein.

This idea lets us relate $\sim_{n,\overline{q}}$-bisimilarity to definability, as we will show now. In order to conveniently state the following proposition, if $\mathcal{K}$ is a class of finite pointed $\overline{p}$-models, then for $n \geq 0$ and $\overline{q} \subseteq \overline{p}$, we will say that the class $\mathcal{K}$ is $(n, \overline{q})$-*upward closed* if, for any $(M, w) \in \mathcal{K}$, and any finite pointed $\overline{p}$-model (M', w'), if $M, w \preceq_{n,\overline{q}} M', w'$, then $(M', w') \in \mathcal{K}$.

▶ **Proposition 8.** *Let $\overline{p}$ be a finite sequence of variables, and let $\mathcal{K}$ be a class of finite pointed $\overline{p}$-models. For any $n \geq 0$ and $\overline{q} \subseteq \overline{p}$, the class $\mathcal{K}$ is $(n, \overline{q})$-upward closed if, and only if, there exists a formula $\chi(\overline{q})$ of depth $\leq n$ such that $[\![\chi]\!]_{\overline{p}} = \mathcal{K}$.*

Proof. The sufficiency is immediate from Proposition 7. For the necessity, assume $\mathcal{K}$ is $(n, \overline{q})$-upward closed, and consider the set $T_{\mathcal{K}} := \{\mathrm{Th}_{n,\overline{q}}(M, w) \mid (M, w) \in \mathcal{K}\}$, which is a subset of $\mathcal{P}(F_{n,\overline{q}})$, and thus finite. Define

$$\chi := \bigvee_{\tau \in T_{\mathcal{K}}} \bigwedge \tau \, ,$$

which is clearly a formula of depth $\leq n$ that only uses variables in $\overline{q}$. We show that $[\![\chi]\!]_{\overline{p}} = \mathcal{K}$. First, if $(M, w) \in \mathcal{K}$, then clearly $M, w \Vdash \bigwedge \mathrm{Th}_{n,\overline{q}}(M, w)$, so $M, w \Vdash \chi$. Conversely, suppose that $M', w' \Vdash \chi$. Pick $(M, w) \in \mathcal{K}$ such that $M', w' \Vdash \bigwedge \mathrm{Th}_{n,\overline{q}}(M, w)$. This means that $\mathrm{Th}_{n,\overline{q}}(M, w) \subseteq \mathrm{Th}_{n,\overline{q}}(M', w')$. By Proposition 7, it follows that $M, w \preceq_{n,\overline{q}} M', w'$. Since $\mathcal{K}$ is $(n, \overline{q})$-upward closed, we conclude that $(M', w') \in \mathcal{K}$. ◀

Combining Proposition 8 with Proposition 5, we see that, in order to establish the existence of uniform interpolants, one needs to show that, for any $\phi(p, \overline{q})$, the sets $\mathcal{E}_p([\![\phi]\!]_{p,\overline{q}})$ and $\mathcal{A}_p([\![\phi]\!]_{p,\overline{q}})$ are $(n, \overline{q})$-upward closed, for some $n \geq 0$. This is the content of the final lemma, the proof of which is a combinatorial argument on Kripke models.

▶ **Lemma 9** (Expansion Lemma)**.** *For any $k \geq 0$, there exists n such that, whenever $\phi(p, \overline{q})$ is a formula of depth $\leq k$, the sets $\mathcal{E}_p([\![\phi]\!]_{p,\overline{q}})$ and $\mathcal{A}_p([\![\phi]\!]_{p,\overline{q}})$ are $(n, \overline{q})$-upward closed.*

We emphasize that Lemma 9 contains the combinatorial core of the uniform interpolation theorem, when viewed semantically. This is the main part of the argument that is highly specific to the class of models (and thus to the logic) under consideration, and often requires an intricate combinatorial argument; we refer to [24, Lem. 4.2], [65, Lem. 5.1], or [30, Lem. 10] for various renderings of such an argument in the case of intuitionistic logic. Note that, in all of these semantic arguments, the number n in general needs to be taken very large compared to k. The name 'Expansion Lemma' comes from [61, Lem. 2.10], where it is used for the analogous combinatorial result that is required for proving uniform interpolation in Gödel-Löb logic **GL**.

Given the above results, we can now also see that any right uniform interpolant for ϕ with respect to p must in fact have $\mathcal{E}_p([\![\phi]\!]_{p,\overline{q}})$ as its semantics; that is, we deduce the converse of Proposition 5.

▶ **Corollary 10.** *Let $\phi(p, \overline{q})$ be a formula.*
1. *If $\varepsilon(\overline{q})$ is a right uniform interpolant for ϕ with respect to p, then $[\![\varepsilon]\!]_{p,\overline{q}} = \mathcal{E}_p([\![\phi]\!]_{p,\overline{q}})$.*
2. *If $\alpha(\overline{q})$ is a left uniform interpolant for ϕ with respect to p, then $[\![\alpha]\!]_{p,\overline{q}} = \mathcal{A}_p([\![\phi]\!]_{p,\overline{q}})$.*

Proof. (1) Assume that ε is a right uniform interpolant for ϕ with respect to p. By Proposition 8 and Lemma 9, pick a formula $\chi(\overline{q})$ such that $[\![\chi]\!]_{p,\overline{q}} = \mathcal{E}_p([\![\phi]\!]_{p,\overline{q}})$. By Proposition 5, χ is a right uniform interpolant for ϕ. It follows from the second remark following Definition 2 that χ and ε are equivalent. In particular, $[\![\varepsilon]\!]_{p,\overline{q}} = [\![\chi]\!]_{p,\overline{q}} = \mathcal{E}_p([\![\phi]\!]_{p,\overline{q}})$. The proof of (2) is analogous. ◀

Corollary 10 depends on the choice of the notion of 'model'. It can happen that a logic with uniform interpolants is complete with respect to a class of models, but the bisimulation quantifiers fail to be definable when restricted to this class; an example in the case of **GL** is given in [11, Exa. 2, p. 108].

2.4 Extensions of the Semantic Method to Other Logics

The proof of uniform interpolation for intuitionistic logic outlined above suggests a general proof strategy for uniform interpolation via definability of bisimulation quantifiers for logics admitting a Kripke semantics; note in particular that the argument given in Proposition 5 does not use any specific properties of intuitionistic logic beyond completeness and usual Craig interpolation (and the latter could have been avoided with some more care). The same proof method is for instance used in [65] to obtain uniform interpolants in Grzegorczyk logic **S4Grz**, and it is also shown there that implication-based uniform interpolants for **K** [20, Sec. 6] and uniform interpolants for **GL** [61] can be obtained in this way.

The article [24], which preceded [65], gave a very similar argument for uniform interpolation in intuitionistic logic, phrased in a more category-theoretic language. The basic idea in [24] (later much expanded in [27]) is that a formula $\phi(\overline{q})$ can be represented as a *sheaf* (i.e., a particular kind of functor), $F_{\overline{q}}(\phi)$, from the category of finite pointed posets with bounded morphisms to the category of sets. This sheaf $F_{\overline{q}}(\phi)$ associates to any finite pointed poset $(W, \preceq, w)$ the set of persistent valuations $\pi\colon \overline{q} \to \mathcal{P}(W)$ such that the model $((W, \preceq, \pi), w)$ forces ϕ. In this setting, existential and universal bisimulation quantifiers with respect to a variable p then correspond to certain natural transformations between these sheaves. The existence of uniform interpolants is obtained by showing that these natural transformations *have an index*, a concept that is analogous to what we called '$(n, \overline{q})$-upward closed' above. Inspired by this work, [30] gives a topological proof of Theorem 1 via Esakia duality, see Subsection 3.2 below. We will go into further detail about the work of [24, 27] in Section 4.

The semantic method also allows one to prove that uniform interpolation *fails* to hold for certain logics. The first instance of this is [25], where it is shown that uniform interpolants fail to exist for the modal logic **S4**; [7, Sec. 3] shows that the same counterexample can be used to show that uniform interpolants do not exist for **K4**. Uniform interpolation also fails for intuitionistic versions of these logics, **iK4**, **iS4** [28, Sec. 3.4].

▶ Remark 11. The difference between implication-based and deductive uniform interpolants can be seen clearly in the case of the modal logic **K**. An algorithm for computing implication-based uniform interpolants for the logic **K** is given in [7], and is also outlined in Subsection 3.3 of Chapter 5. The fact that *deductive* uniform interpolants do not always exist in **K** is essentially due to [23, Thm. 37], where a counterexample is given in the syntax of description logic. The failure is closely related to the failure of the deduction theorem for the global consequence relation, for which see, e.g., [5, Sec. 1.5]. We here give a slightly simpler example, due to G. Metcalfe (personal communication); essentially the same example can be used to show that any strongly complete modal logic which has right uniform interpolants must be weakly transitive. Here, one may even weaken the assumption of having right uniform interpolants to being coherent in the sense of Theorem 13 below [39, Thm. 4.1].

Consider the modal formula $\phi := (q \to p) \wedge (p \to \Box p) \wedge (p \to r)$. We claim that ϕ does not have a right uniform interpolant with respect to p for the global consequence relation $\vdash_{\mathbf{K}}$. The crucial observation towards this claim is that, for any modal formula $\theta(q, r)$,

$$\phi \vdash_{\mathbf{K}} \theta \text{ if, and only if, there exists } N \geq 0 \text{ such that } \bigwedge_{k=0}^{N} (q \to \Box^k r) \vdash_{\mathbf{K}} \theta. \tag{2}$$

For the right-to-left direction of (2), just note that $\phi \vdash_{\mathbf{K}} q \to \Box^k r$ for every $k \geq 0$ and use transitivity of $\vdash_{\mathbf{K}}$. For the left-to-right direction of (2), we reason contrapositively, and assume that there does not exist N such that $\bigwedge_{k=0}^{N}(q \to \Box^k r) \vdash_{\mathbf{K}} \theta$. Using the compactness and completeness theorems for modal logic **K**, there exists a pointed (modal) Kripke (q, r)-model (M, w) such that $M, w \Vdash q \to \Box^k r$ for every $k \geq 0$, but $M, w \nVdash \theta$. Extend the valuation of this model to the variable p by setting $M, w \Vdash p$ if, and only if, $M, w \Vdash \Box^k r$ for every $k \geq 0$. It it then straight forward to check that $M, w \Vdash \phi$, so that $\phi \nvdash_{\mathbf{K}} \theta$ by soundness.

Now suppose that $\varepsilon(q, r)$ would be a deductive right uniform interpolant for ϕ with respect to p. Since $\phi \vdash_{\mathbf{K}} \varepsilon$, by (2) we can pick $N \geq 0$ such that $\bigwedge_{k=0}^{N}(q \to \Box^k r) \vdash_{\mathbf{K}} \varepsilon$. Also, since $\phi \vdash_{\mathbf{K}} q \to \Box^{N+1} r$, we must have $\varepsilon \vdash_{\mathbf{K}} q \to \Box^{N+1} r$. By transitivity of $\vdash_{\mathbf{K}}$, we now obtain that $\bigwedge_{k=0}^{N}(q \to \Box^k r) \vdash_{\mathbf{K}} q \to \Box^{N+1} r$, but this is false, as can be seen from a simple countermodel construction. ⌟

Bisimulation quantifiers were also used to prove that the modal μ-calculus has uniform interpolation [12, Cor. 3.8]. The proof of this result relies on the characterization of the sets of transition systems that are definable in the μ-calculus as being precisely those that are recognizable by so-called μ-automata [34]. The definability of bisimulation quantifiers is then obtained by showing that μ-automata are closed under projections. This method is further used ([12, Thm. 3.9]) to prove that adding bisimulation quantifiers to the syntax of propositional dynamic logic yields a logic that is expressively equivalent to the full modal μ-calculus. An alternative, proof-theoretic approach to uniform interpolation for modal μ-calculus is discussed in depth in Chapter 11 of this volume.

While the semantic method via bisimulation quantifiers is highly versatile, its main drawback is that it does not give an explicit construction of the uniform interpolants of a

formula. In general, the number n in the Expansion Lemma (Lemma 9) is taken in the order of several exponential factors larger than k, so that, while it is theoretically computable, it does not yield an efficient way for calculating uniform interpolants. We come back to this point in Section 5.

3 Uniform Interpolants via Adjunction and Topology

In this section, we will show how the existence of uniform interpolants relates to the existence of adjoints between finitely generated free algebras (Subsection 3.1). We will again focus first on the case of intuitionistic propositional logic, and then show how to generalize this to other logics. Next, in the case of intuitionistic logic, we give a topological argument for why the required adjoints exist (Subsection 3.2). Finally, we relate this to questions in categorical logic that were at the origin of Pitts' investigations (Subsection 3.3).

3.1 Uniform Interpolants as Adjoints

Recall that intuitionistic propositional logic is captured algebraically by the class of *Heyting algebras*, i.e., structures $(A, \wedge, \vee, \Rightarrow, \bot, \top)$ which are bounded distributive lattices having the further property that, for any $a, b \in A$, the element $a \Rightarrow b$ is the maximum of the set $\{c \in A \mid a \wedge c \leq b\}$. If $\overline{p}$ is any finite sequence of propositional variables, then, for any propositional formulas $\phi(\overline{p})$ and $\psi(\overline{p})$, their equivalence classes are elements of the *free* Heyting algebra $\mathbf{H}(\overline{p})$, which we still denote by ϕ and ψ by a slight abuse of notation. The basic connection between Heyting algebras and intuitionistic logic is the fact that an intuitionistic entailment $\phi \vdash \psi$ holds if, and only if, $\phi \leq_{\mathbf{H}(\overline{p})} \psi$.

Let $p, \overline{q}$ be a finite sequence of variables and let $\phi(p, \overline{q})$ be a formula in these variables, which we view as an element of $\mathbf{H}(p, \overline{q})$. Note that the free algebra $\mathbf{H}(\overline{q})$ can be seen as a *subalgebra* of $\mathbf{H}(p, \overline{q})$; formally, we write

$$i \colon \mathbf{H}(\overline{q}) \hookrightarrow \mathbf{H}(p, \overline{q}) \tag{3}$$

for the injective homomorphism that sends any equivalence class of formulas $\psi(\overline{q})$ to itself, now viewed as an element of $\mathbf{H}(p, \overline{q})$. In these terms, the uniform interpolant $\mathsf{E}_p(\phi)$ is an element of $\mathbf{H}(\overline{q})$ with the property that

$$\mathsf{E}_p(\phi) = \min\{\psi \in \mathbf{H}(\overline{q}) \mid \phi \leq_{\mathbf{H}(p,\overline{q})} i(\psi)\}.$$

This equation says precisely that the function $\mathsf{E}_p \colon \mathbf{H}(p, \overline{q}) \to \mathbf{H}(\overline{q})$ is a *lower adjoint* to the homomorphism i of (3).[5]

[5] Note that lower adjoints are sometimes referred to as *left* adjoints, but we prefer 'lower' in this context, in order to avoid confusion with the left-right terminology for uniform interpolants, which uses the *opposite*

The previous paragraph shows that the existence of right uniform interpolants implies the existence of a lower adjoint to the homomorphism i of (3). Similarly, the existence of left uniform interpolants implies the existence of an *upper adjoint* to the homomorphism i. A crucial idea underlying all semantic proof methods for implication-based uniform interpolation is that, conversely, if one wants to construct uniform interpolants, one may first try to construct such lower and upper adjoints, which will yield candidate right and left uniform interpolants, respectively. As already mentioned after the proof of Proposition 5 above, bisimulation quantifiers may be seen as semantic realizations of these adjoints. The following argument is thus, in a sense, an 'algebraic' version of the proof of Proposition 5.

▶ **Proposition 12.** *Let $p, \overline{q}$ be a finite set of variables. Consider the homomorphism i of (3).*

1. *The function i has a lower adjoint if, and only if, every formula $\phi(p, \overline{q})$ has a right uniform interpolant with respect to p.*
2. *The function i has an upper adjoint if, and only if, every formula $\phi(p, \overline{q})$ has a left uniform interpolant with respect to p.*

Proof. (1) The sufficiency of the condition was explained above. For the necessity, let $i^\flat$ be a lower adjoint for i and let $\phi(p, \overline{q})$ be arbitrary. Pick a formula $\psi(\overline{q})$ in the equivalence class $i^\flat(\phi)$. By adjunction, $\phi \leq_{\mathbf{H}(p,\overline{q})} i(\psi)$, so that $\phi \vdash \psi$. Let θ be any p-free formula such that $\phi \vdash \theta$. First, using the (non-uniform) Craig interpolation theorem of intuitionistic propositional logic, pick an interpolating formula θ' using only variables in $\overline{q}$, such that *(a)* $\phi \vdash \theta'$ and *(b)* $\theta' \vdash \theta$. Property *(a)* implies that $\phi \leq_{\mathbf{H}(p,\overline{q})} i(\theta')$, so, by definition of lower adjoints, $i^\flat(\phi) \leq_{\mathbf{H}(\overline{q})} \theta'$. The choice of ψ then gives that $\psi \vdash \theta'$, and now property *(b)*, together with the transitivity of the relation $\vdash$, yield $\psi \vdash \theta$. (2) Similar. ◀

For other logics and their corresponding varieties of algebras, analogous arguments still hold, although one needs to perform them at the level of the *lattices of compact congruences* of free finitely generated algebras. In category-theoretic terms, compact congruences on finitely presented algebras may be seen as *regular epimorphisms* in the category of finitely presented algebras, and this was the original approach taken in [27]. An algebraic rendering of this idea, using the universal-algebraic notion of *coherence*, was developed in [29, 40]. We now briefly recall how this works, referring to Chapter 8 of this volume for more details.

Let $\mathcal{V}$ be a variety of algebras, in the sense of universal algebra, see, e.g., Section 2 of Chapter 8. We write $\mathbf{F}_\mathcal{V}(\overline{x})$ for the free $\mathcal{V}$-algebra over a set of variables $\overline{x}$. Recall that, in Section 3 of that chapter, a consequence relation $\vdash_\mathcal{V}$ was associated to $\mathcal{V}$, and it was shown in Lemma 9 of the same chapter that, for any set of equations $\Sigma \cup \{E\}$ with variables in a set $\overline{x}$, we have $\Sigma \vdash_\mathcal{V} E$ if, and only if, the equation E belongs to the congruence on $\mathbf{F}_\mathcal{V}(\overline{x})$

convention. Below, we will move to homomorphisms between *congruence lattices* rather than between the algebras themselves, and another order flip will take place, which makes the left-right terminology for uniform interpolants coherent with the one for adjoints.

generated by Σ. For any homomorphism $h\colon A \to B$ between algebras in $\mathcal{V}$, we have an adjoint pair of maps

$$h^*\colon \mathsf{Con}\, A \leftrightarrows \mathsf{Con}\, B\colon h^{-1},$$

where h^{-1} sends a congruence β on B to its inverse image $h^{-1}(\beta) := \{(a, a') \in A^2 \mid (ha, ha') \in \beta\}$, and h^* sends a congruence α on A to $\mathrm{Cg}^B(h[\alpha])$, i.e., the congruence on B generated by the forward image $\{(ha, ha') \mid (a, a') \in \alpha\}$. Note that h^* *preserves finitely generated*[6] *congruences*, i.e., when α is a finitely generated congruence, then so is $h^*(\alpha)$. However, it is not the case in general that h^{-1} preserves finitely generated congruences; for a counterexample in the variety of groups, see [29, Example 3.6]. It is shown in [29, Prop. 3.5] that right uniform interpolants with respect to $\vdash_{\mathcal{V}}$ exist if, and only if, for any sets of variables $\overline{x}, \overline{y}$, the inverse image i^{-1} of the inclusion homomorphism

$$i\colon \mathbf{F}_{\mathcal{V}}(\overline{y}) \hookrightarrow \mathbf{F}_{\mathcal{V}}(\overline{x}, \overline{y}) \tag{4}$$

preserves finitely generated congruences. Thus, this property in particular implies that the consequence relation $\vdash_{\mathcal{V}}$ has the interpolation property. It is moreover possible to 'separate out' the usual interpolation property: right uniform interpolants exist if, and only if, (i) the usual interpolation property holds, and (ii) for any *finite* sets of variables $\overline{x}, \overline{y}$, the inverse image i^{-1} of the homomorphism i of (4) preserves finitely generated congruences. Writing $\mathsf{KCon}\, A$ for the semilattice of finitely generated congruences on A, (ii) implies that, for any finite sets of variables $\overline{x}, \overline{y}$, we have an adjunction

$$i^*\colon \mathsf{KCon}\, \mathbf{F}_{\mathcal{V}}(\overline{y}) \leftrightarrows \mathsf{KCon}\, \mathbf{F}_{\mathcal{V}}(\overline{x}, \overline{y})\colon i^{-1}\ .$$

Note that, here, i^{-1} is *upper* adjoint to i^*, whereas in the case of Heyting algebras (Proposition 12) we saw that right uniform interpolants are related to the existence of a *lower* adjoint $i^\flat$ for the homomorphism i. The reason for this order switch is that any Heyting algebra A is isomorphic to the *opposite* of the poset $\mathsf{KCon}\, A$ of compact congruences on A.

It was shown in [29, Prop. 3.8] that (ii) implies that h^{-1} preserves finitely generated congruences for *any* homomorphism between finitely presented algebras. Moreover, [40, Thm. 2.3] establishes that (ii) is also equivalent to the variety $\mathcal{V}$ being *coherent*, meaning that any finitely generated subalgebra of a finitely presented algebra is itself finitely presented. To summarize this discussion, we have the following:

▶ **Theorem 13** ([29, 40])**.** *For any variety $\mathcal{V}$, the following are equivalent:*

1. *for any finite sets of variables $\overline{x}$, $\overline{y}$, the inverse image i^{-1} of the homomorphism i of (4) preserves finitely generated congruences;*

[6] In some literature, in particular [29], 'finitely generated congruences' are also called 'compact congruences', referring to the fact that they are the compact elements of the algebraic lattice $\mathsf{Con}\, A$.

2. *for any homomorphism $h\colon A \to B$ between finitely presented $\mathcal{V}$-algebras, the inverse image h^{-1} preserves finitely generated congruences;*
3. *the variety $\mathcal{V}$ is coherent.*

If, moreover, the consequence relation $\vdash_{\mathcal{V}}$ has the interpolation property, then the above properties are also equivalent to:

4. *for any sets of variables $\overline{x}$, $\overline{y}$, the inverse image i^{-1} of the homomorphism i of (4) preserves finitely generated congruences;*
5. *the consequence relation $\vdash_{\mathcal{V}}$ has right uniform interpolants.*

Similarly, the existence of *left* uniform interpolants is equivalent to the algebraic property that, for any sets $\overline{x}$ and $\overline{y}$ of variables, the direct image map i^* has a *lower adjoint*, also see [29, Sec. 4].

▶ **Example 14.** The variety of MV-algebras, associated to Łukaciewicz multi-valued logic, is coherent, even though implication-style uniform interpolation fails. On the other hand, the variety of modal algebras is not coherent (Remark 11), even though it has implication-style uniform interpolation [20, Sec. 6]. The variety of lattices, which algebraizes a minimal, non-distributive logic of 'and' and 'or', also fails to be coherent [60], see also [40, Sec. 5].

3.2 Esakia Duality and Open Mappings

The point of view that uniform interpolants for intuitionistic logic are given by adjoints to inclusions of free Heyting algebras enables a *topological* analysis of uniform interpolants. The general idea of using topology for studying intuitionistic logic goes back to at least M. H. Stone [63]. To motivate the idea, observe first that the collection $\mathcal{O}(X)$ of open sets of any topological space X is a Heyting algebra, which is even *complete*. The Heyting implication in $\mathcal{O}(X)$ is given, for U and V open sets, by letting $U \Rightarrow V$ be the *topological interior* of the set of points of X which are either in V or not in U. In general, however, *free* Heyting algebras on two or more generators are not complete [4, Thm. 4.2], and can therefore not be of the form $\mathcal{O}(X)$ for any space X.

To obtain *all* Heyting algebras from a topological construction, one may use a theorem due to Stone [63], which implies that any bounded distributive lattice, and thus in particular any Heyting algebra, can be represented as the collection of open *and compact* subsets of a certain topological space. Homomorphisms of Heyting algebras then correspond to certain strongly continuous functions between the spaces in the other direction. We will here use a point of view on this duality due to Priestley [57] and Esakia [15, 16], who consider subsets that are clopen (i.e., closed and open) up-sets (i.e., upward closed) with respect to a partial order that is added as additional structure to the topological space. One way to view the results of Stone, Priestley, and Esakia is that they provide a *canonical* choice for an embedding of a Heyting algebra into a complete Heyting algebra of the form $\mathcal{O}(X)$. Indeed, the space X representing a Heyting algebra H is such that $\mathcal{O}(X)$ is the *ideal completion* of H.

Esakia [15] proved that the category of Heyting algebras is dually equivalent to a subcategory of the category of compact ordered topological spaces, in which the morphisms are the so-called *continuous bounded maps*. We write $\mathbf{E}(\overline{p})$ for the Esakia space dual to the free Heyting algebra $\mathbf{H}(\overline{p})$ over set of generators $\overline{p}$, which is also known as the *canonical model* for intuitionistic propositional logic. Concretely, the points of $\mathbf{E}(\overline{p})$ are the prime theories of intuitionistic propositional logic in variables $\overline{p}$, where a *theory* is a set of formulas closed under entailment and finite conjunction, and a theory T is *prime* if $\bot \notin T$ and $\phi \vee \psi \in T$ implies $\phi \in T$ or $\psi \in T$. The partial order is given by inclusion of prime theories, and the topology on $\mathbf{E}(\overline{p})$ is generated by declaring that, for any $\phi \in \mathbf{H}(\overline{p})$, the set $\widehat{\phi}$ of prime theories containing ϕ is both closed and open (clopen). An important consequence of Esakia duality is that the assignment $\phi \mapsto \widehat{\phi}$ is a Heyting algebra isomorphism between $\mathbf{H}(\overline{p})$ and the Heyting algebra of *clopen up-sets* of the Esakia space X.

We call an Esakia space X *finitely copresented* if X is order-homeomorphic to a clopen up-set of $\mathbf{E}(\overline{p})$, for some finite $\overline{p}$; this name comes from the fact that it is equivalent to saying that the Heyting algebra dual to X is finitely presented. The following theorem is proved in [30, Thm. 2], using methods similar to the Expansion Lemma (Lemma 9) stated above.

▶ **Theorem 15.** *Every continuous bounded map between finitely copresented Esakia spaces is open.*

From Theorem 15, one can deduce in particular that the inclusion homomorphism $i\colon \mathbf{H}(\overline{q}) \hookrightarrow \mathbf{H}(p, \overline{q})$ of (3) has both a lower and an upper adjoint, and thus that uniform interpolants exist in intuitionistic logic, in light of Proposition 12. We explain briefly how this works for the lower adjoint; see [30, Thm. 3] for more details. By Esakia duality, the homomorphism i may equivalently be viewed as p^{-1}, where $p\colon \mathbf{E}(p, \overline{q}) \to \mathbf{E}(\overline{q})$ is the continuous bounded map dual to i. By Theorem 15, this map is open. In particular, for any formula $\phi(p, \overline{q})$, the direct image $p[\widehat{\phi}]$ is also a clopen up-set of $\mathbf{E}(\overline{q})$, so that there must exist some formula ε with $\widehat{\varepsilon} = p[\widehat{\phi}]$. Defining $i^\flat(\phi) := \varepsilon$ now yields a lower adjoint for i.

We remark that Theorem 15 was recently generalized by J. Marquès [49, Thm 1.4.26] to the class of *intuitionistic compact ordered spaces*, also called *generalized Esakia spaces* in [31, Def. 3.1]. In light of a duality of M. Abbadini [1], these spaces, which form a subclass of compact ordered spaces, correspond to a logic where truth values of basic propositions are taken in a *continuous* domain, such as the unit interval, in the vein of Łukaciewicz's many-valued logic, **Ł**. Marquès' result therefore corresponds to a uniform interpolation property for an intuitionistic version of an infinitary many-valued logic [49, Prop. 1.3.23]. A full syntactic analysis of this result has not yet been performed.

3.3 Connections to Categorical Logic

The connection between uniform interpolants and adjoints is analogous to the important idea in categorical logic that quantifiers can be viewed as adjoints [44, 45], and was already pointed out in Pitts' original paper [53, §4]. Indeed, Pitts explains there that the existence

of uniform interpolants for intuitionistic logic implies that *every Heyting algebra can appear as the algebra of truth values of a model of second-order intuitionistic propositional logic*. In more syntactic terms, Pitts' result provides an *interpretation* of intuitionistic second order quantifiers into the propositional fragment. We stress that this interpretation cannot be conservative: the entailment relation in **IPC** is decidable, but the entailment relation in **IPC** enriched with second-order quantifiers is not. A concrete example of a principle that holds for uniform interpolants and not for general second-order quantifiers is given in [53, Exa. 10].

The algebraic consequence of Pitts' theorem points to his original motivation for proving his theorem, which was the following open question:

▶ **Question 16** (Pitts)**.** Is every Heyting algebra the algebra of global truth values of some elementary topos?

An *elementary topos* is a categorical model of higher order intuitionistic logic. The notion may be viewed as a generalization to the categorical level of Heyting algebras, which give an algebraic model of *propositional* intuitionistic logic. A positive answer to Question 16 would mean that any propositional intuitionistic theory can appear as the zeroth level of some higher order intuitionistic theory, and would thus in particular yield a new way of constructing such higher order theories. A classical construction of P. Freyd shows that the answer to the restriction of Question 16 to Boolean algebras is affirmative, see [37, Ex. 9.11], [54, Rem. 4.8 and 4.9] and [42].

In the rest of this subsection, we will show in more detail how Question 16 and Theorem 1 are related; while this has clearly been known since [53], we are not aware of any reference where the relationship is fully explained. In order to do so, let us first quickly recall the categorical notions that play a role here; for a more pedagogical exposition of these notions from a logical perspective, we recommend, e.g., [64].

Let $\mathcal{E}$ be a category. A *subobject* of an object A of $\mathcal{E}$ is an equivalence class of monomorphisms $m\colon S \rightarrowtail A$, where two monomorphisms are equivalent if each can be factored through the other. The collection $\mathsf{Sub}_{\mathcal{E}}(A)$ of subobjects of an object A in $\mathcal{E}$ is then a partially ordered set under the factorization ordering. The association $A \mapsto \mathsf{Sub}_{\mathcal{E}}(A)$ is in fact part of a *contravariant functor*: any morphism $f\colon A \to A'$ in $\mathcal{E}$ naturally gives an order preserving function $f^{-1}\colon \mathsf{Sub}_{\mathcal{E}}(A') \to \mathsf{Sub}_{\mathcal{E}}(A)$, by precomposition with f.

A category $\mathcal{E}$ is called an *elementary topos* if *(a)* $\mathcal{E}$ has finite limits and, *(b)* for every object A in $\mathcal{E}$, there exists a *power object* PA in $\mathcal{E}$ representing the functor sending B to $\mathsf{Sub}_{\mathcal{E}}(A \times B)$; concretely, this means that there is a natural bijection between morphisms $B \to PA$ and subobjects of $A \times B$. Note that, taking A to be the terminal object 1 of $\mathcal{E}$, the object $\Omega := P1$ is then a *subobject classifier* in $\mathcal{E}$, in the sense that, for any object B in $\mathcal{E}$, morphisms from B to Ω are in natural bijection with subobjects of B. For S a subobject of B, we denote the corresponding morphism by $\chi_B\colon B \to \Omega$, and we call it the *characteristic map* of B. The prototypical example of an elementary topos is the category of sets, where finite limits can be computed as subsets of Cartesian products, PA is the power set of A, and Ω is the two-element set.

The two axioms for elementary toposes have many consequences, of which we mention two important ones here. First, for any object A in an elementary topos $\mathcal{E}$, the poset $\mathsf{Sub}_{\mathcal{E}}(A)$ is a Heyting algebra, and for any morphism $f\colon A \to A'$ in $\mathcal{E}$, the morphism f^{-1} is a Heyting algebra homomorphism, see, e.g., [64, Thm. 13.2]. In particular, taking the terminal object 1 of $\mathcal{E}$, we have a Heyting algebra $\mathsf{Sub}_{\mathcal{E}}(1)$, which is called the *algebra of global truth values*. Question 16 now asks if any Heyting algebra may appear as such an algebra of global truth values. One may show that, in order to answer this question in the positive, it would in fact suffice to establish this for free Heyting algebras [55, p. 37]; we omit the proof.

We will now explain, in Proposition 17, why a positive answer to Question 16 would immediately yield Theorem 1, as stated in [53, p. 36]. To this end, recall that, in an elementary topos $\mathcal{E}$, there exist existential and universal quantifiers with respect to any morphism. More precisely, for any morphism $f\colon A \to B$ in $\mathcal{E}$, the morphism $f^{-1}\colon \mathsf{Sub}_{\mathcal{E}}(B) \to \mathsf{Sub}_{\mathcal{E}}(A)$ has an upper adjoint $\forall_f$ and a lower adjoint $\exists_f$, see, e.g., [64, Thm. 13.4 and Thm. 13.5.3]. In the proof of the next proposition, we will apply this fact to the (unique) morphism $!\colon \Omega \to 1$.

▶ **Proposition 17.** *Let $\overline{q}$ be an infinite set of variables and $p \notin \overline{q}$. Suppose that $\mathcal{E}$ is an elementary topos with an isomorphism $\tau\colon \mathsf{Sub}_{\mathcal{E}}(1) \cong \mathbf{H}(\overline{q})\ :\sigma$. There exists a Heyting algebra embedding $j\colon \mathbf{H}(p,\overline{q}) \hookrightarrow \mathsf{Sub}_{\mathcal{E}}(\Omega)$ such that, for any formula $\phi(p,\overline{q})$, the formulas $\tau(\exists_!(j(\phi)))$ and $\tau(\forall_!(j(\phi)))$ are right and left uniform interpolants of ϕ with respect to p.*

Proof. For each $q \in \overline{q}$, let $j(q)$ be the subobject $!^{-1}(\sigma(q))$ of Ω, and moreover let $j(p)$ be the subobject of Ω corresponding to the identity morphism $\Omega \to \Omega$. These assignments extend uniquely to a homomorphism $j\colon \mathbf{H}(p,\overline{q}) \to \mathsf{Sub}_{\mathcal{E}}(\Omega)$. We first show that j is injective. For this, consider the function $\alpha\colon \mathsf{Sub}_{\mathcal{E}}(\Omega) \to \mathbf{H}(\overline{q})^{\mathbf{H}(\overline{q})}$, defined, for $S \in \mathsf{Sub}_{\mathcal{E}}(\Omega)$, by letting $\alpha(S)\colon \mathbf{H}(\overline{q}) \to \mathbf{H}(\overline{q})$ be the function that sends $\phi \in \mathbf{H}(\overline{q})$ to $\tau(\chi^{-1}_{\sigma(\phi)}(S))$. Observe that this function α is a homomorphism of Heyting algebras. One may moreover verify that, for each $q \in \overline{q}$, $\alpha(j(q))$ is the constant function with value q, and that $\alpha(j(p))$ is the identity function. Using that α and j are homomorphisms of Heyting algebras, it follows that, for any $\psi \in \mathbf{H}(p,\overline{q})$, $\alpha(j(\psi))$ is the function which sends $\phi \in \mathbf{H}(\overline{q})$ to the formula $\psi(\phi/p,\overline{q})$, in which ϕ is substituted for all occurrences of p. Now, let $\psi, \psi' \in \mathbf{H}(p,\overline{q})$ be arbitrary, and suppose that $j(\psi) = j(\psi')$. Pick a variable r from $\overline{q}$ which does not appear in ψ or ψ'. Then $\psi(r/p,\overline{q}) = \alpha(j(\psi)) = \alpha(j(\psi')) = \psi'(r/p,\overline{q})$, and thus $\psi = \psi'$.

To finish the proof, consider the commutative diagram

$$\begin{array}{ccc} \mathbf{H}(\overline{q}) & \xrightarrow{\ i\ } & \mathbf{H}(p,\overline{q}) \\ \downarrow{\scriptstyle \sigma} & & \downarrow{\scriptstyle j} \\ \mathsf{Sub}_{\mathcal{E}}(1) & \xrightarrow{\ !^{-1}\ } & \mathsf{Sub}_{\mathcal{E}}(\Omega) \end{array}$$

Note that $\tau \circ \exists_!$ is lower adjoint to $!^{-1} \circ \sigma = j \circ i$. Thus, since j is an order embedding, we get that $\tau \circ \exists_! \circ j$ is lower adjoint to i, and we conclude by Proposition 12. ◀

In an attempt to settle Question 16 in the negative, in light of Proposition 17, Pitts had initially tried to prove the negation of Theorem 1. However, Theorem 1 ended up being true, and Question 16 thus remained open. Theorem 1 has since been used in attempts towards settling Question 16 in the positive, in particular by the late D. Pataraia, although there is no published proof; see, e.g., [36, 35, 42, 55] for more information.

The above category-theoretic viewpoint on uniform interpolation can also be related to the model-theoretic results that can be derived from it, as we will describe in the next section.

4 Model-complete Theories from Uniform Interpolants

In the algebraic viewpoint on propositional logics, we typically study a logical entailment relation as a special case of the equational consequence relation $\vdash_{\mathcal{C}}$ for a class of algebras $\mathcal{C}$ in an algebraic language $\mathcal{L}_{\mathcal{C}}$. For instance, the entailment relation of intuitionistic propositional logic can be viewed in this way as $\vdash_{\mathcal{H}}$, where $\mathcal{H}$ is the class of Heyting algebras, which are structures in the algebraic language $\mathcal{L}_{\mathcal{H}} = \{\wedge, \vee, \Rightarrow, \bot, \top\}$.

In this section, rather than looking at $\vdash_{\mathcal{C}}$ directly, we take a different perspective, and study the *first order theory of* a class of algebras $\mathcal{C}$ itself, which is defined as

$$\mathrm{Th}_{\mathsf{FO}}(\mathcal{C}) := \{\Phi \text{ a first order } \mathcal{L}_{\mathcal{C}}\text{-sentence such that } A \models \Phi \text{ for every } A \in \mathcal{C}\}.$$

For example, $\mathrm{Th}_{\mathsf{FO}}(\mathcal{H})$ contains the sentences $\forall x.\forall y.x \vee y = y \vee x$ and $\exists y.\forall x.x \wedge y = y$, but it does not contain the sentence $\forall x.\exists y.((x \vee y = \top)\ \mathsf{and}\ (x \wedge y = \bot))$, as witnessed, for instance, by the three-element chain.

▶ **Notation.** Note that we write and for the conjunction of first order logic, in order to avoid overloading the symbol $\wedge$, which is already used as an operation symbol of the language $\mathcal{L}_{\mathcal{H}}$ of Heyting algebras; we adopt similar conventions for the other connectives of first order logic. Moreover, in this section we will use the word *term* to refer to a propositional formula, since it is exactly the same thing as a term in the language $\mathcal{L}_{\mathcal{H}}$, while we reserve the word *formula* for formulas of first order logic; as usual, a *sentence* is a first order formula in which all the variables are bound. In order to clearly separate metavariables for propositional formulas from those for first order formulas, as a convention, we continue to use lower case Greek letters to refer to propositional formulas, while we reserve upper case Greek letters for formulas of first order logic.

The starting point of the work of S. Ghilardi and M. Zawadowski in [26, 27] is that Pitts' Theorem 1 can be used to establish that *the first order theory of Heyting algebras has a model completion* [26, p. 29]. The concept of model completion that is used here originates in A. Robinson's model-theoretic algebra [58, 59], and closely relates to the idea of *quantifier elimination*. The prime example is the theory of algebraically closed fields, which is the model completion of the first order theory of integral domains. An example closer to the kind we want to consider here is the first order theory of Boolean algebras, which has a model

completion, namely, the theory of atomless Boolean algebras, see, e.g., [56, Thm. 6.21]. We now recall the definitions and some basic facts about these concepts, referring to, e.g. [10, Sec. 3.5] for more details.

For the remainder of this section, we fix an algebraic language $\mathcal{L}$. A (first order) *theory* is a set T of first order sentences in the language $\mathcal{L}$, and a *model* of T is an $\mathcal{L}$-structure A such that $A \models \Phi$ for every $\Phi \in T$. When T is a theory and $\Phi(\overline{x})$ is a formula, we also write $T \vdash \Phi$ if, for any structure A and any $\overline{a} \in A^{\overline{x}}$, if $A \models \Psi(\overline{x}/\overline{a})$ for every $\Psi \in T$, then $A \models \Phi(\overline{x}/\overline{a})$.

We call a formula Φ *universal* if its prenex form only contains universal quantifiers and we similarly define the notion of *existential formula*. We say that two theories T and T' in the same language are *co-theories* if, for any universal sentence Φ, we have $T \vdash \Phi$ if, and only if, $T' \vdash \Phi$. In other words, writing $T_\forall$ for the set of universal sentences Φ such that $T \vdash \Phi$, T and T' are co-theories means that $T_\forall = T'_\forall$. A theory T is *universally axiomatizable*, or simply *universal*, if, for any $\Phi \in T$, we have $T_\forall \vdash \Phi$. We say that a theory T is *model complete* if, for any formula Φ, there is a universal formula Φ' such that $T \vdash \Phi \leftrightarrow \Phi'$. Notice that it is equivalent to say that, for any formula Ψ, there is an existential formula Ψ' such that $T \vdash \Psi \leftrightarrow \Psi'$, as can be seen by applying the definition of model-completeness to $\neg\Psi$. A stronger property than model-completeness is that the theory T *eliminates quantifiers*, which means that, for any formula Φ, there is a quantifier-free formula Φ' such that $T \vdash \Phi \leftrightarrow \Phi'$.

▶ **Definition 18.** *Let T, T_0 be theories in the language $\mathcal{L}$, and suppose that T_0 is universal.*[7]

- *The theory T is a* model companion *of T_0 if T is a model complete co-theory of T_0.*
- *The theory T is a* model completion *of T_0 if T is a co-theory of T_0 and eliminates quantifiers.*

There is a complementary semantic perspective on model completions, which we also briefly recall now. Let T_0 be a universal theory and let A be a model of T_0. The model A is called *existentially closed* for T_0 provided that, for any quantifier free formula $\Phi(\overline{x}, y)$ and $\overline{a} \in A^{\overline{x}}$, if there exists a model B of T_0 such that B contains A as a submodel and $B \models \exists y.\Phi(\overline{x}/\overline{a}, y)$, then $A \models \exists y.\Phi(\overline{x}/\overline{a}, y)$. Note, for instance, that the existentially closed integral domains are precisely the algebraically closed fields in the usual sense. In general, if T is a model companion of a universal theory T_0, then the models of T are precisely the existentially closed models of T_0. From this, one can deduce that the universal theory T_0 has at most one model companion, up to equivalence, and that this model companion exists if, and only if, there is a first order theory whose models are exactly the existentially closed models of T_0. Moreover, one may show that this model companion is a model completion if, and only if, the class of models of T_0 has the amalgamation property.

We now sketch how a model completion for the theory of Heyting algebras may be obtained from Pitts' quantifiers [26].

[7] There is a more general definition where T_0 is not required to be universal, but it is slightly more complicated to state, and not relevant for the examples of interest to us here.

▶ **Theorem 19.** *The first order theory of Heyting algebras,* $\mathrm{Th}_{\mathsf{FO}}(\mathcal{H})$*, has a model completion.*

Proof (Sketch). It is well-known that the class of Heyting algebras has the amalgamation property, see for instance [47, Prop. 1] or [17] for early algebraic proofs of this fact. In light of the above remarks, it therefore suffices to show that there is a first order theory whose models are exactly the existentially closed Heyting algebras.

Let $\overline{x}, y$ be a finite set of variables and let $\phi_1(\overline{x}, y), \ldots, \phi_n(\overline{x}, y)$ and $\psi_1(\overline{x}, y), \ldots, \psi_m(\overline{x}, y)$ be any terms in these variables, and consider the quantifier-free formula $\Psi(\overline{x})$ given by

$$\exists y.(\phi_1 = \top \text{ and } \cdots \text{ and } \phi_n = \top \text{ and } \psi_1 \neq \top \text{ and } \cdots \text{ and } \psi_m \neq \top) \ .$$

We will show how to express in a first order way that a Heyting algebra satisfies the existential closedness property with respect to this formula Ψ; the case of a general existential formula is then essentially an iteration of this argument. Consider the formula $\Psi'(\overline{x})$ given by

$$\mathsf{E}_y\left(\bigwedge_{i=1}^{n}\phi_i\right) = \top \text{ and } \mathsf{A}_y\left(\left[\bigwedge_{i=1}^{n}\phi_i\right] \Rightarrow \psi_1\right) \neq \top \text{ and } \cdots \text{ and } \mathsf{A}_y\left(\left[\bigwedge_{i=1}^{n}\phi_i\right] \Rightarrow \psi_m\right) \neq \top \ ,$$

where we use the uniform interpolants for propositional formulas, which exist according to Theorem 1. Using the definition of uniform interpolants, one may then prove that, for any Heyting algebra A and $\overline{a} \in A^{\overline{x}}$, we have $A \models \Psi'(\overline{x}/\overline{a})$ if, and only if, $A[y]/\alpha \models \Psi(\overline{x}/\overline{a})$, where $A[y]$ denotes the coproduct of A with the free Heyting algebra over y, and α is the congruence generated by the pairs $(\phi_i(\overline{x}/\overline{a}, y), \top)$, for $i = 1, \ldots, n$. Therefore, a Heyting algebra A satisfies the existential closedness property for the formula $\Psi(\overline{x}, y)$ and a tuple $\overline{a} \in A^{\overline{x}}$ if, and only if, $A \models (\Psi' \to \Psi)(\overline{x}/\overline{a})$. Thus, an existentially closed Heyting algebra must satisfy the first order sentence $\forall\overline{x}.(\Psi' \to \Psi)$. This is the essential property required for axiomatizing existentially closed Heyting algebras. Using this property, a general argument (see, e.g., [10, Thm. 3.5.20]) shows that the model completion of $\mathrm{Th}_{\mathsf{FO}}(\mathcal{H})$ is obtained by adding to the theory $\mathrm{Th}_{\mathsf{FO}}(\mathcal{H})$, for any choice of $\overline{x}, y, \phi_1, \ldots, \phi_n, \psi_1, \ldots, \psi_m$ as above, the formula $\forall\overline{x}.(\Psi' \to \Psi)$. ◀

The above proof method shows that the class of existentially closed Heyting algebras admits a first order axiomatization T, and also gives an effective quantifier elimination procedure for T. [26, Prop. A.2] establishes some concrete properties of existentially closed Heyting algebras, such as the fact that the order must be dense, and that there are no meet-irreducible elements. It is an open problem to obtain a 'manageable' complete axiomatization of the class of existentially closed Heyting algebras. [26, Thm. A.3] further shows that all of the eight varieties of Heyting algebras with the amalgamation property also have model completions. L. Darnière and M. Junker [13] obtain concrete axiomatizations for the model completions of all of these varieties, *except* for the variety of all Heyting algebras. Several more results on the (non-)existence of model completions for classes of modal algebras are obtained in [27]. More generally, an algebraic method for establishing (non-)existence of model companions for many modal logics, substructural logics, and varieties of lattices, is developed in [39, 40].

We remark that Theorem 19 can be viewed as a consequence of a more general phenomenon that goes back to work of W. Wheeler [67, 68], as explained in [26, 27], and also further investigated in [29, 50]. The general idea is that, for T a universal theory in an algebraic language $\mathcal{L}$, the existence of a model companion for T can be characterized via properties of the category $\mathcal{F}_T$ of *finitely presentable T-structures*, with homomorphisms between them. For instance, it is proved in [26, Thm. 3.4] that, for an equational theory T, if the opposite $(\mathcal{F}_T)^{\mathrm{op}}$ is an *r-Heyting category*, then T has a model completion. Moreover, in the presence of a few additional assumptions on T, the converse also holds. From this perspective, a generalized form of Theorem 1 can first be used to establish that the opposite of the category of finitely presented Heyting algebras is an r-Heyting category, from which Theorem 19 then follows by the general theory; this is the approach taken in [27]. Using the converse direction, one may also use this general approach to deduce that certain theories *do not* have a model completion, as is for instance done for many extensions of modal logic **K4** in [27, Ch. 6].

5 Syntactic Methods, Further Results and Connections

Our focus in this chapter so far has been on connections between uniform interpolation and *semantics*. The *syntactic* method of establishing uniform interpolation is discussed in more detail in Section 3 of Chapter 5 in this volume; we will here just make some brief remarks about the method and its relation to the semantic method.

The basic idea of the syntactic proof style is to recursively construct candidate formulas $\mathsf{E}_p(\phi)$ and $\mathsf{A}_p(\phi)$ for a formula ϕ, usually based on a large case distinction on the shape of ϕ, and the rules that may be applied to it in a particular sequent calculus for the logic. The most intricate part of such proofs of uniform interpolation is the proof of correctness of these candidate formulas $\mathsf{E}_p(\phi)$ and $\mathsf{A}_p(\phi)$, which is typically done by induction on the structure of a 'potential derivation' using ϕ in the sequent calculus. In order to make such an induction possible, one needs a well-founded order on the set of potential derivations that use ϕ. For many logics, including **IPC**, it is not a priori obvious how to obtain a sequent calculus that terminates in the strong sense required for constructing uniform interpolants.

The syntactic method was in particular the one that was originally used by Pitts to establish Theorem 1 in [53], and was subsequently used for proving the implication-based uniform interpolation property for a great number of other logics; for instance, this was done by Bílková [7] for the basic modal logic **K** and for the modal logic of reflexive frames, **T**, and for provability logics **GL** and **Grz** in [6], also see [8]. The method was further extended by Iemhoff [32, 33], who identifies precise general criteria for sequent calculi that allow one to establish uniform interpolation; these ideas were subsequently applied to obtain syntactic proofs of uniform interpolation for a number of intuitionistic modal logics, see, e.g., [28, 2, 62], and the further references given in Chapter 5 of this volume. In the context of substructural logic, Alizadeh, Derakhshan, and Ono [3] establish uniform interpolation for full Lambek calculus and for linear and affine logics without exponentials. Interestingly, in that context,

even the *predicate* versions of the logics have uniform interpolation, in contrast with the case of classical logic, see e.g. Theorem 2.17 in Chapter 2.

As mentioned at the end of Section 2, an advantage of this syntactic method over the semantic one is that, at least in theory, it provides better bounds on the complexity of the uniform interpolants. The intuitive reason for this is that, to find a uniform interpolant for a formula ϕ, the semantic method performs a search among 'all possible bisimilarity types of Kripke models, considered up to the depth of ϕ', while the syntactic method searches 'all possible proofs of ϕ'. The latter is in general a (much) smaller number. In practice, however, it is usually not feasible to compute uniform interpolants by hand in a logic such as **IPC**, even with the syntactic method, as the calculations quickly become complex even on small examples.

Uniform interpolation is also studied in the context of knowledge representation via description logics, where it is in particular relevant for ontology extensions [22], and where the complexity and size of uniform interpolants have been studied in detail; see Chapter 15 of this volume.

The algorithms for computing uniform interpolants are often intricate, and it is a non-trivial task to implement them correctly. To mitigate this, in [19], a version of Pitts' algorithm for computing uniform interpolants for **IPC** was implemented and moreover verified to be correct in the Rocq proof assistant. These methods were subsequently extended to provide a verified computation of uniform interpolants for the modal logics **K** (where the uniform interpolants are not deductive but implication-based), for **GL**, and for the intuitionistic modal logic **iSL** [18]. This work in particular provides a uniform interpolant calculator which is available as an online application at `https://hferee.github.io/UIML/`, and has recently been used in theoretical research for computing specific uniform interpolants [38].

In Table 1, we summarize the current state of the art of the literature on uniform interpolation for a number of common logics, and the existence of model companions for their corresponding classes of algebras. Note that this table also implies a number of open questions: for all the empty cells, at the time of writing, the author is unaware of any answers, yet hopeful that readers of this chapter will be inspired to produce some.[8] For more information, we refer to the table maintained by T. Kurahashi on the webpage `https://www2.kobe-u.ac.jp/~tk/jp/notes/ULIP.html`, which surveys Craig interpolation, implication-style uniform interpolation, and their Lyndon variants, for a large number of intermediate, modal, and intuitionistic modal logics.

[8] Just before this chapter was going to press in December 2025, the author was informed that uniform interpolation for iGL has been proved by I. van der Giessen, B. Sierra Miranda, and G. Menéndez Turata.

Table 1 Summary of the literature on uniform interpolation and model companions. The results on Boolean algebras are folklore. A double ✗ means that even the *non-uniform* form of interpolation fails in the logic. An equality sign in the 'deductive UI' column means that a deduction property holds in the logic, so that the result is the same as for implication-based UI; see the third remark following Definition 2. In the last column, the symbol ^{A} indicates that the class of algebras also has amalgamation, so that the model companion is in fact a model completion. Unless mentioned otherwise, the results in the last column are due to [26].

logic	implication UI	deductive UI	FO-theory has model companion
CPC	✓	=	$\checkmark^A$ [folklore]
IPC	✓ [53]	=	$\checkmark^A$
K	✓ [20, Sec. 6]	✗ [23], cf. Remark 11	✗
S4	✗ [25]	=	✗ [46, Thm. 2]
K4	✗ [7]	=	✗
GL	✓ [61, Cor. 2.13]	=	$\checkmark^A$
Grz	✓ [65, Sec. 8]	=	$\checkmark^A$
Ł	✗✗ [41, 51]	✓ [51, Rem. 4]	✗ [43]
LTL	✗✗ [48]	=	✓ (a variant) [21]
$\mathbf{K}_\mu$	=	✓ [12]	
iK, **iKD**	✓ [33]		
iSL	✓ [66, 18]		
iK4, **iS4**	✗ [28]		
iGL, **IK**, **CK**			

Acknowledgments

The author is grateful for several discussions on parts of this chapter with Balder ten Cate, Silvio Ghilardi, Iris van der Giessen, Jérémie Marquès, George Metcalfe, and Frank Wolter. Van der Giessen and Metcalfe also each provided detailed reviews of earlier versions of this chapter, which have been a great help in improving it. This work has received financial support from the Agence Nationale de la Recherche (ANR), project ANR-23-CE48-0012-01.

References

1 Marco Abbadini. The dual of compact ordered spaces is a variety. *Theory and Applications of Categories*, 34:1401–1439, 2019.

2 Amir Akbar Tabatabai, Rosalie Iemhoff, and Rahaleh Jalali. Uniform Lyndon interpolation for intuitionistic monotone modal logic. In *Advances in Modal Logic*, volume 14, pages 77–96, 2022.

3 Majid Alizadeh, Farzaneh Derakhshan, and Hiroakira Ono. Uniform interpolation in substructural logics. *The Review of Symbolic Logic*, 7(3):455–483, 2014. doi:10.1017/S175502031400015X.

4 Fabio Bellissima. Finitely generated free Heyting algebras. *Journal of Symbolic Logic*, 51(1):152–165, 1986. doi:10.2307/2273952.

5 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

6 Marta Bílková. *Interpolation in modal logics*. PhD thesis, Univerzita Karlova, Prague, 2006. URL: http://hdl.handle.net/20.500.11956/15732.

7 Marta Bílková. Uniform Interpolation and Propositional Quantifiers in Modal Logics. *Studia Logica*, 85:1–31, 2007.

8 Marta Bílková. Uniform Interpolation in provability logics, 2022. arXiv:2211.02591.

9 A. Chagrov and M. Zakharyaschev. *Modal Logic*, volume 35 of *Oxford Logic Guides*. Clarendon Press, Oxford, 1997. doi:10.2307/2693587.

10 Chen Chung Chang and H. Jerome Keisler. *Model theory, 3rd edition*. North Holland, 1990.

11 Giovanna D'Agostino. Uniform interpolation, bisimulation quantifiers, and fixed points. In Balder D. ten Cate and Henk W. Zeevat, editors, *Logic, Language, and Computation*, pages 96–116, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

12 Giovanna D'Agostino and Marco Hollenberg. Logical questions concerning the µ-calculus: interpolation, Lyndon and Łoś-Tarski. *J. Symbolic Logic*, 65(1):310–332, 2000. doi:10.2307/2586539.

13 Luck Darnière and Markus Junker. Model completion of varieties of co-Heyting algebras. *Houston J. Math.*, 44(1):49–82, 2018.

14 Jim de Groot and Dirk Pattinson. Hennessy-Milner properties via topological compactness. *Information and Computation*, 287:104757, 2022. Special Issue: Selected Papers from WoLLIC 2019, the 26th Workshop on Logic, Language, Information and Computation. doi:10.1016/j.ic.2021.104757.

15 Leo Esakia. Topological Kripke models. *Soviet Math. Dokl.*, 15:147–151, 1974.

16 Leo Esakia. *Heyting algebras: Duality theory*, volume 50 of *Trends in Logic*. Springer, 1985. 2019 translation of the original. doi:10.1007/978-3-030-12096-2.

17 Isidore Fleischer. Relatively pseudocomplemented semilattices amalgamate strongly. *Algebra Universalis*, 11:130–132, 1980. doi:10.1007/BF02483089.

18 Hugo Férée, Iris van der Giessen, Sam van Gool, and Ian Shillito. Mechanised uniform interpolation for K, GL, and iSL. In Christoph Benzmüller, Marijn J.H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning: 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3–6, 2024, Proceedings, Part I*, volume 14740 of *Lecture Notes in Computer Science*. Springer, 2024. doi:10.1007/978-3-031-63501-4_3.

19 Hugo Férée and Sam van Gool. Formalizing and computing propositional quantifiers. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2023, page 148–158, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3573105.3575668.

20 Silvio Ghilardi. An algebraic theory of normal forms. *Annals of Pure and Applied Logic*, 71:189–245, 1995.

21 Silvio Ghilardi and Sam van Gool. Monadic second order logic as the model companion of temporal logic. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 417–426. ACM, 2016. doi:10.1145/2933575.2933609.

22 Silvio Ghilardi, Carsten Lutz, and Frank Wolter. Did I damage my ontology? A case for conservative extensions in description logics. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 187–197. AAAI Press, 2006. URL: http://www.aaai.org/Library/KR/2006/kr06-021.php.

23 Silvio Ghilardi, Carsten Lutz, and Frank Wolter. Did I damage my ontology? A case for conservative extensions in description logics (long version). Technical report, 2006. URL: https://www.csc.liv.ac.uk/~frank/publ/SCW.pdf.

24 Silvio Ghilardi and Marek Zawadowski. A sheaf representation and duality for finitely presented Heyting algebras. *Journal of Symbolic Logic*, 60(3):911–939, 1995. doi:10.2307/2275765.

25 Silvio Ghilardi and Marek Zawadowski. Undefinability of Propositional Quantifiers in the Modal System S4. *Studia Logica*, 55:259–271, 1995.

26 Silvio Ghilardi and Marek Zawadowski. Model completions and r-Heyting categories. *Annals of Pure and Applied Logic*, 88(1):27–46, 1997.

27 Silvio Ghilardi and Marek Zawadowski. *Sheaves, Games, and Model Completions*. Springer, 2002. doi:10.1007/978-94-015-9936-8.

28 Iris van der Giessen. *Uniform Interpolation and Admissible Rules*. PhD thesis, Utrecht University, 2022. Quaestiones Infinitae 138. doi:10.33540/1486.

29 Sam van Gool, George Metcalfe, and Constantine Tsinakis. Uniform Interpolation and Compact Congruences. *Annals of Pure and Applied Logic*, 168:1927–1948, 2017. doi:10.1016/j.apal.2017.05.001.

30 Sam van Gool and Luca Reggio. An open mapping theorem for finitely copresented Esakia spaces. *Topology and its Applications*, 240:69–77, 2018. doi:10.1016/j.topol.2018.03.006.

31 Dirk Hofmann and Pedro Nora. Esakia spaces via idempotent split completion. *Textos de Matemática (Serie B)*, 46:201–220, 2014.

32 Rosalie Iemhoff. Uniform interpolation and sequent calculi in modal logic. *Arch. Math. Logic*, 58(1-2):155–181, 2019.

33 Rosalie Iemhoff. Uniform interpolation and the existence of sequent calculi. *Ann. Pure Appl. Log.*, 170(11), 2019. doi:10.1016/J.APAL.2019.05.008.

34 David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *CONCUR '96: Concurrency Theory*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1996.

35 Mamuka Jibladze. Comment on: Constructing a topos from a Heyting algebra, 2017. URL: https://mathoverflow.net/questions/258831/constructing-a-topos-from-a-heyting-algebra#comment638566_258831.

36 Peter Johnstone. Hochas and minimal toposes. In *International Workshop on Topological Methods in Logic III*, 2012. URL: https://rmi.tsu.ge/tolo3/Presentations/Johnstone_1.pdf.

37 Peter T. Johnstone. *Topos theory*. Academic Press, 1977.

38 Zoltan A. Kocsis. Proof-theoretic methods in quantifier-free definability. *Annals of Pure and Applied Logic*, 176(4), 2025. 103555.

39 Tomasz Kowalski and George Metcalfe. Coherence in modal logic. In *Advances in Modal Logic*, volume 12, pages 459–472. College Publications, 2018.

40 Tomasz Kowalski and George Metcalfe. Uniform interpolation and coherence. *Annals of Pure and Applied Logic*, 170(7):825–841, 2019. doi:10.1016/j.apal.2019.02.004.

41 Piotr S. Krzystek and Stanislaw Zachorowski. Lukasiewicz logics have not the interpolation property. *Reports on Mathematical Logic*, 9:39–40, 1977.

42 Evgeny Kuznetsov. On Boolean Topos Constructions by Freyd and Pataraia and their generalizations. In *TACL 2024 Book of Abstracts*, pages 183–184, 2024. URL: https://iiia.csic.es/tacl2024/abstracts/conference/contributed/TACL_2024_paper_136.pdf.

43 Francesco Lacava. Alcune proprietà delle Ł-algebre e delle Ł-algebre esistenzialmente chiuse. *Boll. Un. Mat. Ital. A (5)*, 16(2):360–366, 1979.

44 Joachim Lambek. Deductive systems and categories. I. Syntactic calculus and residuated categories. *Mathematical Systems Theory*, 2:287–318, 1968.

45 F. William Lawvere. Adjointness in Foundations. *Dialectica*, 23(3/4):281–296, 1969. doi:10.1111/j.1746-8361.1969.tb01194.x.

46 Paolo Lipparini. Existentially complete closure algebras. *Boll. Un. Mat. Ital. Algebra e Geometria*, VI(I-D)(1):13–19, 1982.

47 Larisa L. Maksimova. Craig's theorem in superintuitionistic logics and amalgamable varieties of pseudo-boolean algebras. *Algebra and Logic*, 16(6):427–455, 1977.

48 Larisa L. Maksimova. Temporal logics with "the next" operator do not have interpolation or the Beth property. *Siberian Mathematical Journal*, 32(6):989–993, 1991.

49 Jérémie Marquès. *Categorical logic from the perspective of duality and compact ordered spaces*. PhD thesis, Université Côte d'Azur, Nice, 2023. URL: https://jeremie-marques.name/thesis.pdf.

50 George Metcalfe and Luca Reggio. Model Completions for Universal Classes of Algebras: Necessary and Sufficient Conditions. *The Journal of Symbolic Logic*, 88(1):381–417, 2023. doi:10.1017/jsl.2022.1.

51 Daniele Mundici. Consequence and Interpolation in Łukasiewicz logic. *Studia Logica*, 99(1):269–278, 2011.

52 Anna Patterson. Bisimulation and propositional intuitionistic logic. In Antoni W. Mazurkiewicz and Józef Winkowski, editors, *CONCUR '97: Proceedings of the 8th International Conference on Concurrency Theory*, 1997.

53 Andrew M. Pitts. On an interpretation of second-order quantification in first-order intuitionistic propositional logic. *J. Symbolic Logic*, 57(1):33–52, 1992.

54 Andrew M. Pitts. Tripos theory in retrospect. *Mathematical Structures in Computer Science*, 12:265–279, 2002.

55 Andrew M. Pitts. Heyting Algebras and Higher-Order Logic. Slides of a lecture at "Heyting Day 2025", 2025. URL: https://www.knaw.nl/en/events/heyting-day-2025-models-intuitionism-and-computability.

56 Bruno Poizat. *A Course in Model Theory*. Springer-Verlag New York, 2000.

57 Hilary A. Priestley. Representation of distributive lattices by means of ordered Stone spaces. *Bull. London Math. Soc.*, 2:186–190, 1970. doi:10.1112/blms/2.2.186.

58 Abraham Robinson. *On the Metamathematics of Algebra*. North-Holland, 1951.

59 Abraham Robinson. *Introduction to model theory and to the metamathematics of algebra*. Studies in logic and the foundations of mathematics. North-Holland, 1963.

60 Peter H. Schmitt. Algebraically complete lattices. *Algebra Universalis*, 17:135–142, 1983.

61 V. Yu. Shavrukov. Subalgebras of Diagonalizable Algebras of Theories Containing Arithmetic. *Dissertationes Mathematicae*, 323, 1993. URL: `http://matwbn.icm.edu.pl/ksiazki/rm/rm323/rm32301.pdf`.

62 Ian Shillito, Iris van der Giessen, Rajeev Goré, and Rosalie Iemhoff. A new calculus for intuitionistic strong löb logic: Strong termination and cut-elimination, formalised. In Revantha Ramanayake and Josef Urban, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 73–93, Cham, 2023. Springer Nature Switzerland.

63 Marshall H. Stone. Topological representations of distributive lattices and Brouwerian logics. *Čas. Mat. Fys.*, 67:1–25, 1938. `doi:10.21136/cpmf.1938.124080`.

64 Thomas Streicher. Introduction to category theory and categorical logic. Lecture Notes, 2004. URL: `https://www2.mathematik.tu-darmstadt.de/~streicher/CTCL.pdf`.

65 Albert Visser. Uniform interpolation and layered bisimulation. In P. Hájek, editor, *Gödel '96 proceedings*, volume 6 of *Lecture Notes in Logic*, pages 139–164. Springer-Verlag, 1996. URL: `http://projecteuclid.org/download/pdf_1/euclid.lnl/1235417019`.

66 Albert Visser and Tadeusz Litak. Lewis and Brouwer meet Strong Löb, 2024. `arXiv:2404.11969`.

67 William H. Wheeler. Model-companions and definability in existentially complete structures. *Israel Journal of Mathematics*, 25:305–330, 1976.

68 William H. Wheeler. A characterization of companionable, universal theories. *J. Symbolic Logic*, 25(3):402–429, 1978.

Chapter 10

From Interpolating Formulas to Separating Languages and Back Again

Agi Kurucz
King's College London, UK

Frank Wolter
University of Liverpool, UK

Michael Zakharyaschev
Birkbeck, University of London, UK

Abstract

Traditionally, research on Craig interpolation is concerned with (*a*) establishing the Craig interpolation property (CIP) of a logic saying that every valid implication in the logic has a Craig interpolant and (*b*) designing algorithms that extract Craig interpolants from proofs. Logics that lack the CIP are regarded as 'pathological' and excluded from consideration. In this chapter, we survey variations and generalisations of traditional Craig interpolation. First, we consider Craig interpolants for implications in logics without the CIP, focusing on the decidability and complexity of deciding their existence. We then generalise interpolation by looking for Craig interpolants in languages L' that can be weaker than the language L of the given implication. Thus, do not only we restrict the non-logical symbols of Craig interpolants but also the logical ones. The resulting L/L'-interpolation problem generalises L/L'-definability, the question whether an L-formula is equivalent to some L'-formula. After that, we move from logical languages to formal languages where interpolation disguises itself as separation: given two disjoint languages in a class $\mathcal{C}$, does there exist a separating language in a smaller class $\mathcal{C}'$? This question is particularly well-studied in the case when the input languages are regular and the separating language is first-order definable. Finally, we connect the different research strands by showing how the decidability of the separation problem for regular languages can be used to prove the decidability of Craig interpolant existence for linear temporal logic LTL.



© Agi Kurucz, Frank Wolter, and Michael Zakharyaschev;
licensed under Creative Commons License CC BY 4.0

In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 10, pages 329–369. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.j

Contents

1 Introduction 330
1.1 Craig Interpolant Existence Problem 331
1.2 Interpolation and Definability in Weaker Languages 331
1.3 Defining and Separating Formal Languages 332
2 First-Order and Normal Modal Logics: a Quick Reminder 332
3 Interpolant Existence in Logics without Craig Interpolation Property . 335
3.1 Interpolant Existence in Normal Modal Logics 335
3.2 The Finite Bisimilar Model Property of Modal Logic with Nominals 339
3.3 Interpolant Existence in Fragments of First-Order Logic 342
4 Interpolation, Separation and Definability in Weaker Languages 344
4.1 Definability and Separation in Fragments of First-Order Logic 345
4.2 Definability and Separation for Logics with Second-Order Quantifiers and Fixed-Point Operators 348
5 Definability and Separation in Formal Languages 352
5.1 Definability 354
5.2 Separation 355
6 Back Again: Interpolant Existence in Linear Temporal Logic 358
7 Research Directions and Open Problems 362
Acknowledgments 363
References 363

1 Introduction

Suppose L is a logic and φ, ψ are formulas in the language of L. A *Craig interpolant* for the entailment $\varphi \models_L \psi$ in L is a formula χ, all of whose non-logical symbols occur in both φ and ψ, such that the entailments $\varphi \models_L \chi$ and $\chi \models_L \psi$ are valid in L. The logic L is said to have the *Craig interpolation property* (CIP, for short) if *every* valid entailment $\varphi \models_L \psi$ has a Craig interpolant χ in L. Since Craig's [34] discovery that first-order logic, FO, enjoys the CIP, two questions have traditionally been addressed for every notable logic L: Does L have the CIP? And if it does, how to uniformly extract Craig interpolants from formal proofs of entailments $\varphi \models_L \psi$? Logics lacking interpolants for some valid entailments—the ugly ducklings of the family—have only recently drawn attention in the Craig interpolation line of research and applications.

Our aim in this chapter is to present a broader view on the interpolation-related research directions and discuss most important results. We proceed in three generalisation steps. The first of them breaks the traditional uniform 'all or nothing' approach to the CIP and turns it into a *decision problem*.

1.1 Craig Interpolant Existence Problem

We start by looking at a few important logics L that *do not* enjoy the CIP: propositional modal logics such as $\mathsf{S4.3}$, weak $\mathsf{K4}$, and Alt_3, modal logics with nominals, decidable fragments of FO such as the two variable and guarded fragments FO^2 and GF, linear temporal logic LTL, and first-order modal logics under the constant domain semantics. For all of these logics L, the existence of a Craig interpolant for an entailment $\varphi \models_L \psi$ does not follow from the validity of that entailment in L. We regard it as a new type of decision problem for L, the *interpolant existence problem* (IEP): given input formulas φ and ψ, decide whether the entailment $\varphi \models_L \psi$ has a Craig interpolant in L. As the IEP for L typically generalises entailment in L, the former might be computationally harder than the latter; in fact, the IEP can turn out to be undecidable for some decidable L. We discuss recent developments in this area, focusing on the logics listed above.

Craig interpolation is concerned with the shared non-logical symbols of the premise φ and conclusion ψ. One can think of Craig interpolants as eliminating the non-shared symbols in the premise and conclusion, thereby providing an 'explanation' of the entailment in relevant terms. For example, in the extreme case when the signatures of propositional formulas φ and ψ are disjoint, only the logical constants $\bot$ or $\top$ can be a Craig interpolant for $\varphi \models_L \psi$, which means that either $\neg\varphi$ or ψ is valid in L. To find an 'explanation in simpler terms', one could look for interpolants that eliminate certain logical constructs from the language, say recursion or counting. This leads us to the second step of generalisation.

1.2 Interpolation and Definability in Weaker Languages

Let L' be a sublanguage of L. For example, L could be the modal μ-calculus while L' the basic modal logic (without fixed-point operators). The *L/L'-interpolant existence problem* (or L/L'-IEP) is to decide whether given L-formulas φ and ψ have an *L'-interpolant*, that is, $\varphi \models_L \chi$ and $\chi \models_L \psi$, for some L'-formula χ. For this generalisation of Craig interpolation, the existence of an L/L'-interpolant typically does not follow from the validity of $\varphi \models_L \psi$, and we are again facing a decision problem that can be much harder than entailment in L.

The L/L'-IEP also generalises the *L/L'-definability problem*, which asks whether a given formula in an expressive logic L is *logically equivalent* to some formula in a weaker language L'. This definability problem has been considered for numerous pairs L/L' of logics including FO and modal logic (van Benthem's theorem) [16], MSO and modal μ-calculus [55], various guarded logics with and without recursion [48, 12] as well as datalog and unions of conjunctive queries in database theory [33, 80]. We discuss work on generalised interpolation and definability for modal logic, the modal μ-calculus, and fragments of FO with and without counting.

The L/L'-IEP can also be viewed as a separation problem for the structures defined by L-formulas. If $\varphi \models_L \psi$, then φ and $\neg\psi$ define disjoint classes S_φ and $S_{\neg\psi}$ of L-structures (in formal verification, they could describe, respectively, a set of initial states in a transition

system and a set of 'bad' states that should not be reached via any computation). Then an L'-interpolant χ for $\varphi \models_L \psi$ defines a class S_χ of L'-structures separating S_φ and $S_{\neg\psi}$ in the sense that $S_\varphi \subseteq S_\chi$ and $S_{\neg\psi} \cap S_\chi = \emptyset$.

1.3 Defining and Separating Formal Languages

Our third step consists of looking into definability and separability in the context of formal languages, where the study of these problems has existed and flourished independently from Craig interpolation for many years. The *separation* problem asks, given two formal languages $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ from some class $\mathcal{C}$ (say, specified by means of some device such as a finite automaton), whether there exists a language $\boldsymbol{L}$ belonging to some subclass $\mathcal{C}'$ (say, specified using a less powerful device, such as a counter-free automaton) that separates $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ in the sense that $\boldsymbol{L}_1 \subseteq \boldsymbol{L}$ and $\boldsymbol{L}_2 \cap \boldsymbol{L} = \emptyset$. If $\mathcal{C}$ is closed under complementation, then separation generalises *definability* (also known as *membership* or *characterisation*), the problem that asks whether a given language in $\mathcal{C}$ belongs to $\mathcal{C}'$.

A large chunk of the research in this area has been focused on deciding and separating regular languages. In Section 5, we discuss in detail a particular instance of these questions: the decidability of defining/separating regular languages consisting of finite words by star-free (or, equivalently, FO-definable) languages. We also provide further references to work in a wider context including languages with infinite and tree-shaped words, employing FO-fragments as definitions/separators, and defining/separating non-regular languages by means of regular ones.

Rather surprisingly, it turns out that this final third step leads to a beautiful solution to a purely logical problem encountered in the first step: In Section 6, we show how the Craig interpolant existence problem for propositional linear temporal logic LTL over finite timelines (or, equivalently, FO over finite words) can be solved by deciding separability of regular languages by star-free languages.

2 First-Order and Normal Modal Logics: a Quick Reminder

The purpose of this section is to introduce the (standard) notation we use when considering first-order logic and normal modal logics later on in the chapter. For more details on the basics of those logics the reader is referred to [29, 28, 92].

The variant of first-order logic, FO, we consider here is equipped with equality but does not have constant and function symbols. Hence FO-*formulas* are defined by the grammar

$$\varphi \quad := \quad \top \mid R(x_1, \dots, x_n) \mid x_1 = x_2 \mid \varphi \wedge \varphi' \mid \neg\varphi \mid \exists x\, \varphi,$$

where $\top$ is the logical constant 'truth', the x_i and x range over individual variables, and R over predicate symbols of arity n, for any $n \geq 0$. For a tuple $\boldsymbol{x}$ of variables, write $\varphi(\boldsymbol{x})$ to indicate that the free variables of φ occur in $\boldsymbol{x}$. A set σ of predicate symbols is called a

signature (for FO); the signature $sig(\varphi)$ of a formula φ comprises those predicate symbols that occur in φ. A *σ-structure* takes the form

$$\mathfrak{A} = \big(\mathsf{dom}(\mathfrak{A}), (R^{\mathfrak{A}})_{R\in\sigma}\big),$$

where $\mathsf{dom}(\mathfrak{A}) \neq \emptyset$ is the *domain* of $\mathfrak{A}$ and $R^{\mathfrak{A}}$ is a relation over $\mathsf{dom}(\mathfrak{A})$ of the same arity as $R \in \sigma$. For $\varphi(\boldsymbol{x})$ with $\boldsymbol{x} = x_1, \dots, x_n$ and a tuple $\boldsymbol{a} = a_1, \dots, a_n$ with $a_i \in \mathsf{dom}(\mathfrak{A})$, we write $\mathfrak{A} \models \varphi(\boldsymbol{a})$ if φ is true in $\mathfrak{A}$ under the assignment $x_i \mapsto a_i$. We write $\varphi \models \psi$ if $\mathfrak{A} \models \varphi(\boldsymbol{a})$ implies $\mathfrak{A} \models \psi(\boldsymbol{a})$ for all σ-structures $\mathfrak{A}$ and tuples $\boldsymbol{a}$. Note that $\varphi \models \psi$ iff $\models \varphi \to \psi$, that is, $\varphi \to \psi$ is true in all structures.

We are also going to consider a few standard FO*-fragments* (that is, subsets of FO-formulas). For example, by $\mathsf{FO}_{\not=}$ we denote the *equality-free fragment* of FO. A fragment F is said to be *closed under the Booleans* if $\top \in \mathsf{F}$ and whenever $\varphi, \psi \in \mathsf{F}$, then $\varphi \wedge \psi \in \mathsf{F}$ and $\neg\varphi \in \mathsf{F}$.

▶ **Definition 1.** Let F be an FO-fragment and $\varphi(\boldsymbol{x}), \psi(\boldsymbol{x}) \in \mathsf{F}$. A formula $\chi \in \mathsf{F}$ is a *Craig interpolant for φ and ψ in* F if

- $sig(\chi) \subseteq sig(\varphi) \cap sig(\psi)$,
- $\varphi(\boldsymbol{x}) \models \chi(\boldsymbol{x})$ and $\chi(\boldsymbol{x}) \models \psi(\boldsymbol{x})$.

We say that F has the *Craig interpolation property* (CIP) if, for all formulas $\varphi(\boldsymbol{x}), \psi(\boldsymbol{x}) \in \mathsf{F}$, whenever $\varphi \models \psi$, then there exists a Craig interpolant for φ and ψ in F.

Another language, ML, we deal with here is that of propositional modal logic with a *possibility operator* $\Diamond$. ML*-formulas* are defined by the grammar

$$\varphi \;\; := \;\; \top \;\mid\; p \;\mid\; \varphi \wedge \varphi' \;\mid\; \neg\varphi \;\mid\; \Diamond\varphi,$$

where p ranges over *propositional variables*. (The other Booleans and the *necessity operator* $\Box$ are regarded as abbreviations; e.g., $\Box\varphi$ stands for $\neg\Diamond\neg\varphi$.) In this case, by a *signature* (for ML) we mean any set σ of propositional variables; $sig(\varphi)$ is the set of all propositional variables occurring in φ. A (*Kripke*) *model* for ML takes the form $\mathfrak{M} = (W, R, \mathfrak{v})$ with a nonempty set W of 'worlds', a binary *accessibility relation* R on W, and a *valuation* $\mathfrak{v}$ mapping every propositional variable (in a given signature σ) to a subset of W. The *truth-relation* $\mathfrak{M}, u \models \varphi$, for any $\mathfrak{M}$, u and φ, is defined inductively by taking

- $\mathfrak{M}, u \models \top$,
- $\mathfrak{M}, u \models p$ iff $u \in \mathfrak{v}(p)$,
- $\mathfrak{M}, u \models \varphi \wedge \psi$ iff $\mathfrak{M}, u \models \varphi$ and $\mathfrak{M}, u \models \psi$,
- $\mathfrak{M}, u \models \neg\varphi$ iff $\mathfrak{M}, u \not\models \varphi$,
- $\mathfrak{M}, u \models \Diamond\varphi$ iff $\mathfrak{M}, v \models \varphi$, for some $v \in W$ with uRv.

Following standard practice, we often regard a Kripke model $\mathfrak{M} = (W, R, \mathfrak{v})$ as a σ-structure $\mathfrak{A}$ with σ containing the binary predicate R and a unary predicate p, for each propositional variable p, such that $\mathsf{dom}(\mathfrak{A}) = W$, $R^{\mathfrak{A}} = R$, and $p^{\mathfrak{A}} = \mathfrak{v}(p)$, for all p. Conversely, for such σ, one can regard any σ-structure as a Kripke model. In what follows, we freely use ML-formulas to talk about σ-structures and FO-formulas to talk about Kripke models. Moreover,

ML-formulas can be regarded as FO-formulas with two individual variables $var = \{x, y\}$: the *standard translation* [19] of an ML-formula φ is an FO-formula φ^*_x with one free variable $x \in var$ that is defined inductively by taking

$$\top^*_x = (x = x),\ \ p^*_x = p(x),\ \ (\neg\varphi)^*_x = \neg\varphi^*_x,\ \ (\varphi\wedge\psi)^*_x = \varphi^*_x \wedge \psi^*_x,\ \ (\Diamond\varphi)^*_x = \exists\bar{x}\,(R(x,\bar{x})\wedge\varphi^*_{\bar{x}}),$$
$$\text{where } \bar{x} = y,\ \bar{y} = x.$$

Then $\mathfrak{M}, w \models \varphi$ iff $\mathfrak{M} \models \varphi^*_x(w)$, for all pointed models $\mathfrak{M}, w$ and ML-formulas φ. Note that $\Diamond$ is a logical symbol in ML, and so does not belong to $sig(\varphi)$, for any ML-formula φ. However, the standard translation converts $\Diamond$ to a non-logical symbol, R, which may belong to $sig(\varphi^*_x)$. Fortunately, this discrepancy does not have any effect on the results for modal logics with a *single* $\Diamond$ we consider in this chapter. The situation is different for logics with *multiple* modalities.

The set of ML-formulas such that $\mathfrak{M}, w \models \varphi$, for all models $\mathfrak{M}$ and all worlds $w \in W$, is often called the *basic* (or *minimal*) *normal modal logic* and denoted by K. In general, a *normal modal logic* L is a subset of ML that contains K and is closed under the rules of modus ponens, necessitation ($\varphi \in L$ implies $\Box\varphi \in L$), and uniform substitution of arbitrary ML-formulas in place of propositional variables. If $\mathfrak{M}, w \models \varphi$, for all $\varphi \in L$ and $w \in W$, we call $\mathfrak{M}$ a *model* for L and write $\mathfrak{M} \models L$. Every normal modal logic $L \subseteq$ ML is known (see, e.g., [28]) to be determined by the class of its models in the sense that

$$\varphi \notin L \quad \text{iff} \quad \text{there exist } \mathfrak{M} \models L \text{ and } w \in W \text{ such that } \mathfrak{M}, w \not\models \varphi.$$

Given a set $\Gamma \subseteq$ ML, we write $\Gamma \models_L \psi$ if $\mathfrak{M}, w \models \Gamma$ implies $\mathfrak{M}, w \models \psi$, for all $\mathfrak{M} \models L$ and $w \in W$. Instead of $\{\varphi\} \models_L \psi$, we write simply $\varphi \models_L \psi$. The consequence relation $\models_L$ is known to be *compact*: if $\Gamma \models_L \varphi$, then there exists a finite $\Gamma' \subseteq \Gamma$ with $\Gamma' \models_L \varphi$. Note also that $\varphi \models_L \psi$ iff $\varphi \to \psi \in L$.

The definition of Craig interpolants and CIP for a normal modal logic L follows Definition 1 for FO-fragments with $\models$ replaced by $\models_L$:[1]

▶ **Definition 2.** Let L be a normal modal logic and $\varphi, \psi \in$ ML. An ML-formula χ is an *L-interpolant for φ and ψ* if $sig(\chi) \subseteq sig(\varphi) \cap sig(\psi)$, $\varphi \models_L \chi$, and $\chi \models_L \psi$. We say that L has the CIP if, for all formulas $\varphi, \psi \in$ ML, whenever $\varphi \models_L \psi$, then there is an L-interpolant for φ and ψ.

Later on in this chapter, we also consider the extensions of FO and ML with constructs for counting (counting quantifiers $\exists^{\geq k}x$ and graded modalities $\Diamond^{\geq k}$) and recursion (second-order quantifiers over unary predicates and least fixed point operators).

[1] There are other meaningful definitions of Craig interpolants; see Chapter 3 and, e.g., [44]. In particular, one can also consider interpolants for the global consequence relation.

3 Interpolant Existence in Logics without Craig Interpolation Property

If a logic L has the CIP, the existence of a Craig interpolant for formulas φ and ψ reduces in constant time to deciding whether $\varphi \models_L \psi$ holds (or, equivalently, whether the implication $\varphi \to \psi$ is valid in L). Although many standard logics do enjoy the CIP, there are countless other logics lacking it, for which this reduction to validity does not work. Of course, this does not mean that Craig interpolants in the latter category are of no interest or do not exist in the majority of interesting cases.[2] Hence we are facing the following decision problem:

interpolant existence problem (IEP) for L: given L-formulas φ and ψ, decide whether φ and ψ have a Craig interpolant in L.

The IEP is meaningful only for those L that do not have the CIP. It has so far been investigated for the following logics L (see Table 1 for a summary of the obtained results on the computational complexity of the IEP for L):

- propositional modal logics determined by linear frames such as S4.3 and K4.3 [68], the logic wK4 (weak K4) of derivative spaces [69] and Alt_n of n-ary trees (for $n \geq 3$) [57], whose lack of the CIP was shown in [62, 44, 109, 57];
- propositional linear temporal logic LTL, to be discussed in Section 6, whose lack of the CIP was observed in [74];
- modal logics with nominals and more expressive decidable hybrid logics [7], which do not have the CIP as shown in [26, 65];
- decidable fragments of FO such as the two variable fragment FO^2 and the guarded fragment GF [60], which do not have the CIP [32, 87, 77, 3, 2, 53];
- decidable fragments of first-order modal logics with constant domains such as $\mathsf{Q}^1\mathsf{S5}$, the one-variable fragment of quantified S5 [67], whose lack of CIP was shown in [41].

In this section, we give a brief introduction to model-theoretic approaches to the IEP using characterisations of logical indistinguishability of models by means of appropriate bisimulations. We focus on propositional modal logics with and without nominals. Algebraic approaches used for linear temporal logic LTL are discussed in Sections 5 and 6.

3.1 Interpolant Existence in Normal Modal Logics

There are a continuum of normal modal logics; continuum-many of them enjoy the CIP and as many lack it. No algorithm can decide whether a finitely axiomatisable modal logic has the CIP, even among extensions of the modal logic K4 (determined by transitive models). On the other hand, there are at most 36 logics with the CIP out of a continuum of extensions of S4 (determined by transitive and reflexive models), and they can be effectively recognised

[2] We refer the reader to Chapter 15 for a discussion of applications of Craig interpolants for logics without CIP in knowledge representation and reasoning.

Table 1 Complexity of validity and the IEP for some modal and first-order logics.

logic (without the CIP)	validity	IEP lower bound	IEP upper bound
finitely axiomatisable extensions of K4.3	coNP	coNP	coNP
wK4	PSpace	coNExpTime	coN3ExpTime
Alt_n, for $n \geq 3$	PSpace	coNExpTime	coNExpTime
LTL	PSpace	PSpace	4ExpTime
$\mathsf{K}^{\mathsf{nom}}$	PSpace	coNExpTime	coNExpTime
FO^2	coNExpTime	2ExpTime	coN2ExpTime
GF	2ExpTime	3ExpTime	3ExpTime
$\mathsf{Q}^1\mathsf{S5}$	coNExpTime	2ExpTime	coN2ExpTime

given the axioms of a logic. The reader can find these and other related results in Chapter 3 and [44, 28]. Here, our concern is deciding the IEP for logics without the CIP.

Denote by $\mathsf{ML}(\sigma)$ the set of ML-formulas in a signature σ, and let $\mathfrak{M}_i = (W_i, R_i, \mathfrak{v}_i)$, $i = 1, 2$, be two models with some $w_i \in W_i$. The pointed models $\mathfrak{M}_1, w_1$ and $\mathfrak{M}_2, w_2$ are called *σ-indistinguishable in* ML when $\mathfrak{M}_1, w_1 \models \varphi$ iff $\mathfrak{M}_2, w_2 \models \varphi$, for all $\varphi \in \mathsf{ML}(\sigma)$, in which case we write $\mathfrak{M}_1, w_1 \equiv_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$. This notion gives the following criterion of the IEP for normal modal logics:

▶ **Theorem 3.** *For any normal modal logic L and any* ML*-formulas φ and ψ, the following conditions are equivalent*:

- *there does not exist an L-interpolant for φ and ψ*;
- *there exist two pointed models $\mathfrak{M}_i, w_i$, $i = 1, 2$, for L such that $\mathfrak{M}_1, w_1 \models \varphi$, $\mathfrak{M}_2, w_2 \models \neg\psi$ and $\mathfrak{M}_1, w_1 \equiv_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$, where $\sigma = sig(\varphi) \cap sig(\psi)$.*

In particular, one can take the $\mathfrak{M}_i$ to be the canonical model for L [28, 19].

Proof. ($\Leftarrow$) Let $\mathfrak{M}_1, w_1 \models \varphi$, $\mathfrak{M}_2, w_2 \models \neg\psi$ and $\mathfrak{M}_1, w_1 \equiv_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$. Suppose χ is an L-interpolant for φ and ψ. Then $\varphi \models_L \chi$, and so $\mathfrak{M}_1, w_1 \models \chi$. As $sig(\chi) \subseteq \sigma$ and the $\mathfrak{M}_i, w_i$ are σ-indistinguishable, we have $\mathfrak{M}_2, w_2 \models \chi$, and so $\mathfrak{M}_2, w_2 \models \psi$, which is a contradiction. It follows that φ and ψ do not have an L-interpolant.

($\Rightarrow$) Conversely, suppose there is no L-interpolant for φ and ψ. Consider the set

$$\Phi^\sigma = \{\chi \in \mathsf{ML}(\sigma) \mid \varphi \models_L \chi\}.$$

By compactness, we have $\Phi^\sigma \not\models_L \psi$, for otherwise there would be a finite $\Phi \subseteq \Phi^\sigma$ with $\Phi \models_L \psi$, and so $\bigwedge \Phi$ would be an L-interpolant for φ and ψ. Then there is a model $\mathfrak{M}_2$ (say, the canonical model) for L such that $\mathfrak{M}_2, w_2 \models \Phi^\sigma \cup \{\neg\psi\}$, for some $w_2 \in W_2$. Let

$$\mathfrak{t}^\sigma_{\mathfrak{M}_2}(w_2) = \{\chi \in \mathsf{ML}(\sigma) \mid \mathfrak{M}_2, w_2 \models \chi\}.$$

Using compactness again, we find a model $\mathfrak{M}_1$ for L with $\mathfrak{M}_1, w_1 \models \mathfrak{t}^{\sigma}_{\mathfrak{M}_2}(w_2) \cup \{\varphi\}$, for some $w_1 \in W_1$ (because otherwise we would have a finite set $\Psi \subseteq \mathfrak{t}^{\sigma}_{\mathfrak{M}_2}(w_2)$ with $\varphi \models_L \neg \bigwedge \Psi$, which means that $\neg \bigwedge \Psi \in \Phi^{\sigma} \subseteq \mathfrak{t}^{\sigma}_{\mathfrak{M}_2}(w_2)$, and so $\mathfrak{t}^{\sigma}_{\mathfrak{M}_2}(w_2)$ would be unsatisfiable). Thus, we have $\mathfrak{M}_1, w_1 \models \varphi$, $\mathfrak{M}_2, w_2 \models \neg\psi$, and $\mathfrak{M}_1, w_1 \equiv_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$. ◀

The criterion of Theorem 3 is difficult to use for deciding the IEP as it quantifies over infinitely many σ-formulas. One can make it more 'operational' by replacing $\equiv_{\mathsf{ML}(\sigma)}$ with a model-theoretic characterisation in terms of σ-bisimulations to be introduced below.

Let again $\mathfrak{M}_i = (W_i, R_i, \mathfrak{v}_i)$, $i = 1, 2$, be models for L and let σ be a signature. A relation $\boldsymbol{\beta} \subseteq W_1 \times W_2$ is called a *σ-bisimulation* between $\mathfrak{M}_1$ and $\mathfrak{M}_2$ if the following conditions are satisfied whenever $u_1 \boldsymbol{\beta} u_2$:

- $\mathfrak{M}_1, u_1 \models p$ iff $\mathfrak{M}_2, u_2 \models p$, for all propositional variables $p \in \sigma$;
- if $u_1 R_1 v_1$, then there is v_2 such that $u_2 R_2 v_2$ and $v_1 \boldsymbol{\beta} v_2$; and, conversely, if $u_2 R_2 v_2$, then there is v_1 with $u_1 R_1 v_1$ and $v_1 \boldsymbol{\beta} v_2$.

If there is such a σ-bisimulation $\boldsymbol{\beta}$ with $w_1 \boldsymbol{\beta} w_2$, we write $\mathfrak{M}_1, w_1 \sim_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$. A remarkable property of the canonical models (and also of ω-saturated σ-structures) $\mathfrak{M}_L$ for L is that $\mathfrak{M}_L, u \equiv_{\mathsf{ML}(\sigma)} \mathfrak{M}_L, v$ iff $\mathfrak{M}_L, u \sim_{\mathsf{ML}(\sigma)} \mathfrak{M}_L, v$, for any u and v in $\mathfrak{M}$. This yields the operational criterion we are after, see also Chapter 4 and Chapter 15:

▶ **Theorem 4.** *For any normal modal logic L and any* ML*-formulas φ and ψ, the following conditions are equivalent*:

- *there does not exist an L-interpolant for φ and ψ*;
- *there exist two pointed models $\mathfrak{M}_i, w_i$, $i = 1, 2$, for L such that $\mathfrak{M}_1, w_1 \models \varphi$, $\mathfrak{M}_2, w_2 \models \neg\psi$ and $\mathfrak{M}_1, w_1 \sim_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$, where $\sigma = sig(\varphi) \cap sig(\psi)$.*

In particular, one can take the $\mathfrak{M}_i$ to be the canonical model for L.

We illustrate this criterion by showing that normal modal logics wK4, S4.3, and Alt_3 do not have the CIP [44, 62]. The first of them is called *weak* K4 [39] as it is characterised by *weakly transitive* models satisfying the condition $\forall x, y, z \in W \big(R(x, y) \land R(y, z) \to (x = z) \lor R(x, z)\big)$; see also recent [9]. S4.3 is characterised by models $\mathfrak{M} = (W, R, \mathfrak{v})$, in which R is transitive, reflexive, and *weakly connected* in the sense that

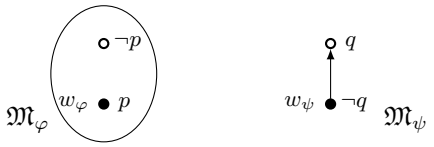
$$\forall x, y, z \in W \big(R(x, y) \land R(x, z) \to R(y, z) \lor R(z, y)\big).$$

Alt_n, for $n \geq 1$, is the logic characterised by models whose underlying frames (W, R) are (irreflexive and intransitive) directed trees of outdegree n.

In the pictures below, • denotes an irreflexive point, ∘ a reflexive one, and an ellipse represents a *cluster*, i.e., a set of points where any two distinct ones 'see' each other via the accessibility relation.

▶ **Example 5.** (*i*) Suppose $\varphi = \Diamond\Diamond p \land \neg\Diamond p$ and $\psi = \Diamond\Diamond\neg q \lor q$ with the common signature $\sigma = \emptyset$. It is easy to see that $\varphi \to \psi$ is true in all weakly transitive models, and so $\varphi \models_{\mathsf{wK4}} \psi$. On the other hand, the models $\mathfrak{M}_\varphi$ and $\mathfrak{M}_\psi$ depicted below are such that $\mathfrak{M}_\varphi, w_\varphi \models \varphi$,

$\mathfrak{M}_\psi, w_\psi \models \neg\psi$, and the *universal relation* $\boldsymbol{\beta}$ between the points of $\mathfrak{M}_\varphi$ and $\mathfrak{M}_\psi$ is a σ-bisimulation with $w_\varphi \boldsymbol{\beta} w_\psi$:

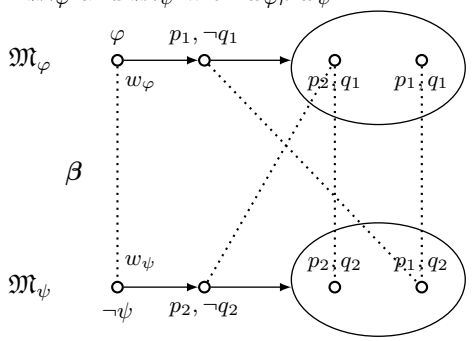


Therefore, φ and ψ do not have a wK4-interpolant, and so wK4 does not enjoy the CIP.

(*ii*) Next, consider the formulas

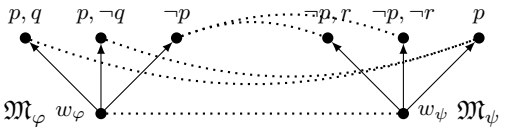
$$\varphi = \Diamond(p_1 \wedge \Diamond\neg q_1) \wedge \Box(p_2 \to \Box q_1), \quad \psi = \neg[\Diamond(p_2 \wedge \Diamond\neg q_2) \wedge \Box(p_1 \to \Box q_2)]$$

with the common signature $\sigma = \{p_1, p_2\}$. It is not hard to see that $\varphi \models_{\mathsf{S4.3}} \psi$ and that, in the picture below, $\mathfrak{M}_\varphi, w_\varphi \models \varphi$, $\mathfrak{M}_\psi, w_\psi \models \neg\psi$, and the relation $\boldsymbol{\beta}$ shown by the dotted lines is a σ-bisimulation between $\mathfrak{M}_\varphi$ and $\mathfrak{M}_\psi$ with $w_\varphi \boldsymbol{\beta} w_\psi$.



Thus, φ and ψ do not have an S4.3-interpolant, and so S4.3 does not enjoy the CIP.

(*iii*) Finally, suppose $\varphi = \Diamond(p \wedge q) \wedge \Diamond(p \wedge \neg q)$ and $\psi = \neg[\Diamond(\neg p \wedge r) \wedge \Diamond(\neg p \wedge \neg r)]$ with common signature $\sigma = \{p\}$. The reader can readily check that $\varphi \to \psi$ is true in all models based on trees of outdegree 3, and so $\varphi \models_{\mathsf{Alt}_3} \psi$. On the other hand, in the picture below, $\mathfrak{M}_\varphi, w_\varphi \models \varphi$, $\mathfrak{M}_\psi, w_\psi \models \neg\psi$, and the relation $\boldsymbol{\beta}$ shown by the dotted lines is a σ-bisimulation between $\mathfrak{M}_\varphi$ and $\mathfrak{M}_\psi$ with $w_\varphi \boldsymbol{\beta} w_\psi$. It follows that φ and ψ do not have an interpolant in Alt_3.



⌟

Theorem 4 can be used as a starting point for establishing decidability of the IEP for normal modal logics lacking the CIP and finding its computational complexity. A standard way of proving decidability of a finitely axiomatisable modal logic L is by showing (if lucky) that it has the *finite model property* (FMP): $\varphi \notin L$ iff there is a finite pointed model $\mathfrak{M}, w$ for L with $\mathfrak{M}, w \not\models \varphi$; see, e.g., [28]. Likewise, to prove decidability of the IEP for such an L, one can try to show that it has the *finite bisimilar model property* (FBMP) in the sense that φ

and ψ do not have an L-interpolant iff this is witnessed by *finite* models $\mathfrak{M}_i, w_i$ in the second item of Theorem 4. We give an example of such a proof in Section 3.2, Proposition 9. Here, we only observe the following obvious analogue of Harrop's theorem [28, Theorem 16.13]:

▶ **Lemma 6.** *If a finitely axiomatisable normal modal logic L has the FBMP, then the IEP for L is decidable.*

We would also obtain an upper complexity bound on the IEP for L if the size of the witnessing models $\mathfrak{M}_i$ could be bounded by some function in the size $|\varphi| + |\psi|$ of φ, ψ.

Establishing the FBMP of L cannot be easier than showing the FMP. We start by considering any models $\mathfrak{M}_i, w_i$ from the second item of Theorem 4 and aim to convert them into finite models $\mathfrak{M}'_i, w'_i$ meeting the same conditions. We can try and apply various filtration or maximal point techniques to each of the $\mathfrak{M}_i$, but have to coordinate the construction of the new models by keeping track of σ-bisimilarity, which requires more complex 'data structures'. For instance, one can show that $\mathsf{S4.3}$ has the polynomial FBMP using the maximal point technique [68], which implies that the IEP for $\mathsf{S4.3}$ is coNP-complete—as complex as the validity problem. On the other hand, using a quite elaborate filtration argument, [69] establishes a triple-exponential FBMP of $\mathsf{wK4}$, which has the exponential FMP and is PSpace-complete. The next theorem shows that the IEP for $\mathsf{wK4}$ and Alt_n, $n \geq 3$, is harder than validity; cf. Table 1.

▶ **Theorem 7** ([68, 69, 57])**.**

(i) *The IEP for* $\mathsf{wK4}$ *is decidable in* coN3ExpTime *while being* coNExpTime*-hard.*

(ii) *The IEP for any finitely axiomatisable logic containing* $\mathsf{K4.3}$ *is* coNP*-complete.*

(iii) *The IEP for any* Alt_n *with* $n \geq 3$ *is* coNExpTime*-complete.*

In Theorem 7 (*ii*), $\mathsf{K4.3}$ is the logic determined by weakly connected transitive models. Not all of its (continuum-many) extensions have the FBMP: a notable example is the logic determined by models based on finite strict linear orders, which has the FMP by definition. The decidability and coNP complexity upper bound are obtained by showing that witness models $\mathfrak{M}_i, w_i$ for Theorem 4 can be assembled from polynomially-many, possibly infinite, blocks with a finitely describable structure.

3.2 The Finite Bisimilar Model Property of Modal Logic with Nominals

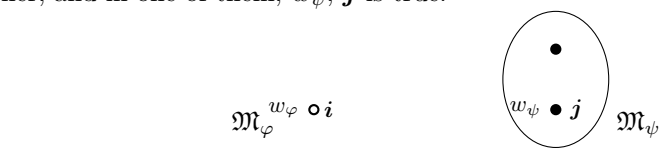
To convey the flavour of a filtration-based construction showing the FBMP, in this section we consider the basic modal logic $\mathsf{K}^{\mathsf{nom}}$ with nominals. Its formulas are given in the language $\mathsf{ML}^{\mathsf{nom}}$ extending ML with another type of atomic formulas, *nominals*, that are denoted by $\boldsymbol{i}$, $\boldsymbol{j}$ and interpreted in $\mathsf{ML}^{\mathsf{nom}}$-models $\mathfrak{M} = (W, R, \mathfrak{v})$ by singleton sets, i.e., $|\mathfrak{v}(\boldsymbol{i})| = 1$, for every nominal $\boldsymbol{i}$. Nominals are regarded as *non-logical* symbols, so they are included in $sig(\varphi)$ whenever they occur in an $\mathsf{ML}^{\mathsf{nom}}$-formula φ. The definition of a *σ-bisimulation $\boldsymbol{\beta}$* between $\mathsf{ML}^{\mathsf{nom}}$-models $\mathfrak{M}_1$ and $\mathfrak{M}_2$ requires an additional item for nominals:

- whenever $u_1 \boldsymbol{\beta} u_2$, then $\mathfrak{M}_1, u_1 \models \boldsymbol{i}$ iff $\mathfrak{M}_2, u_2 \models \boldsymbol{i}$, for all nominals $\boldsymbol{i} \in \sigma$.

We write $\mathfrak{M}_1, w_1 \sim_{\mathsf{ML}^{\mathsf{nom}}(\sigma)} \mathfrak{M}_2, w_2$ to say that there is a σ-bisimulation β between $\mathsf{ML}^{\mathsf{nom}}$-models $\mathfrak{M}_1$ and $\mathfrak{M}_2$ with $w_1 \beta w_2$. We denote by $\mathsf{K}^{\mathsf{nom}}$ the basic modal logic with nominals, which comprises those $\mathsf{ML}^{\mathsf{nom}}$-formulas that are true in all worlds of all models. The decision problem for $\mathsf{K}^{\mathsf{nom}}$ is known to be PSPACE-complete [6]. Theorems 3 and 4 easily generalise to the case $L = \mathsf{K}^{\mathsf{nom}}$ and $\mathsf{ML}^{\mathsf{nom}}$ in place of ML.

▶ **Proposition 8.** $\mathsf{K}^{\mathsf{nom}}$ *does not have the CIP.*

Proof. Consider the formulas $\varphi = \boldsymbol{i} \wedge \Diamond \boldsymbol{i}$ and $\psi = \boldsymbol{j} \to \Diamond \boldsymbol{j}$ with distinct nominals $\boldsymbol{i}$ and $\boldsymbol{j}$, so $\sigma = sig(\varphi) \cap sig(\psi) = \emptyset$. As $\mathfrak{M}, u \models \varphi$ means that u is reflexive, $\varphi \models_{\mathsf{K}^{\mathsf{nom}}} \psi$. Let $\mathfrak{M}_\varphi$ consist of one reflexive world w_φ, where $\boldsymbol{i}$ is true, and let $\mathfrak{M}_\psi$ consist of two irreflexive worlds seeing each other, and in one of them, w_ψ, $\boldsymbol{j}$ is true:



The universal relation between these two models is a σ-bisimulation, $\mathfrak{M}_\varphi, w_\varphi \models \varphi$ and $\mathfrak{M}_\psi, w_\psi \models \neg\psi$. Therefore, by Theorem 4, φ and ψ do not have a Craig interpolant in $\mathsf{K}^{\mathsf{nom}}$. ◀

▶ **Proposition 9.** $\mathsf{K}^{\mathsf{nom}}$ *has the FBMP.*

Proof. We first remind the reader of a standard filtration proof showing that $\mathsf{K}^{\mathsf{nom}}$ has the FMP and then generalise it to a proof of the FBMP. Let $sub(\varphi)$ be the set of all subformulas of an $\mathsf{ML}^{\mathsf{nom}}$-formula φ together with their negations. By the φ-*type* of a world w in a model $\mathfrak{M} = (W, R, \mathfrak{v})$ we mean the set

$$\mathfrak{t}_{\mathfrak{M}}(w) = \{\psi \in sub(\varphi) \mid \mathfrak{M}, w \models \psi\}.$$

Suppose $\mathfrak{M}, w \models \varphi$. Define a new finite model $\mathfrak{M}' = (W', R', \mathfrak{v}')$ by taking

- $W' = \{\mathfrak{t}_{\mathfrak{M}}(u) \mid u \in W\}$;
- $\mathfrak{t}_{\mathfrak{M}}(u) \in \mathfrak{v}'(p)$ iff $p \in \mathfrak{t}_{\mathfrak{M}}(u)$, for all propositional variables p and all $u \in W$;
- $\mathfrak{t}_{\mathfrak{M}}(u) \in \mathfrak{v}'(\boldsymbol{i})$ iff $\boldsymbol{i} \in \mathfrak{t}_{\mathfrak{M}}(u)$, for all nominals $\boldsymbol{i}$ and all $u \in W$;
- $(\mathfrak{t}, \mathfrak{t}') \in R'$ iff there are $u, v \in W$ with $(u, v) \in R$, $\mathfrak{t}_{\mathfrak{M}}(u) = \mathfrak{t}$ and $\mathfrak{t}_{\mathfrak{M}}(v) = \mathfrak{t}'$.

One can show by induction on φ that $\mathfrak{M}', \mathfrak{t}_{\mathfrak{M}}(u) \models \bigwedge \mathfrak{t}_{\mathfrak{M}}(u)$ for all $u \in W$, so $\mathfrak{M}', \mathfrak{t}_{\mathfrak{M}}(w) \models \varphi$.

To generalise this filtration construction to σ-bisimilar models, suppose $\sigma = sig(\varphi_1) \cap sig(\varphi_2)$ and we have $\mathfrak{M}_1, w_1 \sim_{\mathsf{ML}^{\mathsf{nom}}(\sigma)} \mathfrak{M}_2, w_2$ and $\mathfrak{M}_l, w_l \models \varphi_l$, $l = 1, 2$. Our aim is to convert the $\mathfrak{M}_l$ into finite models, denoted $\mathfrak{M}'_l$, with the same properties. We define the (φ_1, φ_2)-*bisimulation type* of $u \in W_l$, $l = 1, 2$, as the triple $\mathfrak{bt}_{\mathfrak{M}_l}(u)$ defined by taking

$$\mathfrak{bt}_{\mathfrak{M}_l}(u) = \big(\mathfrak{t}_{\mathfrak{M}_l}(u), \mathfrak{b}_{\mathfrak{M}_1}(u), \mathfrak{b}_{\mathfrak{M}_2}(u)\big),$$
$$\mathfrak{t}_{\mathfrak{M}_l}(u) = \{\psi \in sub(\varphi_1) \cup sub(\varphi_2) \mid \mathfrak{M}_l, u \models \psi\},$$
$$\mathfrak{b}_{\mathfrak{M}_k}(u) = \{\mathfrak{t}_{\mathfrak{M}_k}(v) \mid \mathfrak{M}_l, u \sim_{\mathsf{ML}^{\mathsf{nom}}(\sigma)} \mathfrak{M}_k, v\}, \quad \text{for } k \in \{1, 2\}.$$

Note that now the type $\mathfrak{t}_{\mathfrak{M}_l}(u)$ consists of subformulas of both φ_1 and φ_2. Define finite models $\mathfrak{M}'_l = (W'_l, R'_l, \mathfrak{v}'_l)$, $l = 1, 2$, by taking

- $W'_l = \{\mathfrak{bt}_{\mathfrak{M}_l}(u) \mid u \in W_l\}$;
- $\mathfrak{bt}_{\mathfrak{M}_l}(u) \in \mathfrak{v}'_l(p)$ iff $p \in \mathfrak{t}_{\mathfrak{M}_l}(u)$, for all propositional variables p and all $u \in W_l$;
- $\mathfrak{bt}_{\mathfrak{M}_l}(u) \in \mathfrak{v}'_l(\boldsymbol{i})$ iff $\boldsymbol{i} \in \mathfrak{t}_{\mathfrak{M}_l}(u)$, for all nominals $\boldsymbol{i}$ and all $u \in W_l$;
- $(\mathfrak{bt}, \mathfrak{bt}') \in R'_l$ iff there exist worlds $u, v \in W_l$ such that $(u, v) \in R_l$, $\mathfrak{bt}_{\mathfrak{M}_l}(u) = \mathfrak{bt}$ and $\mathfrak{bt}_{\mathfrak{M}_l}(v) = \mathfrak{bt}'$.

Then the following hold:

▶ **Lemma 10.**

(i) $\mathfrak{M}'_l, \mathfrak{bt}_{\mathfrak{M}_l}(u) \models \bigwedge \mathfrak{t}_{\mathfrak{M}_l}(u)$, *for all* $u \in W_l$ *with* $l = 1, 2$.

(ii) $\mathfrak{M}'_1, (\mathfrak{t}, \mathfrak{b}_1, \mathfrak{b}_2) \sim_{\mathsf{ML}^{\mathsf{nom}}(\sigma)} \mathfrak{M}'_2, (\mathfrak{t}', \mathfrak{b}_1, \mathfrak{b}_2)$, *for all* $(\mathfrak{t}, \mathfrak{b}_1, \mathfrak{b}_2) \in W'_1$ *and* $(\mathfrak{t}', \mathfrak{b}_1, \mathfrak{b}_2) \in W'_2$.

Proof. The first claim is shown by proving inductively that $\mathfrak{M}'_l, \mathfrak{bt}_{\mathfrak{M}_l}(u) \models \chi$ iff $\mathfrak{M}, u \models \chi$, for all $\chi \in \mathit{sub}(\varphi_1) \cup \mathit{sub}(\varphi_2)$ and all $u \in W_l$, which is straightforward. For the second claim, we define $\boldsymbol{\beta} \subseteq W_1 \times W_2$ by setting $((\mathfrak{t}, \mathfrak{b}_1, \mathfrak{b}_2), (\mathfrak{t}', \mathfrak{b}'_1, \mathfrak{b}'_2)) \in \boldsymbol{\beta}$ if $\mathfrak{b}_1 = \mathfrak{b}'_1$ and $\mathfrak{b}_2 = \mathfrak{b}'_2$, for all $(\mathfrak{t}, \mathfrak{b}_1, \mathfrak{b}_2) \in W'_1$ and $(\mathfrak{t}', \mathfrak{b}'_1, \mathfrak{b}'_2) \in W'_2$. We show that $\boldsymbol{\beta}$ is a σ-bisimulation between $\mathfrak{M}'_1$ and $\mathfrak{M}'_2$. Assume $(\mathfrak{t}, \mathfrak{b}_1, \mathfrak{b}_2) \in W_1$ and $(\mathfrak{t}', \mathfrak{b}_1, \mathfrak{b}_2) \in W_2$ are given. Observe that all types in $\mathfrak{b}_1 \cup \mathfrak{b}_2$ contain the same propositional variables and nominals in σ, since they are types realised in σ-bisimilar worlds in $\mathfrak{M}_1$ and $\mathfrak{M}_2$, respectively. Hence $(\mathfrak{t}, \mathfrak{b}_1, \mathfrak{b}_2)$ and $(\mathfrak{t}', \mathfrak{b}_1, \mathfrak{b}_2)$ satisfy the same propositional variables and nominals from σ in $\mathfrak{M}'_1$ and $\mathfrak{M}'_2$, respectively. Now assume that $\mathfrak{bt} = (\mathfrak{t}, \mathfrak{b}_1, \mathfrak{b}_2)$, $\mathfrak{bt}^* = (\mathfrak{t}^*, \mathfrak{b}^*_1, \mathfrak{b}^*_2)$, $\mathfrak{bt}' = (\mathfrak{t}', \mathfrak{b}_1, \mathfrak{b}_2)$, and $(\mathfrak{bt}, \mathfrak{bt}^*) \in R'_1$. By definition, there are $u, v \in W_1$ with uR_1v, $\mathfrak{bt} = \mathfrak{bt}_{\mathfrak{M}_1}(u)$, and $\mathfrak{bt}^* = \mathfrak{bt}_{\mathfrak{M}_1}(v)$. Also by definition, there exists $u' \in W_2$ with $\mathfrak{t}_{\mathfrak{M}_2}(u') = \mathfrak{t}'$ and $\mathfrak{M}_1, u \sim_{\mathsf{ML}^{\mathsf{nom}}(\sigma)} \mathfrak{M}_2, u'$. By the definition of σ-bisimulations, there exists v' such that $u'R_2v'$ and $\mathfrak{M}_1, v \sim_{\mathsf{ML}^{\mathsf{nom}}(\sigma)} \mathfrak{M}_2, v'$. Then we have $(\mathfrak{bt}^*, \mathfrak{bt}_{\mathfrak{M}_2}(v')) \in \boldsymbol{\beta}$ and $(\mathfrak{bt}', \mathfrak{bt}_{\mathfrak{M}_2}(v')) \in R'_2$, as required. The converse condition is shown in the same way. ◀

Returning to the proof of Proposition 9, we obtain by Lemma 10 that $\mathfrak{M}'_l, \mathfrak{bt}_{\mathfrak{M}_l}(u) \models \varphi_l$ and $\mathfrak{M}'_1, \mathfrak{bt}_{\mathfrak{M}_1}(w_1) \sim_{\mathsf{ML}^{\mathsf{nom}}(\sigma)} \mathfrak{M}'_2, \mathfrak{bt}_{\mathfrak{M}_2}(w_2)$, as required.

Note that rather than building models directly from types we now build σ-bisimilar models from sets $\mathfrak{b}_{\mathfrak{M}_k}(u)$ of types satisfied in σ-bisimilar nodes. The constructed models are of double-exponential size in $|\varphi_1| + |\varphi_2|$. ◀

The obtained double-exponential FBMP gives an obvious coN2ExpTime algorithm deciding the IEP for $\mathsf{K}^{\mathsf{nom}}$. This algorithm is not optimal. In fact, by unfolding the bisimilar models into bisimilar forest-shaped models (except for worlds denoted by nominals) and cutting them off at depth coinciding with the maximal modal depth of φ and ψ, one can establish the exponential FBMP, which yields the following result:

▶ **Theorem 11** ([7]). *The IEP for* $\mathsf{K}^{\mathsf{nom}}$ *is* coNExpTime*-complete.*

We illustrate the reason why the complexity of the IEP for $\mathsf{K}^{\mathsf{nom}}$ is higher than the complexity of its decision problem by sketching the idea of the lower bound construction. The

proof is by reduction of the NExpTime-complete $2^n \times 2^n$-tiling problem to the complement of the IEP for $\mathsf{K}^{\mathsf{nom}}$. Given any $n < \omega$ and a finite set T of tile types, we define formulas $\varphi_{n,T}$ and $\psi_{n,T}$ of polynomial size in $|T|$ and n such that $\varphi_{n,T} \to \psi_{n,T} \in \mathsf{K}^{\mathsf{nom}}$ and T can tile a $2^n \times 2^n$-grid iff $\varphi_{n,T}$ and $\psi_{n,T}$ do not have an interpolant in $\mathsf{K}^{\mathsf{nom}}$. The common signature $\sigma_{n,T}$ of $\varphi_{n,T}$ and $\psi_{n,T}$ contains $O(n)$-many variables that are used to speak about binary representations of cells in the $2^n \times 2^n$ grid; $\sigma_{n,T}$ also contains variables for the tile types in T. The formula $\varphi_{n,T}$ contains the conjunct $\Diamond^{2n}\boldsymbol{i} \wedge \Box^{2n}\boldsymbol{i}$, which is true in a pointed model $\mathfrak{M}, w$ iff the only world accessible from w in $2n$ steps in $\mathfrak{M}$ is where nominal $\boldsymbol{i}$ holds. The formula $\psi_{n,T}$ contains standard conjuncts with $O(n)$-many non-$\sigma_{n,T}$ variables that generate a full binary tree of depth $2n$ [28, 19] whose leaves $u_1, \dots, u_{2^{2n}}$ are regarded as the cells of the grid. Now, suppose that $\mathfrak{M}_1, w_1 \sim_{\mathsf{ML}^{\mathsf{nom}}(\sigma_n)} \mathfrak{M}_2, w_2$, $\mathfrak{M}_1, w_1 \models \varphi_{n,T}$ and $\mathfrak{M}_2, w_2 \models \psi_{n,T}$. Then all the leaves u_l of that full binary tree of depth $2n$ in $\mathfrak{M}_2$ are $\sigma_{n,T}$-bisimilar to the world where $\boldsymbol{i}$ is true in $\mathfrak{M}_1$, and so all trees with the roots $u_1, \dots, u_{2^{2n}}$ in $\mathfrak{M}_2$ are $\sigma_{n,T}$-bisimilar to each other. The other conjuncts of $\psi_{n,T}$ make sure that each tree rooted at u_l describes the tile types of the grid cell represented by u_l and of its neighbours in the grid; the $\sigma_{n,T}$-bisimilarity of these trees ensures that these local descriptions are consistent.

Further Reading. The IEP has been investigated in the context of learning description logic concepts from positive and negative data examples [59, 7]. Logics with nominals are particularly important in this application of interpolants as they are needed to encode constants used in description logic knowledge bases. We refer the reader to Chapter 15 for a detailed discussion. It is also of interest to note that no logic with the CIP that extends $\mathsf{K}^{\mathsf{nom}}$ and satisfies certain natural closure conditions is decidable [26]. Hence, the CIP cannot be restored easily by adding new connectives to $\mathsf{K}^{\mathsf{nom}}$ without sacrificing decidability.

3.3 Interpolant Existence in Fragments of First-Order Logic

Locating expressive and yet decidable fragments of FO has been a popular research topic for more than a century [23, 92]. It turns out, however, that, unlike full FO, many of its prominent decidable fragments do not enjoy the CIP, and for these, the question arises as to whether the IEP is decidable, and, if so, whether its complexity is the same as that of entailment. We refer the reader to Chapter 2 and [27] for a detailed discussion of the landscape of fragments of FO with and without the CIP.

The investigation of the IEP for FO-fragments without the CIP has only just begun. Two prominent fragments of FO that lack the CIP are the two-variable fragment FO^2 and the guarded fragment GF. In FO^2, one restricts the number of individual variables in formulas to two. By re-using variables, one can show that ML (and even ML with nominals) are fragments of FO^2 under the standard translation [19] (see Section 2). Satisfiability of formulas in FO^2 is NExpTime-complete, which can be shown by proving its exponential FMP [23, 92]. By generalising the technique discussed in the previous section for $\mathsf{K}^{\mathsf{nom}}$, one can show that FO^2 has the double-exponential FBMP, for an appropriate notion of bisimulation characterising

indistinguishability of models in FO^2. In contrast to the exponential FBMP of $\mathsf{K}^{\mathsf{nom}}$ and the exponential FMP of FO^2, this double exponential bound is optimal.

▶ **Theorem 12** ([60]). *The IEP for* FO^2 *is in* coN2ExpTime *and* 2ExpTime*-hard.*

The investigation of decidable extensions of FO^2 has been an important research topic, with results ranging from coNExpTime-completeness of FO^2 with *counting quantifiers* $\exists^{\geq k} x$, denoted C^2, to the decidability of FO^2 extended with one (built-in) transitive relation and undecidability of FO^2 with two transitive relations [92]. At least one such extension provides an example with decidable entailment and undecidable IEP, namely FO^2 extended with two (built-in) equivalence relations whose entailment is co2NExpTime-complete.

▶ **Theorem 13** ([110]). *The IEP for* FO^2 *with two equivalence relations is undecidable.*

The decidability of the IEP for other extensions of FO^2 such as C^2 and FO^2 with one equivalence relation remains open.

Intermezzo

FO^2 is closely related to prominent decidable fragments of first-order modal logics with constant first-order domains, another family of logics lacking the CIP [41]. In fact, the one-variable fragment $\mathsf{Q}^1\mathsf{S5}$ of first-order modal logic $\mathsf{S5}$ can be regarded as a fragment of FO^2. Exactly the same complexity bounds as in Theorem 12 for FO^2 are known for this logic, but with subtly different proofs [67]. In particular, the lower bound proof is more involved as it cannot use equality. Actually, this proof can be used to strengthen Theorem 12 since 2ExpTime-hardness of the IEP for FO^2 even without equality can be shown as a corollary. The IEP of the one-variable fragment of first-order modal logic K is also decidable, but in this case no elementary algorithm is known [67].

The guarded fragment GF of FO restricts quantification to guarded quantifiers of the form $\exists \boldsymbol{x}\,(R(\boldsymbol{x},\boldsymbol{y}) \wedge \varphi(\boldsymbol{x},\boldsymbol{y}))$. It also generalises modal logic and is decidable in ExpTime if the arity of predicate symbols is fixed and 2ExpTime-complete otherwise. While its two-variable fragment, $\mathsf{FO}^2 \cap \mathsf{GF}$ enjoys the CIP, GF itself does not. Regarding the IEP, it is open whether it has the FBMP for the appropriate notion of bisimulation called *guarded bisimulation*, but one can use instead models of bounded treewidth constructed from sets of types satisfied in guarded bisimilar tuples to show the following:

▶ **Theorem 14** ([60]). *The IEP for* GF *is* 2ExpTime*-complete if the arity of predicates is bounded and* 3ExpTime*-complete otherwise.*

Further Reading. We close our discussion of the IEP with a few comments on Horn logics. While the Horn fragment of FO enjoys the CIP [29, Exercise 6.2.6], Horn fragments of modal and description logics typically do not enjoy the CIP. The IEP for these logics is investigated in [42], using a model-theoretic approach. Results on the exact reformulation of conjunctive queries modulo tuple-generating dependencies can also be formulated as results about the IEP in rule-based languages lacking the CIP [15, 14].

4 Interpolation, Separation and Definability in Weaker Languages

In this section, we discuss a generalisation of Craig interpolation, which restricts the *logical constructs* of the language for interpolants between formulas rather than the *vocabulary* of non-logical symbols. All of our languages are assumed to be sublanguages of *monadic second-order logic* MSO that extends FO with quantifiers of the form $\exists P$, where P is any unary predicate symbol (thus, $\exists x$ quantifies over the elements of a domain in question, while $\exists P$ over the subsets of that domain). To simplify presentation, we also assume that our languages are closed under negation and that their signature comprises one binary predicate symbol, R, and arbitrarily-many unary ones. Unless indicated otherwise, we only consider formulas $\varphi(x)$ with at most one free variable x. Modal formulas are often regarded as their standard translations into FO or MSO.

For any such languages $L \supseteq L'$, we are interested in the following decision problem:

L/L'-interpolation: given two L-formulas φ and ψ, decide whether there exists an L'-formula χ—called an *L'-interpolant* for φ and ψ—such that $\varphi \models \chi$ and $\chi \models \psi$.

Note that, in this definition, we do not require the interpolant χ to be constructed from the shared non-logical symbols of φ and ψ. In various areas of computer science, the interpolation problem is often formulated as a separation problem, which is sometimes more convenient to work with (see, e.g., Sections 5 and 6 below):

L/L'-separation: given L-formulas φ and ψ, decide whether there is an L'-formula χ— called an *L'-separator* for φ and ψ—such that $\varphi \models \chi$ and $\chi \models \neg\psi$.

Clearly, these two problems are equivalent to each other because L is closed under (classical) negation.

▶ **Example 15.** Consider the FO-formulas

$$\varphi(x) = \exists^{=1} y\, R(x,y), \quad \psi(x) = \exists^{=1} y \big(R(x,y) \land A(y)\big) \land \exists^{=1} y \big(R(x,y) \land \neg A(y)\big)$$

(where $\exists^{=1} y\, \alpha(y)$ says that there exists exactly one y for which $\alpha(y)$ holds). Clearly, $\varphi \models \neg\psi$ and the ML-formula $\chi = \Box A \lor \Box \neg A$ separates $\varphi(x)$ and $\psi(x)$. For $\psi'(x) = \exists^{=2} y\, R(x,y)$, we also have $\varphi \models \neg\psi'$, but $\varphi(x)$ and $\psi'(x)$ are not ML-separable as follows from Theorem 17 below. ⌟

L/L'-interpolation and separation are in turn generalisations of the more familiar

L/L'-definability: given an L-formula φ, decide whether there exists an L'-formula χ such that $\varphi \models \chi$ and $\chi \models \varphi$.

▶ **Example 16.** The FO-formulas

$$\varphi_n(x) = \exists y \big[R(x,y) \land \bigwedge_{1 \le i \le n} \big(A_i(x) \leftrightarrow A_i(y)\big)\big], \quad n < \omega,$$

are ML-definable. Indeed, consider the set T of all conjunctions $B_1 \wedge \cdots \wedge B_n$ such that $B_i \in \{A_i, \neg A_i\}$, for $i = 1, \ldots, n$. Then each φ_n is equivalent to $\chi_n = \bigvee_{t \in T}(t \wedge \Diamond t)$. It might be of interest to note that these χ_n are of exponential-size in n and there are no ML-formulas χ_n that are equivalent to φ_n and have polynomial-size $|sub(\chi_n)|$ [66]. ⌟

Below, we discuss definability and separation in fragments of FO, including various types of modal logic, and in logics with second-order quantification or recursion such as MSO and the modal μ-calculus μML. While model-theoretic methods based on bisimulations are prominent for fragments of FO, they have to be augmented by automata theoretic methods when dealing with second-order quantifiers. We also introduce an elegant 'good model method' that has been used to study definability and construct defining formulas whenever they exist.

4.1 Definability and Separation in Fragments of First-Order Logic

The next theorem is an analogue of Theorem 3, characterising the existence of L'-separators in terms of L'-indistinguishable structures.

▶ **Theorem 17.** *For any $L' \subseteq L \subseteq$ FO and any L-formulas $\varphi(x)$ and $\psi(x)$, the following conditions are equivalent*:

- *there does not exist an L'-separator for $\varphi(x)$ and $\psi(x)$*;
- *there exist structures $\mathfrak{A}_i$ and $a_i \in \mathsf{dom}(\mathfrak{A}_i)$, $i = 1, 2$, such that $\mathfrak{A}_1 \models \varphi(a_1)$, $\mathfrak{A}_2 \models \neg\psi(a_2)$ and $\mathfrak{A}_1, a_1 \equiv_{L'(\sigma)} \mathfrak{A}_2, a_2$, where $\sigma = sig(\varphi) \cup sig(\psi)$.*

Here, $\mathfrak{A}_1, a_1 \equiv_{L'(\sigma)} \mathfrak{A}_2, a_2$ means that $\mathfrak{A}_1 \models \gamma(a_1)$ iff $\mathfrak{A}_2 \models \gamma(a_2)$, for all $L'(\sigma)$-formulas γ. This characterisation translates to the following criterion of L/L'-definability:

▶ **Theorem 18.** *Let $L' \subseteq L \subseteq$ FO. Then an L-formula φ is logically equivalent to some L'-formula iff φ is preserved under $\equiv_{L'(\sigma)}$, where $\sigma = sig(\varphi)$.*

As before, one can make these criteria operational by replacing $\equiv_{L'(\sigma)}$ with a model-theoretic characterisation via suitable bisimulations. Below, we focus on definability and separation in ML, and give pointers to the literature for more expressive languages. Using the fact that modal equivalence is characterised by bisimulations on ω-saturated structures, we obtain van Benthem's theorem:

▶ **Theorem 19** ([16])**.** *For any* FO*-formula $\varphi(x)$ the following conditions are equivalent*:

- *$\varphi(x)$ is equivalent to (the standard translation of) an* ML*-formula*;
- *$\varphi(x)$ is invariant under σ-bisimulations, for $\sigma = sig(\varphi)$, which means that, for any pointed models $\mathfrak{M}_i, u_i$, $i = 1, 2$, whenever $\mathfrak{M}_1, u_1 \sim_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, u_2$, then $\mathfrak{M}_1 \models \varphi(u_1)$ iff $\mathfrak{M}_2 \models \varphi(u_2)$.*

We aim to apply Theorem 19 to the problem of deciding F/ML-definability for FO-fragments F. If F = FO, this problem is undecidable because FO is undecidable. For decidable fragments

of FO, one can try to use the approach of the previous section and construct finite or tree-shaped well-behaved bisimilar models refuting the second item above. Here, we first follow a different path and use a more informative version of van Benthem's theorem whose proof does not use compactness but a more combinatorial approach (that also works on finite models, see below). It employs the notion of n-bisimilar pointed models [47], which characterises modal indistinguishability up to formulas of modal depth at most n, where the *modal depth* of an ML-formula is the maximum number of nested modal operators in it. We write $\mathfrak{M}_1, w_1 \sim^n_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$ if $\mathfrak{M}_1, w_1$ and $\mathfrak{M}_2, w_2$ are σ-n-bisimilar, and $\mathfrak{M}_1, w_1 \equiv^n_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$ if $\mathfrak{M}_1, w_1$ and $\mathfrak{M}_2, w_2$ satisfy the same $\mathsf{ML}(\sigma)$-formulas of modal depth $\leq n$.

▶ **Lemma 20.** *For any $n \geq 0$, we have $\mathfrak{M}_1, w_1 \sim^n_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$ iff $\mathfrak{M}_1, w_1 \equiv^n_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$.*

Now a non-trivial proof uses n-bisimilarity to bound the depth of an equivalent modal formula in terms of the depth of the FO-formula [82]. We remind the reader that the *quantifier rank* of an FO-formula is the maximum nesting depth of quantifiers in it.

▶ **Theorem 21** ([82])**.** *Any FO-formula $\varphi(x)$ of quantifier rank q that is invariant under bisimulations is equivalent to an ML-formula of modal depth at most $2^q - 1$.*

As one can compute a finite list of all ML-formulas (up to logical equivalence) in a finite signature and of bounded modal depth, we obtain:

▶ **Theorem 22.** *F/ML-definability is decidable, for any decidable FO-fragment F containing ML.*

Note that this global approach, which applies to all decidable fragments of FO containing ML, gives no elementary upper complexity bound since the number of modal formulas of a fixed modal depth is non-elementary. Moreover, for full FO, there is no elementary bound on the size of an ML-formula that is equivalent to a given FO-formula [49]. We now discuss another technique that can sometimes be used to investigate L/L'-definability and even construct small defining formulas.

Good Model Method

Let $L \supseteq L'$ be a pair of first-order or modal languages. A class $\mathcal{G}$ of pointed models is called *good for L/L'-definability* if, for any L-formula φ, there is a *simplifying L'-formula* φ^s such that

(L'-transfer) for every pointed model $\mathfrak{M}, w$ using symbols in a signature σ, there exists a pointed model $\mathfrak{M}^s, w^s$ in $\mathcal{G}$ such that $\mathfrak{M}, w \equiv_{L'(\sigma)} \mathfrak{M}^s, w^s$;

(L/L'-indistinguishability) for every pointed model $\mathfrak{M}, w$ in $\mathcal{G}$, we have $\mathfrak{M}, w \models \varphi$ iff $\mathfrak{M}, w \models \varphi^s$.

The following easily proved lemma shows that the good model method provides explicit L'-definitions whenever they exist and reduces L/L'-definability to L-validity.

▶ **Lemma 23.** *If $\mathcal{G}$ is good for L/L'-definability, then an L-formula φ is L'-definable iff $\models \varphi \leftrightarrow \varphi^s$.*

We first illustrate the method for GML/ML-definability, which asks whether counting modalities $\Diamond^{\geq k}$ in a given GML-formula are really needed or can be replaced by standard modalities. The answer is surprisingly simple. For any Kripke model $\mathfrak{M} = (W, R, \mathfrak{v})$, let $\mathfrak{M}^\omega = (W^\omega, R^\omega, \mathfrak{v}^\omega)$ be the model obtained from $\mathfrak{M}$ by replacing every world w in it by ω-many copies (w, i), $i < \omega$, that is, by taking

$$W^\omega = \bigcup_{w \in W} \bar{w}, \quad R^\omega = \bigcup_{(w,v) \in R} \bar{w} \times \bar{v}, \quad \mathfrak{v}^\omega(p) = \bigcup_{w \in \mathfrak{v}^\omega(p)} \bar{w}, \quad \text{where } \bar{w} = \{(w, i) \mid 0 \leq i < \omega\}.$$

It is not difficult to show that the class of all pointed Kripke models of the form $\mathfrak{M}^\omega, (w, 0)$ is good for GML/ML-definability. For (ML-transfer), observe that $\mathfrak{M}, w$ and $\mathfrak{M}^\omega, (w, 0)$ satisfy the same ML-formulas. For (GML/ML-indistinguishability), we obtain from any GML-formula φ a simplifying formula φ^f by replacing any $\Diamond^{\geq k}$ in φ with $\Diamond$. We call φ^f the *flattening* of φ. It not difficult to show that $\mathfrak{M}^\omega, (w, 0) \models \varphi$ iff $\mathfrak{M}^\omega, (w, 0) \models \varphi^f$ since any world has infinitely many copies. Now, to decide ML-definability of $\varphi \in \mathsf{ML}$, it is enough to check whether the GML-equivalence $\varphi \leftrightarrow \varphi^f$ is true in all models, which can be done in PSPACE [92]. Thus, we obtain:

▶ **Theorem 24** ([66])**.** GML/ML-*definability is* PSPACE-*complete.*

The good model method works well for various pairs L/L', in which L is a logic with counting and L' without counting. For instance, it can be used to show that C^2/ML-definability is coNEXPTIME-complete [66], where C^2 is the extension of FO^2 with counting quantifiers $\exists^{\geq k} x$ [92].

The good model methods can also be used to show Theorem 21. The proof is in two steps by first analysing FO/ML$^{\mathsf{u}}$-definability and then ML$^{\mathsf{u}}$/ML-definability, where ML$^{\mathsf{u}}$ is the extension of ML with the universal modality, denoted $\Diamond_u$ and interpreted by setting $\mathfrak{M}, w \models \Diamond_u \varphi$ if there exists $v \in W$ with $\mathfrak{M}, v \models \psi$. To obtain good models for FO/ML$^{\mathsf{u}}$-definability, one additionally unfolds $\mathfrak{M}^\omega$ into a tree-shaped model known as the *ω-expansion* $\mathfrak{M}^{\uparrow\omega}$ of $\mathfrak{M}$ [5, 55]. Given $\varphi(x)$ in FO, the simplifying formula φ^s is the conjunction of all ML$^{\mathsf{u}}$-consequences of $\varphi(x)$ of modal depth bounded by 2^q with q the quantifier rank of φ, see [52]. Good models for ML$^{\mathsf{u}}$/ML-definability are obtained by taking for any pointed $\mathfrak{M}, w$ its disjoint union with some model $\mathfrak{M}$ satisfying every satisfiable ML-formula (for instance the canonical model, see above). A simplifying formula is constructed from φ in ML$^{\mathsf{u}}$ by replacing any occurrence of $\Diamond_u \psi$ by $\top$ if ψ is satisfiable and by $\bot$ otherwise, starting with innermost occurrences.

While definability has been studied in depth for many pairs L/L' of logics, the work on L/L'-separation has only just begun. Computationally, separation is often much harder than definability. For example, using the fact that ML-indistinguishability in Theorem 17 can be replaced by ML-bisimilarity, one can show that C^2/ML-separation is undecidable by a

reduction of Minsky machines. It follows that Theorem 22 does not generalise to separation. For fragments of C^2, however, a few decidability results are known. For instance, by proving the FBMP for GML-formulas and ML-bisimulations using an approach similar to the proof of Proposition 9, one can show that GML/ML-separation is coNExpTime-complete—more complex than GML/ML-definability but still decidable.

▶ **Theorem 25** ([66]). **(i)** C^2/ML-*separation is undecidable.*
(ii) GML/ML-*separation is* coNExpTime-*complete.*

Further Reading. Theorems 21 and 22 have been generalised (sometimes with different bounds on the modal depth of the equivalent formula) to extensions such as graded modal logic [83], modal logics with inverse or the universal modality [82], and modal logic with nominals [51]. First results on the construction and size of separators have recently been obtained in [58]. For instance, for GML/ML-separation in Theorem 25, there always exist separators of elementary size if separators exist at all. It remains an open problem to determine tight bounds. In description logic, definability and separation have been studied at the level of TBoxes, which corresponds to the global consequence relation in modal logic: 'if φ is true in all worlds of a model, then ψ is true in all worlds of that model'. Using model-theoretic methods such as bisimulations, some decidability and complexity results are obtained in [72, 8]. A result to highlight is the undecidability of $\mathcal{ALCO}^u/\mathcal{ELO}$-separation (formulated in [8] in terms of definability), which shows that separation can be undecidable for rather weak languages.

4.2 Definability and Separation for Logics with Second-Order Quantifiers and Fixed-Point Operators

Next, we consider two prominent logics, MSO and μML, that add expressive power to FO and ML in the form of quantifiers over sets and fixed-point operators, respectively, and are used in formal verification and many other areas of computer science. Logics with second-order quantifiers or fixed points are typically not compact because of which some of the model-theoretic methods discussed in the previous section do not work, at least when applied to standard relational structures. In this case, automata-theoretic approaches, possibly based on finitary approximations of bisimulations, are often more appropriate.

In this section, we mainly focus on two questions:

- Can the analysis of definability and separation for FO/ML be lifted to MSO/μML?
- How hard is it to decide definability and separation for formulas with fixed-point operators by formulas without those operators?

In many cases, the latter question is equivalent to asking whether the use of recursion in a given formula is spurious and can be eliminated. We begin by reminding the reader of the basic μ-calculus.

Given a set W, a function $f\colon 2^W \to 2^W$ is called *monotone* if $X \subseteq Y$ implies $f(X) \subseteq f(Y)$. The *least fixed point* of f is the smallest (w.r.t. $\subseteq$) set $X \subseteq W$ such that $f(X) = X$. If f

is monotone, the least fixed point of f always exists and can be computed by transfinite induction as follows:

$$\begin{array}{lcl} f^0(X) & = & X, \\ f^{\alpha+1}(X) & = & f(f^\alpha(X)), \text{ for any ordinal } \alpha, \\ f^\alpha(X) & = & \bigcup\limits_{\beta<\alpha} f^\beta(X), \text{ for a limit ordinal } \alpha. \end{array}$$

The least fixed point of f is then given by $f^\alpha(\emptyset)$, for the least ordinal α with $f^\alpha(\emptyset) = f^{\alpha+1}(\emptyset)$; this α is called the *closure ordinal* of f.

The *modal μ-calculus* $\mu\mathsf{ML}$ is the extension of basic modal logic ML with least fixed-point operators. More precisely, $\mu\mathsf{ML}$*-formulas* are defined by the grammar

$$\varphi \;:=\; \top \mid p \mid \varphi \wedge \varphi' \mid \neg\varphi \mid \Diamond\varphi \mid P \mid \mu P.\varphi,$$

where p ranges over propositional variables and P over unary predicate symbols (second-order variables). For formulas of the form $\mu P.\varphi$, it is required that every free occurrence of P in φ is *positive* (i.e., in the scope of an even number of $\neg$). A $\mu\mathsf{ML}$*-sentence* is any $\mu\mathsf{ML}$-formula φ whose every second-order variable P is in the scope of some μP. Given a $\mu\mathsf{ML}$-formula φ, a Kripke model $\mathfrak{M} = (W, R, \mathfrak{v})$, $u \in W$, and a function $\mathfrak{V}$ mapping second-order variables P to 2^W, the *truth-relation* $\mathfrak{M}, u \models_{\mathfrak{V}} \varphi$ is defined inductively by extending the respective definition for ML with two items:

- $\mathfrak{M}, u \models_{\mathfrak{V}} P$ iff $u \in \mathfrak{V}(P)$;
- $\mathfrak{M}, u \models_{\mathfrak{V}} \mu P.\varphi$ iff u is in the least fixed point of the function

$$f_\varphi \colon X \mapsto \{v \in W \mid \mathfrak{M}, v \models_{\mathfrak{V}[P\mapsto X]} \varphi\},$$

where $\mathfrak{V}[P \mapsto X]$ maps P to X and coincides with $\mathfrak{V}$ on all other second-order variables. (Note that f_φ is monotone because P occurs only positively in φ.) For further details and discussions, we refer the reader to [24]. Here we only show some standard examples of what can be said in $\mu\mathsf{ML}$.

▶ **Example 26.** Using the definitions of truth and least fixed point, one can readily see that

- $\mathfrak{M}, u \models \mu P.(\Diamond q \vee \Diamond P)$ iff there is an R-path $uRu_1R \dots Ru_n$ with $n \geq 1$ in $\mathfrak{M}$ such that $\mathfrak{M}, u_n \models q$;
- $\mathfrak{M}, u \models \mu P.\big(\Diamond q \vee (r \wedge \Diamond P)\big)$ iff there is an R-path as above, in which $\mathfrak{M}, u_i \models r$ for all i, $1 \leq i < n$;
- $\mathfrak{M}, u \models \mu P.\Box P$ iff $\mathfrak{M}$ does not contain an infinite path of the form $uRu_1R \dots Ru_nR \dots$.

In fact, $\mu\mathsf{ML}$ is powerful enough to express the operators of temporal logics LTL (see Section 6 below), CTL, CTL^* and propositional dynamic logic PDL (in which case a multimodal $\mu\mathsf{ML}$ is required). ⌟

In many ways, μML has the same relationship to MSO as ML has to FO. First, we observe that the standard translation $\varphi \mapsto \varphi^*_x$ of ML to FO defined in Section 2 can naturally be extended to a translation of μML to MSO by taking

$$P^*_x = P(x), \quad (\mu P.\varphi)^*_x = \forall P \left[\forall \bar{x} \left(\varphi^*_{\bar{x}} \leftrightarrow P(\bar{x})\right) \rightarrow P(x)\right], \text{ where } \bar{x} = y,\ \bar{y} = x.$$

Second, μML-sentences are still invariant under ML-bisimulations and, in fact, we have the following extension of Theorems 19 and 22:

▶ **Theorem 27** (Janin-Walukiewicz Theorem [55])**.**
(i) *An* MSO*-formula* $\varphi(x)$ *is* μML*-definable iff it is invariant under* ML*-bisimulations.*
(ii) M/μML*-definability is decidable, for any decidable* MSO*-fragment* M *containing* μML*.*

The proof is by the good model method. The good models are the tree-shaped ω-expansions $\mathfrak{M}^{\uparrow\omega}$ introduced above. However, it is harder now to construct appropriate simplifying formulas. They are obtained by translating the input MSO-formula $\varphi(x)$ to an automaton $\mathcal{A}$ accepting the tree-shaped models of φ. The automaton $\mathcal{A}$ is then transformed to an automaton $\mathcal{A}^s$ accepting exactly the same models as a μML-sentence φ^s and exactly the same models of the form $\mathfrak{M}^{\uparrow\omega}$ as $\mathcal{A}$. Similarly to the flattening φ^f defined above for GML/ML, the transformation from $\mathcal{A}$ to $\mathcal{A}^s$ forgets the quantitative information in $\mathcal{A}$. The construction is effective, and so, by Lemma 23, one can decide μML-definability of φ for decidable fragments of MSO by checking the equivalence $\models \varphi \leftrightarrow \varphi^s$.

Further Reading. Different methods are needed to show Theorems 19 and 27 on finite models. For Theorem 19, this was first achieved by Rosen [93]. Otto's proof of Theorem 21 can also be used for this purpose. Theorem 27 has only very recently been shown on finite models [31], see also [20, 86].

We next consider the definability and separation of formulas with least fixed-point operators by formulas without such operators. A basic variant of this problem is the *boundedness problem* that has been investigated for a few decades in many different contexts. Let $\varphi(P, x)$ be an FO-formula with a unary predicate P that occurs only positively in φ. Given a structure $\mathfrak{A}$, define a monotone function $f_\varphi \colon 2^{\mathsf{dom}(\mathfrak{A})} \to 2^{\mathsf{dom}(\mathfrak{A})}$ by taking $f_\varphi(X) = \{a \in \mathsf{dom}(\mathfrak{A}) \mid \mathfrak{A} \models \varphi(X, a)\}$. Similarly to μML we can introduce the fixed point operator $\mu P.\varphi(P, x)$ with the following truth-condition: $\mathfrak{A} \models \mu P.\varphi(a)$ iff a belongs to the least fixed point of f_φ on $\mathfrak{A}$.

▶ **Example 28.** For $\varphi(P, x) = A(x) \vee \exists y \left(R(x, y) \wedge P(y)\right)$, we have $\mathfrak{A} \models \mu P.\varphi(a)$ iff $\mathfrak{A}$ contains an R-path from a to some $b \in \mathsf{dom}(\mathfrak{A})$ with $\mathfrak{A} \models A(b)$. Thus, the formula $\mu P.\varphi(x)$ is equivalent to the monadic datalog query $(\Pi_\varphi, P(x))$, in which the program Π_φ consists of two rules: $P(x) \leftarrow A(x)$ and $P(x) \leftarrow R(x, y) \wedge P(y)$. ⌟

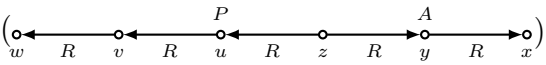
A formula $\varphi(P, x)$ is called *bounded* if there is some $n < \omega$ such that, for every structure $\mathfrak{A}$, the closure ordinal of f_φ applied to $\mathfrak{A}$ does not exceed n. It is readily seen that if $\varphi(P, x)$ is

bounded, then $\mu P.\varphi(P,x)$ is FO-definable. The classical Barwise-Moschovakis theorem below states that the converse also holds for FO:

▶ **Theorem 29** ([10]). *For every FO-formula $\varphi(P,x)$ with only positive occurrences of P, the formula $\mu P.\varphi$ is FO-definable iff $\varphi(P,x)$ is bounded over the class of all structures.*

This notion of boundedness formalises a fundamental question in computer science: when is recursion really needed? For example, is it possible to transform (optimise) a given database query with recursion (say, a datalog query) to an equivalent SQL-query, i.e., essentially an FO-formula?

▶ **Example 30.** One can check that, for $\psi(P,x)$ depicted below, the closure ordinal of f_ψ is 2:

$\psi(P,x) = A(x) \vee \exists w,v,u,z,y\,($  $)$.

The formula $\mu P.\psi(P,x)$ corresponds to the datalog query $(\Pi_\psi, P(x))$, where Π_ψ contains the rules

$$P(x) \leftarrow A(x), \quad P(x) \leftarrow R(v,w) \wedge R(u,v) \wedge P(u) \wedge R(z,u) \wedge R(z,y) \wedge A(y) \wedge R(y,x),$$

and so this datalog query can be rewritten to an equivalent FO-query. ⌟

The problem of deciding whether $\mu P.\varphi$ is FO-definable, for an input formula $\varphi(P,x)$ in a given language L, has been investigated extensively, mostly in the equivalent boundedness formulation. For example, this problem is undecidable for FO^2 [64] but decidable for GF [13]. For a wide range of results on the computational complexity of deciding boundedness of datalog queries and ontology-mediated queries in various description logics the reader can consult, e.g., [33, 18, 11, 73] and further references therein. For instance, deciding boundedness of $\mu P.\varphi$ such as the one in Example 30 is 2ExpTime-complete [13, 63].

We next consider a more general case, where fixed-point operators are part of the language L (and so can be nested), focusing on $\mu\mathsf{ML}/\mathsf{ML}$-definability and separation.

▶ **Theorem 31** ([81, 56]). *$\mu\mathsf{ML}/\mathsf{ML}$-definability and separation are both ExpTime-complete.*

To discuss the proof, given any signature σ, let $\mathfrak{M}_1, w_1 =^n_\sigma \mathfrak{M}_2, w_2$ denote that $\mathfrak{M}_1$ and $\mathfrak{M}_2$ are tree-shaped with roots w_1 and w_2, respectively, and the σ-reducts of the submodels induced by the set of nodes in $\mathfrak{M}_1$ and $\mathfrak{M}_2$ reachable from w_1 and w_2 in n steps are isomorphic. The next lemma gives the key steps of the proof:

▶ **Lemma 32.** *For any $\mu\mathsf{ML}$-sentences φ_1 and φ_2, $n \geq 0$, and finite signature σ, the following conditions are equivalent*:

(**a**) *there is no $\mathsf{ML}(\sigma)$-formula of modal depth $\leq n$ separating φ_1 and φ_2;*
(**b**) *there are tree-shaped $\mathfrak{M}_1, w_1 \models \varphi_1$ and $\mathfrak{M}_2, w_2 \models \neg\varphi_2$ such that $\mathfrak{M}_1, w_1 \equiv^n_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$;*
(**c**) *there are tree-shaped $\mathfrak{M}_1, w_1 \models \varphi_1$ and $\mathfrak{M}_2, w_2 \models \neg\varphi_2$ such that $\mathfrak{M}_1, w_1 \sim^n_{\mathsf{ML}(\sigma)} \mathfrak{M}_2, w_2$;*
(**d**) *there are tree-shaped $\mathfrak{M}_1, w_1 \models \varphi_1$ and $\mathfrak{M}_2, w_2 \models \neg\varphi_2$ such that $\mathfrak{M}_1, w_1 =^n_\sigma \mathfrak{M}_2, w_2$;*

(**e**) *there are tree-shaped* $\mathfrak{M}_1, w_1 \models \varphi_1$ *and* $\mathfrak{M}_2, w_2 \models \neg\varphi_2$ *of outdegree* $\leq |\varphi_1| + |\varphi_2|$ *such that* $\mathfrak{M}_1, w_1 =^n_\sigma \mathfrak{M}_2, w_2$.

Observe that, for $\sigma = sig(\varphi) \cup sig(\varphi)$, (**a**) holds for all $n > 0$ iff there is no ML-separator for φ_1, φ_2. The main step is (**c**) $\Rightarrow$ (**d**), which is an amalgamation property that typically fails when dealing with logics that lack the CIP or when investigating L/L'-separation. The ExpTime upper bound is derived from (**e**) using automata-theoretic techniques by showing that (**e**) holds for all $n > 0$ iff (**e**) holds for some n exponential in $|\varphi_1| + |\varphi_2|$. Interestingly, (**c**) $\Rightarrow$ (**d**) does not hold if the outdegree of models is bounded and ≥ 3, which reflects that Alt_n does not have the CIP for $n \geq 3$. Also, $\mu\mathsf{ML}/\mathsf{ML}$-separation on n-ary trees, $n \geq 3$, is 2ExpTime-complete, a result that is harder to show than the ExpTime-bound above [57].

Further Reading. Many definability results discussed in this chapter have been (at least partially) extended to guarded fragments of FO with and without fixed points. Theorem 19 has been lifted to a characterisation of GF within FO using *guarded bisimulations* [4]. It appears to be open whether an analogue of Theorem 22 holds for GF, but rather general positive results shown by reduction to languages over trees are known [12]. For instance, it is 2ExpTime-complete to decide whether a formula in the guarded negation fragment of FO is definable in GF. The characterisation part of Theorem 27 has been lifted, using guarded bisimulations, to a characterisation of $\mu\mathsf{GF}$ within a fragment GSO of SO whose second-order quantification is restricted to guarded sets [48]. Note that the characterisation does not hold if GSO is replaced by full SO. Again, it appears to be open whether the decidability part of Theorem 27 still holds, but general positive results are shown in [12].

5 Definability and Separation in Formal Languages

In this section, we consider the following decision problems for classes $\mathcal{C}' \subsetneq \mathcal{C}$ of formal languages:

***$\mathcal{C}/\mathcal{C}'$-definability*:** given a language $\boldsymbol{L} \in \mathcal{C}$, decide whether $\boldsymbol{L} \in \mathcal{C}'$.

***$\mathcal{C}/\mathcal{C}'$-separation*:** given languages $\boldsymbol{L}_1, \boldsymbol{L}_2 \in \mathcal{C}$, decide whether there is a language $\boldsymbol{L} \in \mathcal{C}'$—called a $\mathcal{C}'$-*separator* for $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$—such that $\boldsymbol{L}_1 \subseteq \boldsymbol{L}$ and $\boldsymbol{L}_2 \cap \boldsymbol{L} = \emptyset$.

A significant amount of related work has been focusing on defining and separating regular languages. The study of regular languages brings together combinatorial methods on words with automata-theoretic, algebraic and logical techniques. These were originally developed for languages consisting of finite words, but have been generalised to languages over infinite words, trees, and graphs.

Here, we discuss in detail a particular instance of these problems: the decidability of defining and separating regular languages consisting of finite words by star-free (or, equivalently, FO-definable) languages. We also discuss possible bounds on the size of a definition/separator

when it exists. Some topics in the wider context (such as languages consisting of infinite words and various kinds of labelled trees; using some FO-fragments as definitions/separators; and defining/separating non-regular languages with regular ones) are only briefly sketched.

We begin by recalling the basic notions from the theory of formal languages; see, e.g., [54, 95]. An *alphabet* is any finite set A. Given an alphabet A, we denote by A^+ the set of all nonempty finite words over A. A subset $\boldsymbol{L} \subseteq A^+$ is called a *language over* A. (To simplify presentation, we only consider languages without the empty word, but similar results hold if the empty word is also allowed.) The class of *regular languages* is the smallest class that contains A^+, all singletons $\{a\}$, for $a \in A$, and is closed under concatenation, finite union, and the +-version of the Kleene star (iteration). The class of *star-free languages* is the smallest class that contains A^+, all singletons $\{a\}$, for $a \in A$, and is closed under concatenation, finite union, and complementation. (As the class of regular languages is closed under complementation, star-free languages are regular.)

Automata-Theoretic Approach to Regular Languages. It is well known that a language is regular iff it can be specified by a deterministic finite automaton (DFA) or, equivalently, by a nondeterministic finite automaton (NFA). Given an NFA $\mathfrak{A}$, every word $w \in A^+$ determines a binary *transition relation* Δ_w on the states of $\mathfrak{A}$: it consists of all those state-pairs (q, q'), for which $\mathfrak{A}$ can reach q' having started at q and read w. If $\mathfrak{A}$ is a DFA, then each Δ_w is actually a state-to-state *transition function* δ_w. We say that such a δ_w *has a non-trivial cycle* if there exist a state q and n, $1 < n < \omega$, such that $\delta_w(q) \neq q$ and $\delta_{w^n}(q) = q$. A DFA is called *counter-free* if no δ_w has a non-trivial cycle, for any $w \in A^+$.

Algebraic Approach to Regular Languages. A *semigroup* is a structure $\boldsymbol{S} = (S, \cdot)$, where $\cdot$ is an associative binary operation on S. For example, the following are semigroups: A^+ with concatenation (also denoted by A^+), and the set of transition relations in an NFA (transition functions in a DFA) $\mathfrak{A}$ with composition, called the *transition semigroup of* $\mathfrak{A}$. Given $s \in S$ and n with $0 < n < \omega$, we write s^n for $\underbrace{s \cdot \ldots \cdot s}_{n}$. We call s *idempotent* if $s^2 = s$. It is easy to see that, for every finite semigroup $\boldsymbol{S}$, there is $i_{\boldsymbol{S}}$, $0 < i_{\boldsymbol{S}} \leq |S|!$, such that $s^{i_{\boldsymbol{S}}}$ is idempotent, for all s in $\boldsymbol{S}$. We call $\boldsymbol{S}$ *aperiodic* if $s^{i_{\boldsymbol{S}}} = s^{i_{\boldsymbol{S}}+1}$, for all $s \in S$.

We say that a language $\boldsymbol{L}$ is *recognised* by a semigroup $\boldsymbol{S}$ if there exist a semigroup homomorphism $\alpha\colon A^+ \to \boldsymbol{S}$ and some $F \subseteq S$ such that $\boldsymbol{L} = \alpha^{-1}(F)$. It is easy to see that if $\boldsymbol{L}$ is recognised by a finite semigroup, then $\boldsymbol{L}$ is regular. On the other hand, any language $\boldsymbol{L}$ that is specified by an NFA $\mathfrak{A}$ is always recognised by the transition semigroup of $\mathfrak{A}$. As the latter is finite (of size at most 2^{n^2}, for the number n of states in $\mathfrak{A}$), we obtain:

▶ **Theorem 33.** *A language is regular iff it is recognised by a finite semigroup.*

In fact, for any regular language $\boldsymbol{L}$, one can obtain a 'canonical' finite semigroup recognising $\boldsymbol{L}$, called the *syntactic semigroup of* $\boldsymbol{L}$: it is the transition semigroup of the minimal DFA specifying $\boldsymbol{L}$.

Logical Approach to Regular Languages. Any word $w \in A^+$ can be regarded as an FO-structure in the signature σ_A consisting of a binary predicate symbol $<$ (interpreted in w as the strict linear order over its positions $\{0, \dots, |w|-1\}$) and unary predicate symbols $\boldsymbol{a}$, for $a \in A$ (selecting the subset of $\{0, \dots, |w|-1\}$ where the symbol in w is a). A language $\boldsymbol{L} \subseteq A^+$ is said to be $\mathsf{MSO}(<)$-*definable* if there is some MSO-sentence φ (in which all first- and second-order variables are bound) in the signature σ_A such that $\boldsymbol{L} = \{w \in \Sigma^+ \mid w \models \varphi\}$. If such a φ can be chosen to be an MSO-sentence of the form $\exists P_1 \dots \exists P_n\, \psi$ with all P_i being unary predicate symbols and ψ not having any second-order quantification, then $\boldsymbol{L}$ is called $\exists\mathsf{MSO}(<)$-*definable*; and if φ can be an FO-sentence, then $\boldsymbol{L}$ is called $\mathsf{FO}(<)$-*definable*. The following statement is known as the Büchi–Elgot–Trakhtenbrot theorem (see also [100, Theorem III.1.1] and [71, Theorem 7.21]):

▶ **Theorem 34** ([25, 38, 105]). *A language is regular iff it is* $\mathsf{MSO}(<)$*-definable iff it is* $\exists\mathsf{MSO}(<)$*-definable.*

Languages over Infinite Words. Given an alphabet A, we denote by A^ω the set of all infinite words, called ω-*words*, over A. A subset $\boldsymbol{L} \subseteq A^\omega$ is called an ω-*language over* A. The notions of *regularity* and *star-freeness* of ω-languages can be defined by appropriately adjusting the closure conditions given above for finite-word languages. The automata-theoretic, algebraic and logical approaches can be adapted as well. Regular ω-languages are those that are specified by *nondeterministic Büchi automata* (*NBAs*) [25]. Theorem 33 also holds in this setting: regular ω-languages coincide with those that are recognised by finite ω-*semigroups* (an appropriate generalisation of finite semigroups) [85]. Also, any ω-word $w \in A^\omega$ can be regarded as an FO-structure in the signature σ_A whose underlying strict linear order is isomorphic to $(\omega, <)$, and Theorem 34 also holds: An ω-language over A is regular iff it is definable over $(\omega, <)$ by an MSO-sentence in the signature σ_A [25].

5.1 Definability

The problem of deciding whether a regular language is star-free can be addressed via giving automata-theoretic, algebraic, or logical characterisations of star-free languages:

▶ **Theorem 35** ([79, 94, 98]). *Suppose* $\boldsymbol{L}$ *is a regular language. Then the following are equivalent*:

(i) $\boldsymbol{L}$ *is star-free*;

(ii) $\boldsymbol{L}$ *is specified by a counter-free DFA* (*or, equivalently, the minimal DFA specifying* $\boldsymbol{L}$ *is counter-free*);

(iii) $\boldsymbol{L}$ *is recognised by a finite aperiodic semigroup* (*or, equivalently, the syntactic semigroup of* $\boldsymbol{L}$ *is aperiodic*);

(iv) $\boldsymbol{L}$ *is* $\mathsf{FO}(<)$*-definable.*

Observe that, by Theorems 34 and 35.iv, the question whether a regular language is star-free corresponds to asking whether an MSO-sentence in the signature σ_A is equivalent to

some FO-sentence over finite strict linear orders. There are many other properties equivalent to the ones listed in Theorem 35; see [37] for a survey. (As we shall see in the next section, definability in linear temporal logic LTL over finite timelines is one of them.) Note also that there are several other equivalent algebraic definitions of aperiodicity, leading to different proofs of the equivalence of the other properties and Theorem 35.iii. See Examples 39 and 40 below for some languages that are FO(<)-definable/undefinable.

Using the automata-theoretic and algebraic characterisations of FO(<)-definability, one can show that it is decidable and establish a tight complexity bound:

▶ **Corollary 36** ([98, 30, 17]). *Deciding* FO(<)*-definability of the language specified by a DFA* $\mathfrak{A}$ *is* PSPACE*-complete in the size of* $\mathfrak{A}$.

Constructing a Possible Definition. Given a counter-free DFA $\mathfrak{A}$ specifying $\boldsymbol{L}_{\mathfrak{A}} \subseteq A^+$, Wilke [107] constructs an FO-definition φ of $\boldsymbol{L}_{\mathfrak{A}}$ such that the size of φ is linear in $|A|$ and double-exponential in the size of $\mathfrak{A}$.

Languages over Infinite Words. FO(<)-definable ω-languages also have equivalent characterisations that are similar to those in Theorem 35. The equivalence of FO(<)-definability and star-freeness is shown in [70, 102]. Given any NBA specifying an ω-language $\boldsymbol{L}$, one can obtain a 'canonical' structure (the *syntactic ω-semigroup* of $\boldsymbol{L}$) [106], whose *aperiodicity* (that is, a property similar to aperiodicity of semigroups) is equivalent to $\boldsymbol{L}$ being star-free [84]. The equivalence of FO(<)-definability and being specified by deterministic counter-free Büchi automata with Rabin–Streett acceptance conditions is shown in [103], based on earlier work in [78]. The survey [37] also covers several other equivalent characterisations.

Languages over Trees. The question of deciding whether a given language consisting of various versions of labelled trees is definable in first-order logic is a long-standing open problem in language theory [104]. There have been a lot of (unsuccessful) attempts to fully generalise Theorem 35 for tree-languages; see, e.g., the surveys [21, 22].

5.2 Separation

As the class $\mathcal{C}$ of regular languages is closed under complementation, $\mathcal{C}/\mathcal{C}'$-definability can be reduced to $\mathcal{C}/\mathcal{C}'$-separation, for any $\mathcal{C}' \subseteq \mathcal{C}$: definability of $\boldsymbol{L}$ can be decided by testing separability of $\boldsymbol{L}$ from its complement. Also, by the above, the problem of deciding whether two regular languages are separable by a star-free language has equivalent automata-theoretic, algebraic and logical formulations:

1. Can two languages given by a DFA be separated by a language specified by a counter-free DFA?
2. Can two languages, each recognised by a finite semigroup, be separated by a language recognised by a finite aperiodic semigroup?

3. Can two MSO(<)-definable languages be separated by an FO(<)-definable language? (In other words, can two MSO-sentences in the signature σ_A be separated by some FO-sentence over finite strict linear orders?)

As we shall see in the next section, this question also corresponds to the IEP in linear temporal logic LTL over finite timelines.

The following result was first obtained with the help of purely algebraic methods:

▶ **Theorem 37** ([50, 1]). *It is decidable whether two regular languages are separable by an* FO(<)*-definable language.*

A more general result of [89] (obtained using a mix of algebraic and combinatorial techniques) shows the decidability of the separation (and the more general *covering*) problem for regular languages by any class $\mathcal{C}'$ of regular languages satisfying certain closure conditions. Here, we present the algorithm from [89] for the case when $\mathcal{C}'$ is the class of all FO(<)-definable languages.

Observe first that if $\boldsymbol{S}_1, \boldsymbol{S}_2$ are semigroups recognising regular languages $\boldsymbol{L}_1, \boldsymbol{L}_2$ with respective homomorphisms α_1, α_2, then the direct product $\boldsymbol{S}_1 \times \boldsymbol{S}_2$ recognises both $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ with the homomorphism $\alpha\colon A^+ \to \boldsymbol{S}_1 \times \boldsymbol{S}_2$ defined by taking $\alpha(w) = \big(\alpha_1(w), \alpha_2(w)\big)$. So we may assume that we are given two regular languages $\boldsymbol{L}_1, \boldsymbol{L}_2$ recognised by the same semigroup $\boldsymbol{S}$, homomorphism $\alpha\colon A^+ \to \boldsymbol{S}$, and respective sets $F_1, F_2 \subseteq S$. Given two subsets T, T' of S, we set $T \cdot T' = \{t \cdot t' \mid t \in T,\ t' \in T'\}$. For $T \in 2^S$ and $0 < n < \omega$, we write T^n for $\underbrace{T \cdot \ldots \cdot T}_{n}$. We let $\boldsymbol{S}^\dagger$ be the smallest subset of 2^S that contains the singletons $\{\alpha(w)\}$, $w \in A^+$, and is closed under the following operations:

- **C1** if $T \in \boldsymbol{S}^\dagger$, then $T' \in \boldsymbol{S}^\dagger$ for all $T' \subseteq T$;
- **C2** if $T, T' \in \boldsymbol{S}^\dagger$, then $T \cdot T' \in \boldsymbol{S}^\dagger$;
- **C3** if $T \in \boldsymbol{S}^\dagger$, then $T^{is} \cup T^{is+1} \in \boldsymbol{S}^\dagger$.

It is shown in [89] that

$\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ are separable by an FO(<)-definable language iff

$$\{t_1, t_2\} \notin \boldsymbol{S}^\dagger \text{ for all } t_1 \in F_1,\ t_2 \in F_2. \quad (1)$$

Thus, we obtain the following:

▶ **Theorem 38** ([89]). *Deciding* FO(<)*-separability of two regular languages is in* ExpTime *in the size of their recognising semigroups, and so in* 2ExpTime *in the size of the NFAs specifying the given languages.*

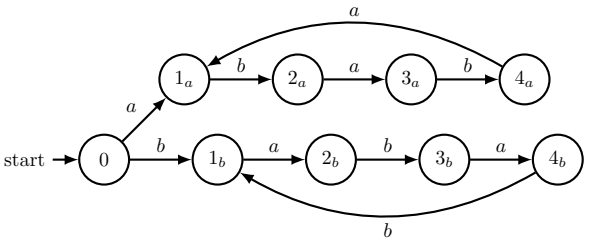
By Theorem 36, this problem is PSpace-hard in the size of the DFAs. We illustrate the algorithm by two examples.

▶ **Example 39.** It is well known (see, e.g., [71]) that

there is no $\mathsf{FO}(<)$-sentence expressing parity of the length of a finite strict linear order. (2)

Now, let $\boldsymbol{L}_1 = (abab)^+$ and $\boldsymbol{L}_2 = (baba)^+$. An easy adaptation of the usual Ehrenfeucht-Fraïssé argument for (2) shows that these languages are not $\mathsf{FO}(<)$-definable. On the other hand, the language $\boldsymbol{L} = (ab)^+$ clearly separates them, and it is not hard to see that $\boldsymbol{L}$ is $\mathsf{FO}(<)$-definable: it is the set of all words that start with an a, end with a b, and do not contain aa or bb as a subword.

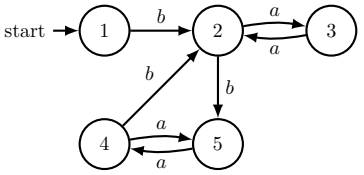
As concerns the algorithm detecting this, observe first that both languages are recognised by the transition semigroup $\boldsymbol{S}$ of the DFA depicted below, with 4_a as an accepting state for $\boldsymbol{L}_1$, 4_b as an accepting state for $\boldsymbol{L}_2$, and the missing transitions going to a 'sink' state $\perp$.



The recognising homomorphism α maps any $w \in A^+$ to the state-to-state transition function δ_w. It is easy to see that $S = \{\delta_a, \delta_b, \delta_{aa}, \delta_{ab}, \delta_{ba}, \delta_{aba}, \delta_{bab}, \delta_{abab}, \delta_{baba}\}$, with idempotent elements δ_{abab}, δ_{baba} and δ_{aa} (with δ_{aa} being the constant $\perp$-function), and $i_{\boldsymbol{S}} = 2$. Further, $\boldsymbol{L}_1 = \alpha^{-1}(\{\delta_{abab}\})$ and $\boldsymbol{L}_2 = \alpha^{-1}(\{\delta_{baba}\})$. Also, it is not hard to see that the only non-singleton elements of $\boldsymbol{S}^\dagger$ are $\{\delta_{abab}, \delta_{ab}\}$, $\{\delta_a, \delta_{aba}\}$, $\{\delta_{baba}, \delta_{ba}\}$, and $\{\delta_b, \delta_{bab}\}$. Therefore, $\{\delta_{abab}, \delta_{baba}\} \notin \boldsymbol{S}^\dagger$, and so $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ are $\mathsf{FO}(<)$-separable by the criterion (1). ⌟

The next example is borrowed from [89]:

▶ **Example 40.** Let $\boldsymbol{L}_1 = \big(b(aa)^*b(aa)^*a\big)^+$ and $\boldsymbol{L}_2 = \big(b(aa)^*b(aa)^*a\big)^*b(aa)^*$ (where $*$ is the 'real' Kleene star). Again, an easy adaptation of the Ehrenfeucht-Fraïssé argument proving (2) shows that $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ are not $\mathsf{FO}(<)$-separable. To explain the algorithm detecting this, observe first that both languages are recognised by the transition semigroup $\boldsymbol{S}$ of the DFA below, with 4 as an accepting state for $\boldsymbol{L}_1$, 2 as an accepting state for $\boldsymbol{L}_2$, and the missing transitions going to a 'sink' state $\perp$.



The recognising homomorphism α maps any $w \in A^+$ to the state-to-state transition function δ_w. It is easy to see that $\boldsymbol{L}_1 = \alpha^{-1}(\{\delta_{b^2a}\})$ and $\boldsymbol{L}_2 = \alpha^{-1}(\{\delta_b, \delta_{b^2ab}\})$. We claim that $\{\delta_{b^2a}, \delta_{b^2ab}\}$ in $\boldsymbol{S}^\dagger$, and so $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ are not $\mathsf{FO}(<)$-separable by the criterion (1). Indeed,

observe that $i_{\boldsymbol{S}} = 3$, so $\{\delta_a\}^{i_{\boldsymbol{S}}} = \{\delta_a\}$ and $\{\delta_a\}^{i_{\boldsymbol{S}}+1} = \delta_{a^2}$. Thus, $\{\delta_a, \delta_{a^2}\} \in \boldsymbol{S}^\dagger$ by C3. Then, by C2, we obtain that $\{\delta_a, \delta_{a^2}\} \cdot \{\delta_b\} = \{\delta_{ab}, \delta_{a^2b}\} \in \boldsymbol{S}^\dagger$. Let $X = \{\delta_{ab}, \delta_{a^2b}\}$. As $\delta_{a^2b} \cdot \delta_{ab} \cdot \delta_{a^2b} = \delta_{bab^2}$ and $\delta_{a^2b} \cdot \delta_{ab} \cdot \delta_{a^2b} \cdot \delta_{ab} = \delta_{bab}$, we have $\delta_{bab^2}, \delta_{bab} \in X^{i_{\boldsymbol{S}}} \cup X^{i_{\boldsymbol{S}}+1}$, and so $\{\delta_{bab^2}, \delta_{bab}\} \in \boldsymbol{S}^\dagger$ by C3 and C1. And as $\delta_b \cdot \delta_{bab^2} \cdot \delta_a = \delta_{b^2a}$ and $\delta_b \cdot \delta_{bab} \cdot \delta_{aa} = \delta_{b^2ab}$, we obtain that $\{\delta_{b^2a}, \delta_{b^2ab}\} \in \boldsymbol{S}^\dagger$ by C2 and C1, as required. ⌟

Constructing a possible separator. The following is obtained by generalising Wilke's [107] construction of FO-definitions for FO(<)-definable languages:

▶ **Theorem 41** ([89]). *Suppose $\boldsymbol{L}_1, \boldsymbol{L}_2 \subseteq A^+$ are* FO(<)*-separable languages that are recognised by a finite semigroup $\boldsymbol{S}$. Then one can construct an* FO*-sentence φ in the signature σ_A such that $\boldsymbol{L} = \{w \in \Sigma^+ \mid w \models \varphi\}$ separates $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$, and the quantifier depth of φ is linear in $|A|$ and exponential in the size of $\boldsymbol{S}$ (and so double-exponential in the size of the DFAs specifying $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$).*

Languages over Infinite Words. According to [89], the analogues of Theorems 37, 38, and 41 also hold for ω-languages.

Definability and Separability of Regular Languages by 'Simpler' Languages. Several natural hierarchies of star-free languages have been introduced within each of the automata-theoretic, algebraic and logical approaches. There has been a lot of work on the connections among these and, using these connections, on deciding the corresponding definability and separation problems; see, e.g., [90] for a relatively recent survey on definability and [91] for recent results on separation.

Definability and Separability of Non-Regular Languages. The definability and separability problems for languages specified by devices more powerful than finite automata (such as automata with counters, Parikh automata, vector addition systems with states, etc.) have been mostly studied for the case when the defining/separating language is regular. Even these questions attracted much less attention than defining/separating regular languages, possibly because of the lack of algebraic methods and the scarcity of positive results. A famous early negative result is that separability of context-free languages by a regular one is undecidable [101]. As most studied non-regular language-classes are not closed under complementation, definability for these cannot be reduced to separation. In fact, there are cases when the separation problem is decidable, while the definability problem is undecidable; see, e.g., [35] for examples and a wider context.

6 Back Again: Interpolant Existence in Linear Temporal Logic

In this final section, we use the decidability of FO(<)-separability of regular languages (cf. Theorems 34, 35 and 37) to decide the IEP for propositional linear temporal logic LTL [43, 36] over finite timelines, which is known to lack the CIP [75] (see Example 45 below).

First, we briefly remind the reader of the relevant definitions. LTL-*formulas* are built from a countably-infinite set $p_0, p_1, \dots$ of propositional variables using the Booleans and temporal operators $\bigcirc$ (at the next moment), $\Diamond$ (some time in the future) and $\mathcal{U}$ (until). We interpret them in *temporal models* that are Kripke models of the form

$$\mathfrak{M} = \big(\{0, \dots, \ell\}, <, \mathfrak{v}_{\mathfrak{M}}\big),$$

where $\ell < \omega$ and $<$ is the standard strict linear order. For any set ϱ of propositional variables, we call a temporal model $\mathfrak{M}$ a ϱ-*model* if $\mathfrak{v}_{\mathfrak{M}}(p_i) = \emptyset$, for any $p_i \notin \varrho$. The *truth-relation* in $\mathfrak{M}$ (under the *strict semantics* of $\Diamond$ and $\mathcal{U}$) is defined by taking, for every $n \leq \ell$ and every propositional variable p_i,

- $\mathfrak{M}, n \models p_i$ iff $n \in \mathfrak{v}_{\mathfrak{M}}(p_i)$,
- $\mathfrak{M}, n \models \bigcirc\varphi$ iff $n < \ell$ and $\mathfrak{M}, n+1 \models \varphi$,
- $\mathfrak{M}, n \models \Diamond\varphi$ iff $\mathfrak{M}, m \models \varphi$, for some m with $n < m \leq \ell$,
- $\mathfrak{M}, n \models \varphi \,\mathcal{U}\, \psi$ iff there is m such that $n < m \leq \ell$, $\mathfrak{M}, m \models \varphi$ and $\mathfrak{M}, k \models \psi$ for all k with $n < k < m$.

Observe that temporal models can be regarded as σ_{lin}-structures in the FO-signature σ_{lin} that consists of a binary predicate symbol $<$ and a unary predicate symbol P_i, for each propositional variable p_i. Thus, similarly to the *standard translation* for modal logic, the truth-conditions above can be expressed in the FO-language in σ_{lin}. Moreover, there is a translation backwards, as the following generalisation of Kamp's Theorem [61] by Gabbay, Pnueli, Shelah and Stavi [45] states:

▶ **Theorem 42** ([61, 45]). *For every* FO*-formula $\vartheta(x)$ in the signature σ_{lin} having one free variable x, there is an* LTL*-formula φ_ϑ such that $\mathfrak{M} \models \vartheta(0)$ iff $\mathfrak{M}, 0 \models \varphi_\vartheta$, for all temporal models $\mathfrak{M}$.*

Note that φ_ϑ is of non-elementary size in the size of ϑ in the worst case [99]. If we consider the symbols of an alphabet A as propositional variables, then any word $w \in A^+$ can be regarded as a temporal model. We say that a language $\boldsymbol{L} \subseteq A^+$ is LTL-*definable* if there is an LTL-formula φ such that $\boldsymbol{L} = \{w \in A^+ \mid w, 0 \models \varphi\}$.

▶ **Corollary 43.** *A language is* FO(<)*-definable iff it is* LTL*-definable.*

On the other hand, Theorems 34 and 42 suggest a characterisation of regular languages in terms of existentially quantified LTL-formulas. More precisely, define ∃LTL as the language that comprises LTL-formulas possibly prefixed by quantifiers $\exists q$ over propositional variables q that are interpreted by taking $\mathfrak{M}, n \models \exists q\, \varphi$ iff $\mathfrak{M}', n \models \varphi$, for some $\mathfrak{M}'$ that may only differ from $\mathfrak{M}$ on the interpretation of q. Observe that the truth-condition for $\exists q_1 \dots \exists q_n \varphi$ can be expressed by an MSO-formula of the form $\exists Q_1 \dots \exists Q_n\, \psi_\varphi$, where ψ_φ is the standard translation (in the signature σ_{lin}) for φ, and each Q_i is the unary predicate symbol corresponding to q_i. Moreover, the following is a consequence of Theorems 34 and 42:

▶ **Theorem 44.**

(i) *For every* MSO*-formula* $\vartheta(x)$ *in the signature* σ_{lin} *having one free first-order variable* x*, there is a* $\exists$LTL*-formula* φ_ϑ *such that* $\mathfrak{M} \models \vartheta(0)$ *iff* $\mathfrak{M}, 0 \models \varphi_\vartheta$*, for all temporal models* $\mathfrak{M}$*. It follows that:*

(ii) *A language is regular iff it is* $\exists$LTL*-definable.*

Given LTL or $\exists$LTL-formulas φ, ψ, we write $\varphi \models_{\mathsf{LTL}} \psi$ whenever $\mathfrak{M}, 0 \models \varphi$ implies $\mathfrak{M}, 0 \models \psi$, for every $\mathfrak{M}$. (Note that this is equivalent to the definition of $\models_L$ for modal logics given in Section 2.)

▶ **Example 45.** We show two LTL-formulas that do not have a Craig interpolant in LTL. Consider

$$\varphi = p \wedge \Box^+(p \wedge \bigcirc\top \leftrightarrow \bigcirc\neg p) \wedge \Diamond(\neg p \wedge \neg\bigcirc\top), \quad \psi = q \wedge \Box^+(q \wedge \bigcirc\top \leftrightarrow \bigcirc\neg q) \to \Diamond(\neg q \wedge \neg\bigcirc\top).$$

Then, for every $\mathfrak{M}$, we have

$$\mathfrak{M}, 0 \models \varphi \ \text{ iff } \ p \text{ holds in } \mathfrak{M} \text{ at even numbers and the number of points in } \mathfrak{M} \text{ is even,}$$

and so $\varphi \models_{\mathsf{LTL}} \psi$. Any interpolant χ for φ, ψ in LTL should be variable-free and express that the number of points in an $\emptyset$-model is even; the standard translation of such a χ should be an FO(<)-sentence expressing parity of the length of a finite strict linear order, which is impossible by (2). ⌟

So far, the decidability status of the IEP for LTL has been open, with the methods of analysing the problem discussed in Section 3 being non-applicable in this case. The next theorem settles this open question by employing Theorem 42 to reduce—for the price of a multi-exponential blow-up—the IEP for LTL to FO(<)-separability of regular languages and then using Theorems 35, 37 and 38.

▶ **Theorem 46.** *The IEP for* LTL *is decidable in* 3ExpTime*, being* PSpace*-hard.*

Proof. Suppose we are given LTL-formulas φ and ψ whose shared propositional variables comprise a set ϱ. Our aim is to decide whether there exists an LTL-formula χ built from variables in ϱ such that $\varphi \models_{\mathsf{LTL}} \chi$ and $\chi \models_{\mathsf{LTL}} \psi$. To this end, we take the alphabet $A_\varrho = 2^\varrho$ and observe that any ϱ-model $\mathfrak{M}$ can be regarded as a word $w \in A_\varrho^+$, where $w = \boldsymbol{a}_0 \dots \boldsymbol{a}_\ell$, for $\boldsymbol{a}_i = \{p \in \varrho \mid i \in \mathfrak{v}_{\mathfrak{M}}(p)\}$. Let $q_1, \dots, q_m$ and $r_1, \dots, r_k$ be the respective lists of variables in φ and ψ that are not in ϱ. We define two languages $\boldsymbol{L}_\varphi, \boldsymbol{L}_{\neg\psi} \subseteq A_\varrho^+$ by taking

$$\boldsymbol{L}_\varphi = \{\varrho\text{-model } \mathfrak{M} \mid \mathfrak{M}, 0 \models \exists q_1 \dots \exists q_m\, \varphi\}, \quad \boldsymbol{L}_{\neg\psi} = \{\varrho\text{-model } \mathfrak{M} \mid \mathfrak{M}, 0 \models \exists r_1 \dots \exists r_k\, \neg\psi\}.$$

By Theorem 44.ii, $\boldsymbol{L}_\varphi$ and $\boldsymbol{L}_{\neg\psi}$ are both regular languages. We claim that

$$\text{there is a Craig interpolant for } \varphi, \psi \text{ in LTL} \quad \text{iff} \quad \boldsymbol{L}_\varphi \text{ and } \boldsymbol{L}_{\neg\psi} \text{ are FO(<)-separable.} \tag{3}$$

Observe that if χ is any LTL-formula not containing the variables $q_1, \dots, q_m, r_1, \dots, r_k$, then

$$\varphi \models_{\mathsf{LTL}} \chi \text{ iff } \exists q_1 \dots \exists q_m\, \varphi \models_{\mathsf{LTL}} \chi, \qquad \text{and} \qquad \chi \models_{\mathsf{LTL}} \psi \text{ iff } \chi \models_{\mathsf{LTL}} \neg\exists r_1 \dots \exists r_k \neg\psi. \quad (4)$$

To prove (3), suppose first that χ is an LTL-formula built up from variables in ϱ such that $\varphi \models_{\mathsf{LTL}} \chi$ and $\chi \models_{\mathsf{LTL}} \psi$. We define a language $\boldsymbol{L}_\chi \subseteq A_\varrho^+$ as $\boldsymbol{L}_\chi = \{\varrho\text{-model } \mathfrak{M} \mid \mathfrak{M}, 0 \models \chi\}$. By (4), $\exists q_1 \dots \exists q_m\, \varphi \models_{\mathsf{LTL}} \chi$ and $\chi \models_{\mathsf{LTL}} \neg\exists r_1 \dots \exists r_k \neg\psi$, and so $\boldsymbol{L}_\varphi \subseteq \boldsymbol{L}_\chi$ and $\boldsymbol{L}_{\neg\psi} \cap \boldsymbol{L}_\chi = \emptyset$. Thus, $\boldsymbol{L}_\chi$ separates $\boldsymbol{L}_\varphi$ and $\boldsymbol{L}_{\neg\psi}$. Moreover, $\boldsymbol{L}_\chi$ is $\mathsf{FO}(<)$-definable. Indeed, the standard translation of χ is an equivalent FO-formula $\vartheta_\chi(x)$ in the signature σ_{lin} such that every predicate symbol P in ϑ_χ corresponds to some $p \in \varrho$. By replacing each subformula in ϑ_χ of the form $P(y)$, $p \in \varrho$, with $\bigvee_{\boldsymbol{a} \in A_\varrho,\, p \in \boldsymbol{a}} \boldsymbol{a}(y)$, we obtain an FO-formula $\vartheta'_\chi(x)$ in the signature σ_{A_ϱ} that is equivalent to $\vartheta_\chi(x)$ over ϱ-models.

Conversely, suppose that $\boldsymbol{L}_\varphi$ and $\boldsymbol{L}_{\neg\psi}$ are separated by an $\mathsf{FO}(<)$-definable language $\boldsymbol{L}$, and ϑ is a FO-sentence in the signature σ_{A_ϱ} such that $\boldsymbol{L} = \{\varrho\text{-model } \mathfrak{M} \mid \mathfrak{M} \models \vartheta\}$. By replacing each subformula of ϑ of the form $\boldsymbol{a}(y)$ by $\bigwedge_{p \in \boldsymbol{a}} P(y)$, we obtain an FO-sentence ϑ^- in the signature σ_{lin}, containing only unary predicates P for $p \in \varrho$, that is equivalent to ϑ over ϱ-models. By Theorem 42, there is some LTL-formula χ_{ϑ^-} containing only propositional variables from ϱ and such that $\mathfrak{M} \models \vartheta^-$ iff $\mathfrak{M}, 0 \models \chi_{\vartheta^-}$, for all ϱ-models $\mathfrak{M}$. Now, it follows from $\boldsymbol{L}_\varphi \subseteq \boldsymbol{L}$ that $\exists q_1 \dots \exists q_m\, \varphi \models_{\mathsf{LTL}} \chi_{\vartheta^-}$, and so $\varphi \models_{\mathsf{LTL}} \chi_{\vartheta^-}$ by (4). Similarly, it follows from $\boldsymbol{L}_{\neg\psi} \cap \boldsymbol{L} = \emptyset$ that $\chi_{\vartheta^-} \models_{\mathsf{LTL}} \neg\exists r_1 \dots \exists r_k \neg\psi$, and so $\chi_{\vartheta^-} \models_{\mathsf{LTL}} \psi$ by (4). Therefore, χ_{ϑ^-} is an interpolant for φ, ψ in LTL.

As for the computational complexity of deciding the IEP for LTL, the above reduction gives a 3ExpTime upper bound: the size of an NFA $\mathfrak{A}$ specifying the language defined by an LTL-formula φ is exponential in the size of φ [108], the NFA specifying $\boldsymbol{L}_\varphi$ is a projection of $\mathfrak{A}$ [97], and so has size linear in the size of $\mathfrak{A}$; and then two more exponentials come from Theorem 38. ◀

It is to be noted that, conversely, $\mathsf{FO}(<)$-separability of regular languages can also be reduced to the IEP for LTL by using Corollary 43 and Theorem 44. The exact computational complexities of both of these decision problems remain open.

LTL over $(\omega, <)$. LTL-formulas can also be interpreted in infinite temporal models of the form $(\omega, <, \mathfrak{v})$. We denote the resulting logic by LTL_ω. This kind of temporal models can also be regarded as σ_{lin}-structures of the FO-signature σ_{lin} above. As discussed above, the analogues of Theorems 35, 37, and 38 hold for ω-languages. Moreover, Theorem 42 also holds in this setting, and so a proof similar to that of Theorem 46 shows that the IEP for LTL_ω is also decidable, with its exact complexity remaining also open.

Fragments of LTL. Denote by $\mathsf{LTL}[\Diamond]$ the fragment of LTL that only allows $\Diamond$ as a temporal operator, and by $\mathsf{LTL}[\bigcirc\Diamond]$ the fragment, in which both $\bigcirc$ and $\Diamond$ are allowed. Neither of them enjoys the CIP. Using the method outlined in Section 3, it is shown in [68] that the IEP for $\mathsf{LTL}[\Diamond]$ over both finite and infinite linear timelines is coNP-complete, that is, as complex as validity [96]. The decidability of the IEP for $\mathsf{LTL}[\bigcirc\Diamond]$ over finite and infinite

linear timelines can be established via the 'detour' to language separation described earlier in this section, using the following two facts. First, every $\mathsf{FO}^2[<, suc]$-formula with one free variable is equivalent to an $\mathsf{LTL}[\bigcirc\Diamond]$-formula [40, 45, 76]. And second, it is decidable whether two regular languages are separable by an $\mathsf{FO}^2[<, suc]$-definable language [88]. The exact complexity of the IEP for $\mathsf{LTL}[\bigcirc\Diamond]$ is open.

It is also of interest to note that quantified LTL (with both existential and universal quantification over propositional variables) is the smallest 'natural' logic containing $\mathsf{LTL}[\Diamond]$ and having the CIP [46].

7 Research Directions and Open Problems

In this chapter, we have provided an overview of recent research into variations and generalisations of Craig interpolation, the common feature of which is that the existence of an interpolant does not reduce to entailment, either because the logic in question does not enjoy the CIP, or because the use of logical symbols in the interpolant is restricted. Numerous research questions remain to be explored. We conclude this chapter by discussing a few of them:

The role of proof theory. The methods presented in this chapter have been mainly model-theoretic or automata-based. In fact, proof theory has not yet been applied to decide interpolant existence or construct interpolants for logics without the CIP or for separation. Developing proof-theoretic methods would be of great interest, in particular since very little is known about the actual construction of interpolants and separators, and proof-theoretic methods are a promising tool to address this problem as demonstrated by their wide applications for logics with the CIP, both in theory and practice.

Non-classical logics. The focus of this chapter has been on modal and first-order logics, and their extensions with second-order quantifiers and fixed points. We have not discussed important families of non-classical logics such as superintuitionistic, many-valued, substructurual, or relevant logics. To the best of our knowledge the variations of Craig interpolation investigated in this chapter have not yet been considered for these logics, despite the fact that many non-classical logics do not enjoy the CIP and that it appears natural to compare the expressive power of different non-classical logics by studying separation.

General results. Table 1 shows first results on computational complexity of the IEP, but there are large gaps. For instance, except for $\mathsf{S5}$, hardly anything is known about the IEP or separation in first-order modal and temporal logic, despite the fact that under constant domain semantics they all lack the CIP. Similarly, are there general results about the IEP for logics containing $\mathsf{S4}$ or $\mathsf{K4}$, extending the result for logics above $\mathsf{K4.3}$? Again, this would be of interest since at most 36 extensions of $\mathsf{S4}$ have the CIP. Finally, although a few results have been discussed, the decidability status for interpolant existence for many decidable fragments of FO remains open. An example is the two-variable fragment of FO with counting.

Acknowledgments

The authors thank Jean Christoph Jung and Jędrzej Kołodziejski for their valuable comments and suggestions.

References

1 Jorge Almeida. Some algorithmic problems for pseudovarieties. *Publ. Math. Debrecen*, 54:531–552, 1999.

2 Hajnal Andréka, Stephen D. Comer, Judit X. Madarász, István Németi, and Tarek Sayed-Ahmed. Epimorphisms in cylindric algebras and definability in finite variable logic. *Algebra Univers.*, 61:261–282, 2009.

3 Hajnal Andréka, Stephen D. Comer, and István Németi. Epimorphisms in cylindric algebras. Technical report, Mathematical Institute of the Hungarian Academy of Sciences, Budapest, 1983.

4 Hajnal Andréka, István Németi, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *J. Philosophical Logic*, 27(3):217–274, 1998.

5 Hajnal Andréka, Johan van Benthem, and István Németi. Back and forth between modal logic and classical logic. *Log. J. IGPL*, 3(5):685–720, 1995. doi:10.1093/JIGPAL/3.5.685.

6 Carlos Areces, Patrick Blackburn, and Maarten Marx. A road-map on complexity for hybrid logics. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *Computer Science Logic, 13th International Workshop, CSL '99, 8th Annual Conference of the EACSL, Madrid, Spain, September 20-25, 1999, Proceedings*, volume 1683 of *Lecture Notes in Computer Science*, pages 307–321. Springer, 1999. doi:10.1007/3-540-48168-0_22.

7 Alessandro Artale, Jean Christoph Jung, Andrea Mazzullo, Ana Ozaki, and Frank Wolter. Living without Beth and Craig: Definitions and interpolants in description and modal logics with nominals and role inclusions. *ACM Trans. Comput. Log.*, 24(4):34:1–34:51, 2023.

8 Alessandro Artale, Andrea Mazzullo, Ana Ozaki, and Frank Wolter. On free description logics with definite descriptions. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, pages 63–73, 2021.

9 Alexandru Baltag, Nick Bezhanishvili, and David Fernández-Duque. The topological mu-calculus: Completeness and decidability. *J. ACM*, 70(5):33:1–33:38, 2023. doi:10.1145/3623268.

10 Jon Barwise and Yiannis N. Moschovakis. Global inductive definability. *J. Symb. Log.*, 43(3):521–534, 1978. doi:10.2307/2273529.

11 Michael Benedikt, Pierre Bourhis, Georg Gottlob, and Pierre Senellart. Monadic datalog, tree validity, and limited access containment. *ACM Trans. Comput. Log.*, 21(1):6:1–6:45, 2020. doi:10.1145/3344514.

12 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. Definability and interpolation within decidable fixpoint logics. *Log. Methods Comput. Sci.*, 15(3), 2019. doi:10.23638/LMCS-15(3:29)2019.

13 Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *30th Annual ACM/IEEE Symposium on*

Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015, pages 293–304. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.36.

14 Michael Benedikt, Egor V. Kostylev, Fabio Mogavero, and Efthymia Tsamoura. Reformulating queries: Theory and practice. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 837–843. ijcai.org, 2017. doi:10.24963/IJCAI.2017/116.

15 Michael Benedikt, Julien Leblay, Balder ten Cate, and Efthymia Tsamoura. *Generating Plans from Proofs: The Interpolation-based Approach to Query Reformulation*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2016.

16 Johan van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, Napoli, 1983.

17 László Bernátsky. Regular expression star-freeness is PSPACE-complete. *Acta Cybern.*, 13(1):1–21, 1997.

18 Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.

19 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

20 Achim Blumensath and Felix Wolf. Bisimulation invariant monadic-second order logic in the finite. *Theor. Comput. Sci.*, 823:26–43, 2020. doi:10.1016/J.TCS.2020.03.001.

21 Mikołaj Bojańczyk. Languages recognised by finite semigroups, and their generalisations to objects such as trees and graphs, with an emphasis on definability in monadic second-order logic. *CoRR*, abs/2008.11635, 2020.

22 Mikołaj Bojańczyk. Algebra for trees. In Jean-Éric Pin, editor, *Handbook of Automata Theory*, pages 801–838. European Mathematical Society Publishing House, Zürich, Switzerland, 2021.

23 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

24 Julian C. Bradfield and Colin Stirling. Modal mu-calculi. In Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 721–756. North-Holland, 2007. doi:10.1016/S1570-2464(07)80015-2.

25 J.R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1–6):66–92, 1960.

26 Balder ten Cate. Interpolation for extended modal languages. *J. Symb. Log.*, 70(1):223–234, 2005.

27 Balder ten Cate and Jesse Comer. Craig interpolation for decidable first-order fragments. In Naoki Kobayashi and James Worrell, editors, *Foundations of Software Science and Computation Structures - 27th International Conference, FoSSaCS 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part II*, volume 14575 of *Lecture Notes in Computer Science*, pages 137–159. Springer, 2024. doi:10.1007/978-3-031-57231-9_7.

28 Alexander V. Chagrov and Michael Zakharyaschev. *Modal Logic*, volume 35 of *Oxford logic guides*. Oxford University Press, 1997.

29 Chen Chung Chang and H. Jerome Keisler. *Model Theory*. Elsevier, 1998.

30 Sang Cho and Dung T. Huynh. Finite-automaton aperiodicity is PSPACE-complete. *Theor. Comput. Sci.*, 88(1):99–116, 1991.

31 Thomas Colcombet, Amina Doumane, and Denis Kuperberg. Tree algebras and bisimulation-invariant MSO on finite graphs. *CoRR*, abs/2407.12677, 2024. doi:10.48550/ARXIV.2407.12677.

32 Stephen D. Comer. Classes without the amalgamation property. *Pacific J. Math.*, 28(2):309–318, 1969.

33 Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 477–490. ACM, 1988. doi:10.1145/62212.62259.

34 William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. Symb. Log.*, 22(3):269–285, 1957. doi:10.2307/2963594.

35 Wojciech Czerwiński and Sławomir Lasota. Regular separability of one counter automata. *Log. Methods Comput. Sci.*, 15(2), 2019.

36 Stéphane Demri, Valentin Goranko, and Martin Lange. *Temporal Logics in Computer Science: Finite-State Systems*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016.

37 Volker Diekert and Paul Gastin. First-order definable languages. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2008.

38 Calvin Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–51, 1961.

39 Leo Esakia. Weak transitivity—a restitution. *Logical Investigations*, 8:244–255, 2001. (In Russian).

40 Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2):279–295, 2002.

41 Kit Fine. Failures of the interpolation lemma in quantified modal logic. *J. Symb. Log.*, 44(2):201–206, 1979.

42 Marie Fortin, Boris Konev, and Frank Wolter. Interpolants and explicit definitions in extensions of the description logic $\mathcal{EL}$. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022*, 2022.

43 Dov M. Gabbay, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. *Many-dimensional Modal Logics: Theory and Applications*, volume 148 of *Studies in Logic and The Foundations of Mathematics*. Elsevier, 2003.

44 Dov M. Gabbay and Larisa Maksimova. *Interpolation and Definability: Modal and Intuitionistic Logics*. Oxford University Press, 2005.

45 Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. In Paul W. Abrahams, Richard J. Lipton, and Stephen R. Bourne, editors, *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages, Las Vegas, Nevada, USA, January 1980*, pages 163–173. ACM Press, 1980.

46 Amélie Gheerbrant and Balder ten Cate. Craig interpolation for linear temporal languages. In Erich Grädel and Reinhard Kahle, editors, *Computer Science Logic, 23rd international*

Workshop, CSL 2009, 18th Annual Conference of the EACSL, Coimbra, Portugal, September 7-11, 2009. Proceedings, volume 5771 of *Lecture Notes in Computer Science*, pages 287–301. Springer, 2009. doi:10.1007/978-3-642-04027-6_22.

47 Valentin Goranko and Martin Otto. Model theory of modal logic. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 249–329. Elsevier, 2007.

48 Erich Grädel, Colin Hirsch, and Martin Otto. Back and forth between guarded and modal logics. *ACM Trans. Comput. Log.*, 3(3):418–463, 2002. doi:10.1145/507382.507388.

49 Lauri Hella and Miikka Vilander. Formula size games for modal logic and μ-calculus. *J. Log. Comput.*, 29(8):1311–1344, 2019. doi:10.1093/LOGCOM/EXZ025.

50 Karsten Henckell. Pointlike sets: the finest aperiodic cover of a finite semigroup. *J. Pure Appl. Algebra*, 55(1-2):85–126, 1988.

51 Ian Hodkinson. Characterizations of two basic hybrid logics. *Journal of Logic and Computation*, 35(4):exaf025, 2025. doi:10.1093/logcom/exaf025.

52 Marco Hollenberg. *Logic and Bisimulation*. PhD thesis, University of Utrecht, 1998.

53 Eva Hoogland, Maarten Marx, and Martin Otto. Beth definability for the guarded fragment. In Harald Ganzinger, David A. McAllester, and Andrei Voronkov, editors, *Logic Programming and Automated Reasoning, 6th International Conference, LPAR'99, Tbilisi, Georgia, September 6-10, 1999, Proceedings*, volume 1705 of *Lecture Notes in Computer Science*, pages 273–285. Springer, 1999. doi:10.1007/3-540-48242-3_17.

54 John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation, 3rd Edition*. Pearson international edition. Addison-Wesley, 2007.

55 David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26-29, 1996, Proceedings*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1996. doi:10.1007/3-540-61604-7_60.

56 Jean Christoph Jung and Jędrzej Kołodziejski. Modal separability of fixpoint formulae. In Laura Giordano, Jean Christoph Jung, and Ana Ozaki, editors, *Proceedings of the 37th International Workshop on Description Logics (DL 2024), Bergen, Norway, June 18-21, 2024*, volume 3739 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3739/paper-5.pdf.

57 Jean Christoph Jung and Jędrzej Kołodziejski. Modal separation of fixpoint formulae. In *42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025)*. LIPIcs, 2025. available from https://logic-in.cs.tu-dortmund.de/forschung/knowledge-based-systems/publikationen/.

58 Jean Christoph Jung, Jędrzej Kołodziejski, and Frank Wolter. Computation of interpolants for description logic concepts in hard cases. *arXiv preprint arXiv:2507.15689*, 2025.

59 Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. Logical separability of labeled data examples under ontologies. *Artif. Intell.*, 313:103785, 2022. doi:10.1016/J.ARTINT.2022.103785.

60 Jean Christoph Jung and Frank Wolter. Living without Beth and Craig: Definitions and interpolants in the guarded and two-variable fragments. In *36th Annual ACM/IEEE Symposium*

on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021, pages 1–14. IEEE, 2021. doi:10.1109/LICS52264.2021.9470585.

61 Hans W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Computer Science Department, University of California at Los Angeles, USA, 1968.

62 A. Karpenko and Larisa Maksimova. Simple weakly transitive modal algebras. *Algebra and Logic*, 49(3):233–245, 2010.

63 Stanislav Kikot, Agi Kurucz, Vladimir V. Podolskii, and Michael Zakharyaschev. Deciding boundedness of monadic sirups. In Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo, editors, *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 370–387. ACM, 2021. doi:10.1145/3452021.3458332.

64 Phokion G. Kolaitis and Martin Otto. On the boundedness problem for two-variable first-order logic. In *Thirteenth Annual IEEE Symposium on Logic in Computer Science, Indianapolis, Indiana, USA, June 21-24, 1998*, pages 513–524. IEEE Computer Society, 1998. doi:10.1109/LICS.1998.705684.

65 Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Formal properties of modularisation. In Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, editors, *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*, pages 25–66. Springer, 2009. doi:10.1007/978-3-642-01907-4_3.

66 Louwe Kuijer, Tony Tan, Frank Wolter, and Michael Zakharyaschev. Separation and definability in fragments of two-variable first-order logic with counting. *CoRR*, abs/2504.20491, 2025. doi:10.48550/ARXIV.2504.20491.

67 Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. Definitions and (uniform) interpolants in first-order modal logic. In Pierre Marquis, Tran Cao Son, and Gabriele Kern-Isberner, editors, *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023*, pages 417–428, 2023.

68 Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. A non-uniform view of Craig interpolation in modal logics with linear frames. *CoRR*, abs/2312.05929, 2023.

69 Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. The interpolant existence problem for weak K4 and difference logic. In Agata Ciabattoni, David Gabelaia, and Igor Sedlár, editors, *Advances in Modal Logic, AiML 2024, Prague, Czech Republic, August 19-23, 2024*, pages 465–484. College Publications, 2024.

70 Richard E. Ladner. Application of model theoretic games to discrete linear orders and finite automata. *Inf. Control.*, 33(4):281–303, 1977.

71 Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

72 Carsten Lutz, Robert Piro, and Frank Wolter. Description Logic TBoxes: Model-Theoretic Characterizations and Rewritability. In *IJCAI*, pages 983–988, 2011.

73 Carsten Lutz and Leif Sabellek. A complete classification of the complexity and rewritability of ontology-mediated queries based on the description logic EL. *Artif. Intell.*, 308:103709, 2022. doi:10.1016/J.ARTINT.2022.103709.

74 Larisa Maksimova. Temporal logics with the next operator do not have interpolation or Beth property. *Siberian Mathematical Journal*, 32(6):989–993, 1991.

75 Larisa Maksimova. Temporal logics of "the next" do not have the Beth property. *J. Appl. Non Class. Logics*, 1(1):73–76, 1999. doi:10.1080/11663081.1991.10510772.

76 Nicolas Markey. Temporal logic with past is exponentially more succinct. *Bull. EATCS*, 79:122–128, 2003.

77 Maarten Marx and Carlos Areces. Failure of interpolation in combined modal logics. *Notre Dame J. Formal Log.*, 39(2):253–273, 1998. doi:10.1305/NDJFL/1039293067.

78 Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Inf. Control.*, 9(5):521–530, 1966.

79 Robert McNaughton and Seymour Papert. *Counter-free automata*. The MIT Press, 1971.

80 Jeffrey F. Naughton. Data independent recursion in deductive databases. *J. Comput. Syst. Sci.*, 38(2):259–289, 1989. doi:10.1016/0022-0000(89)90003-2.

81 Martin Otto. Eliminating recursion in the μ-calculus. In *STACS*, pages 531–540, 1999.

82 Martin Otto. Modal and guarded characterisation theorems over finite transition systems. *Ann. Pure Appl. Log.*, 130(1-3):173–205, 2004. doi:10.1016/J.APAL.2004.04.003.

83 Martin Otto. Graded modal logic and counting bisimulation. *CoRR*, abs/1910.00039, 2019. URL: http://arxiv.org/abs/1910.00039.

84 Dominique Perrin. Recent results on automata and infinite words. In Michal Chytil and Václav Koubek, editors, *Mathematical Foundations of Computer Science 1984, Praha, Czechoslovakia, September 3-7, 1984, Proceedings*, volume 176 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 1984.

85 Dominique Perrin and Jean-Eric Pin. *Infinite words - automata, semigroups, logic and games*, volume 141 of *Pure and applied mathematics series*. Elsevier Morgan Kaufmann, 2004.

86 Maximilian Pflueger, Johannes Marti, and Egor V. Kostylev. A characterisation theorem for two-way bisimulation-invariant monadic least fixpoint logic over finite structures. In Pawel Sobocinski, Ugo Dal Lago, and Javier Esparza, editors, *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024*, pages 63:1–63:14. ACM, 2024. doi:10.1145/3661814.3662107.

87 Don Pigozzi. Amalgamation, congruence-extension, and interpolation properties in algebras. *Algebra Univers.*, 1(1):269–349, 1971.

88 Thomas Place and Marc Zeitoun. Separation and the successor relation. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 662–675. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

89 Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *Log. Methods Comput. Sci.*, 12(1), 2016.

90 Thomas Place and Marc Zeitoun. Generic results for concatenation hierarchies. *Theory Comput. Syst.*, 63(4):849–901, 2019.

91 Thomas Place and Marc Zeitoun. Navigational hierarchies of regular languages. *CoRR*, abs/2402.10080, 2025.

92 Ian Pratt-Hartmann. *Fragments of First-Order Logic*. Oxford Logic Guides. Oxford University Press, United Kingdom, March 2023.

93 Eric Rosen. Modal logic over finite structures. *J. Log. Lang. Inf.*, 6(4):427–439, 1997. doi:10.1023/A:1008275906015.

94 Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Inf. Control.*, 8(2):190–194, 1965.

95 Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

96 A. Prasad Sistla and Edmund Clarke. The complexity of propositional linear temporal logics. *Journal of the Association for Computing Machinery*, 32:733–749, 1985.

97 A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The complementation problem for Büchi automata with appplications to temporal logic. *Theor. Comput. Sci.*, 49:217–237, 1987.

98 Jacques Stern. Complexity of some problems from the theory of automata. *Inf. Control.*, 66(3):163–176, 1985.

99 Larry J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. Phd thesis, Department of Electrical Engineering, MIT, 1974.

100 Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhauser Verlag, 1994.

101 Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM J. Comput.*, 5(2):231–250, 1976.

102 Wolfgang Thomas. Star-free regular sets of omega-sequences. *Inf. Control.*, 42(2):148–156, 1979.

103 Wolfgang Thomas. A combinatorial approach to the theory of omega-automata. *Inf. Control.*, 48(3):261–283, 1981.

104 Wolfgang Thomas. Logical aspects in the study of tree languages. In Bruno Courcelle, editor, *CAAP'84, 9th Colloquium on Trees in Algebra and Programming, Bordeaux, France, March 5-7, 1984, Proceedings*, pages 31–50. Cambridge University Press, 1984.

105 Boris Trakhtenbrot. Finite automata and the logic of one-place predicates. *Siberian Mathematical Journal*, 3:103–131, 1962. English translation in: AMS Transl. 59 (1966) 23–55.

106 Thomas Wilke. An Eilenberg theorem for infinity-languages. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, volume 510 of *Lecture Notes in Computer Science*, pages 588–599. Springer, 1991.

107 Thomas Wilke. Classifying discrete temporal properties. In Christoph Meinel and Sophie Tison, editors, *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, volume 1563 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 1999.

108 Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths (extended abstract). In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*, pages 185–194. IEEE Computer Society, 1983.

109 Frank Wolter. A note on the interpolation property in tense logic. *J. Philos. Log.*, 26(5):545–551, 1997. doi:10.1023/A:1017956722866.

110 Frank Wolter and Michael Zakharyaschev. Interpolant existence is undecidable for two-variable first-order logic with two equivalence relations. In Laura Giordano, Jean Christoph Jung, and Ana Ozaki, editors, *Proceedings of the 37th International Workshop on Description Logics (DL 2024), Bergen, Norway, June 18-21, 2024*, volume 3739 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3739/paper-11.pdf.

Chapter 11

Interpolation via Cyclic Proofs

Bahareh Afshari
University of Gothenburg, Sweden

Graham E. Leigh
University of Gothenburg, Sweden

Abstract

This chapter provides an expository account of cyclic proofs for fragments of the modal μ-calculus and demonstrates how the Maehara–Pitts method of interpolant construction can be lifted to the formalism. Existing results on separation and decidability in the modal μ-calculus are recast as consequences of the form of obtained interpolants. The wider applicability of the methods to other modal fixed point logics and μ-calculi is discussed.

Contents

1 Introduction 372
2 The modal μ-calculus 374
2.1 Syntax 374
2.2 Semantics 377
2.3 Important Fragments 379
3 Cyclic Derivations 381
3.1 Cyclic Proofs at the First Level 384
3.2 Proof Search 386
4 Lyndon Interpolation at the First Level 390
4.1 Lyndon Candidates 390
4.2 Lyndon Interpolation and Applications 394
5 Uniform Interpolation at the First Level 395
5.1 Uniform Candidates 397
5.2 Uniform Interpolation and Applications 400



© Bahareh Afshari and Graham E. Leigh;
licensed under Creative Commons License CC BY 4.0

In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 11, pages 371–413. Ubiquity Press, 2026. https://www.doi.org/10.5334/bdg.k

6 Cyclic Proofs and Interpolation 402
6.1 A Normal Form for Cyclic Derivations 404
6.2 Lyndon Interpolation for the Full Language 406
6.3 Uniform Interpolation for the Full Language 408
7 Discussion 410
Acknowledgments 411
References 411

1 Introduction

Many modal and temporal logics can be realised as extensions of a basic modal logic by certain inductive and co-inductive definitions. Common examples are *linear-time temporal logic* which extends the basic logic of a linear modality by the inductively defined 'until' operator, *propositional dynamic logic*, extending a multi-agent version of modal logic K by inductive operators for program recursion, and the *logic of common knowledge* which extends multi-agent epistemic logic by the co-inductive 'common knowledge' operator, as well as many *description logics* which might extend various modal logics by transitive concepts or even fixed point definitions. At the extreme are the so-called μ-calculi which incorporate explicit quantifiers for defining inductive and co-inductive operators of any (definable) specification. Linear-time temporal logic is, for example, a definable fragment of the linear-time μ-calculus and propositional dynamic logic is definable in the multi-modal μ-calculus. Other notable logics which are definable in the (single-agent) μ-calculus are *Gödel–Löb provability logic* and *compositional tree logic*.

Axiomatic systems for fixed point logics and μ-calculi invariably include axioms or rules of induction (and co-induction) to enforce the correct interpretation of operators. These rules present a challenge for the proof-theoretic approach to interpolation. Whether it be the Maehara method for Craig–Lyndon interpolation [20] or Pitts' construction of uniform interpolants [26], a clear obstacle for *any* logic is the isolation of a finitary sound and complete analytic[1] proof system. For the construction of uniform interpolants one expects, in addition, the analytic calculus to admit a terminating proof-search algorithm. General properties of a proof calculus that are sufficient for interpolation were explored in Chapter 5.

If the addition of induction principles to a logic does not outright block completeness of analytic proofs (as it does over first-order logic), it certainly complicates their design and theory. For many modal logics, analyticity can be regained by making structural changes

[1] For the purpose of the present chapter, it suffices to take *analytic* to mean that the premises of all rules employ the same (or smaller) signature than the conclusion, though stronger conditions are needed to ensure the existence of interpolants (cf. Chapter 5). Notably, this simple constraint already eliminates the most problematic inference rule from the perspective of interpolation, the cut rule, which permits the derivation of an implication $\alpha \to \beta$ from implications $\alpha \to \iota$ and $\iota \to \beta$ for arbitrary ι.

to the proof calculus. In some cases it suffices to switch from a pure sequent calculus to a calculus of *labelled*, *hyper-* or *nested* sequents (see Chapter 5 for detail). When the fixed point logic is given by adding only inductively and co-inductively defined operators to a robust logic such as K, it is possible to obtain an analytic calculus by embedding induction into the *structure* of proofs. Proofs typically become infinite objects, either as well-founded trees over rules that may have infinitely many premises or as finitely branching *illfounded* trees. In the latter class, soundness of proofs is ensured by a syntactic correctness condition on admissible infinite branches.

This chapter derives interpolation from illfounded proofs for the modal μ-calculus and the various fixed point logics it encompasses. The expressive 'simplicity' of modal logic ensures that the focus can remain on *finite* proof objects, specifically, the illfounded proofs that are represented as tree unravellings of finite graphs, known as the *cyclic proofs*. In this representation, the correctness condition requires that every infinite path through the cyclic proof fulfils a decidable property known as the *global trace condition*. The cyclic proofs are sound and complete for the modal μ-calculus [24] and linear-time μ-calculus [11]. Analyticity of the inference rules means that the cyclic proofs are also sound and complete for any fragment that is determined by inductive and co-inductive (modal) operators.[2]

The construction of Craig, Lyndon and uniform interpolants from cyclic proofs proceeds in the usual fashion for analytic sequent calculi: candidate interpolants are assigned to each leaf of the proof-tree (in the case of uniform interpolation a proof-search-tree) and, recursively, to inner vertices. Unlike the purely finitary proofs, a cyclic proof can contain leaves that are not initial sequents and merely cycle back to 'earlier' points in the proof. These leaves, called 'buds', cannot be interpolated directly. Instead, buds are assigned placeholder 'interpolants' which are explicitly defined once the recursive construction has traversed the entire associated cycle. Correctness of the construction (that the candidate interpolant fulfils the requirements of a Craig/Lyndon/uniform interpolant) follows from a normal form theorem for cyclic proofs which ensures that the relative ordering of cycles in a proof internalises salient properties of the global trace condition.

As the definition and required properties of cyclic proofs for the modal μ-calculus is quite involved, it is introduced in two stages. First, we treat the fragment involving *only* inductive (or co-inductive) operators where the notion of cyclic proof is close to the naive one and the aforementioned normal form theorem is vacuous (Section 3). Lyndon and uniform interpolation are established for this fragment in Sections 4 and 5 respectively. Cyclic proofs for the full calculus are defined in Section 6 and followed by proofs of Lyndon and uniform interpolation in the general case. The chapter closes with a discussion of the wider applicability of the cyclic Maehara method to modal fixed point logics and other μ-calculi.

---

[2] Presumably, the result also holds for robust fragments of guarded first order fixed point logic (cf Chapter 2) though to the best of the authors' knowledge this has not been examined.

Beyond proofs of Lyndon and uniform interpolation, the chapter explores the complexity of constructed interpolants in terms of alternations between inductive and co-inductive operators. Several well-known theorems on the expressivity of the modal μ-calculus can be realised as corollaries of interpolation augmented by structural observations on the form of cyclic proofs for specific quantifier classes. The construction of Craig–Lyndon interpolants witnesses, for example, the collapse of the quantifier class Δ_1 to the modal fragment and the collapse of Δ_2 to the alternation-free fragment. Corollaries of the uniform interpolation theorem include Otto's theorem on the decidability of equivalence to the modal fragment [25], Küsters and Wilkie's corresponding theorem for the co-induction-free fragment [18] and the disjunctive normal form theorem of Janin and Walukiewicz [15].

This chapter does not contribute any new results. Uniform interpolation for the modal μ-calculus was established by D'Agostino and Hollenberg in 2000 via a semantic argument [10]. The argument of [10] is discussed in Chapter 9 and will not be rehearsed here. The approach of this chapter is a generalisation of the proof-theoretic verification of Lyndon interpolation and uniform Lyndon interpolation in, respectively, [3] and [4]. The first explicit use of cyclic proofs to establish interpolation is Shamkanov's proof of the Lyndon interpolation property for Gödel–Löb provability logic [29]. Predating most of the above results, however, are various separation theorems for the modal μ-calculus and fragments [25, 18, 28, 7] which employ tableaux or related syntactic methods.

2 The modal μ-calculus

The following basic notions are used throughout this work. The set of finite sequences over a set X is denoted $X^{<\omega}$ and assumed to be canonically ordered by the (strict) prefix relation $<$. A *tree* is any non-empty set $T \subset \omega^{<\omega}$ closed under the prefix relation. Elements of T are called *vertices* with the empty sequence referred to as the *root* of T. By $<_T$ and $\leq_T$ we denote the strict and weak prefix relations restricted to vertices of T. The set of *immediate successors (in T)* of a vertex u is denoted $\mathrm{Suc}_T(u)$. If $|\mathrm{Suc}_T(u)| = 1$, the single immediate successor of u will often be denoted u^+. A *branch* of T is a (finite or infinite) sequence $(u_i)_i$ of vertices such that u_0 is the root and $u_{i+1} \in \mathrm{Suc}_T(u_i)$ for every i. A tree *over (a set) X* is a tree T equipped with a *labelling function* $\lambda\colon T \to X$ over X. Every finite tree F is assigned a *height*, in symbols $h(F)$, defined in the expected way with the trivial tree having height 0.

2.1 Syntax

Fix disjoint sets $\mathrm{Var} = \{x, y, z, \ldots\}$ and $\mathrm{Cons} = \{a, b, c, \ldots\}$ of, respectively, *(propositional) variables* and *(propositional) constants*. The *formulas* of the modal μ-calculus are the expressions α generated by the grammar:

$$\text{Formulas: } \alpha ::= x \mid a \mid \neg a \mid \bigwedge\Gamma \mid \bigvee\Gamma \mid \Box\alpha \mid \Diamond\alpha \mid \nu x\,\alpha \mid \mu x\,\alpha$$
$$\text{Cedents: } \Gamma ::= \emptyset \mid \Gamma \cup \{\alpha\}.$$

That is, formulas are constructed from variables, constants and negated constants by arbitrary finite conjunctions and disjunctions, the (unary) modal operators $\Box$ and $\Diamond$ and two propositional quantifiers,[3] μ and ν. The set of formulas is denoted L_μ. Symbols α, β, etc. range over formulas, and Γ, Δ, Θ, etc. over finite sets of formulas, called *cedents*. A constant or negated constant is called a *literal* and the set of literals is denoted $\mathrm{Lit} = \mathrm{Cons} \cup \{\neg a \mid a \in \mathrm{Cons}\}$. Expressions l, l', etc. range over literals, and L, L' over cedents of literals. Variables are not literals and are never negated. Formulas $\mu x\, \alpha$ and $\nu x\, \alpha$ are called *quantified* and express, respectively, the *least* and *greatest* fixed points of the propositional function $\varphi_\alpha \coloneqq \lambda x\, \alpha$. Symbols σ, σ', etc., act as metavariables for the two quantifiers. A formula without quantifiers is called *quantifier-free*.

▶ **Definition 2.1** (Signature). *The* signature *of a formula/cedent is the set of freely occurring literals and variables in the formula/cedent, given by* $\mathrm{sig}(\alpha) = \{\alpha\}$ *for* $\alpha \in \mathrm{Lit} \cup \mathrm{Var}$, $\mathrm{sig}(\sigma x\, \alpha) = \mathrm{sig}(\alpha) \setminus \{x\}$, $\mathrm{sig}(\Box\alpha) = \mathrm{sig}(\Diamond\alpha) = \mathrm{sig}(\alpha)$, *and* $\mathrm{sig}(\bigwedge\Gamma) = \mathrm{sig}(\bigvee\Gamma) = \mathrm{sig}(\Gamma) = \bigcup_{\gamma\in\Gamma} \mathrm{sig}(\gamma)$. *A formula* α *is* closed *if* $\mathrm{sig}(\alpha) \cap \mathrm{Var} = \emptyset$.

We introduce abbreviations for commonly used formulas and sets:

$$\top \coloneqq \bigwedge\emptyset \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \coloneqq \bigwedge\{\alpha_1, \ldots, \alpha_n\} \qquad \Box\Gamma \coloneqq \{\Box\gamma \mid \gamma \in \Gamma\}$$
$$\bot \coloneqq \bigvee\emptyset \qquad \alpha_1 \vee \cdots \vee \alpha_n \coloneqq \bigvee\{\alpha_1, \ldots, \alpha_n\} \qquad \Diamond\Gamma \coloneqq \{\Diamond\gamma \mid \gamma \in \Gamma\}$$

Substitution of formulas for variables is expressed by notation $\alpha(\beta/x)$, meaning the result of replacing each free occurrence of x in α by the formula β and renaming bound variables in α to avoid variable capture. As variables do not occur in literals, $\alpha(\beta/x)$ is always a formula. Negation is implemented through logical duality by an operation $\alpha \mapsto \alpha^\perp \in \mathsf{L}_\mu$ defined below. The exception is variables which, having no negated form, are considered self-dual. This approach is motivated by the action of negation on quantified formulas, which should identify the formula $\mu x(\Diamond x \vee a)$ with its 'complement' $\nu x(\Box x \wedge \neg a)$.

$$a^\perp \coloneqq \neg a \quad \big(\textstyle\bigwedge\Gamma\big)^\perp \coloneqq \bigvee \Gamma^\perp \quad (\Box\alpha)^\perp \coloneqq \Diamond\alpha^\perp \quad (\nu x\, \alpha)^\perp \coloneqq \mu x\, \alpha^\perp \quad \Gamma^\perp \coloneqq \{\gamma^\perp \mid \gamma \in \Gamma\}$$
$$\neg a^\perp \coloneqq a \quad \big(\textstyle\bigvee\Gamma\big)^\perp \coloneqq \bigwedge \Gamma^\perp \quad (\Diamond\alpha)^\perp \coloneqq \Box\alpha^\perp \quad (\mu x\, \alpha)^\perp \coloneqq \nu x\, \alpha^\perp \quad x^\perp \coloneqq x$$

There are two notions of subformula pertinent to the intension of quantified formulas. The *direct subformulas* of α are the formulas encountered during the construction of α in the generating grammar. So α is a direct subformula of $\Box\alpha$ and $\Diamond\alpha$, of $\bigwedge\Gamma$ and $\bigvee\Gamma$ if $\alpha \in \Gamma$, and of $\mu x\alpha$ and $\nu x\alpha$ for all $x \in \mathrm{Var}$. Literals and variables have no direct subformulas. The *complexity* of a formula α, in symbols $|\alpha|$, is the length of the longest chain of direct subformulas starting from α.

[3] The term 'quantifier' refers to the role of the operators in Tarskian semantics in which they are interpreted as quantifiers ranging over truth values; cf Proposition 2.11.

A second notion of subformula is more relevant to the proof systems and semantics utilised in this chapter, where the provability or validity of a quantified formula $\sigma x\alpha$ is made contingent on the provability/validity of the fixed point 'unfolding' $\alpha(\sigma x\alpha/x)$. This notion is called the Fischer–Ladner subformula relation, or *FL-subformula*, after its introduction in [13]. Unlike the relation of direct subformula, the FL-subformula relation is not well-founded: $\alpha = \mu x\,\Box x$ is a proper FL-subformula of $\Box\alpha$, which is a proper FL-subformula of α. A formula α is an *immediate FL-subformula* of β iff either

1. $\beta = \triangle\alpha$ for $\triangle \in \{\Box, \Diamond\}$,
2. $\beta = \bigcirc\Gamma$ for $\bigcirc \in \{\bigwedge, \bigvee\}$ and $\alpha \in \Gamma$, or
3. $\beta = \sigma x\gamma$ and $\alpha = \gamma(\beta/x)$ for some $\sigma \in \{\mu, \nu\}$, $x \in \mathrm{Var}$ and γ.

The relation of *FL-subformula* is the reflexive and transitive closure of the *immediate FL-subformula*. The set of FL-subformulas of α is denoted $\mathsf{FL}(\alpha)$. An induction on the complexity of α confirms that $\mathsf{FL}(\alpha)$ is finite.

▶ **Proposition 2.2.** *$\mathsf{FL}(\alpha)$ is bounded in cardinality by the number of direct subformulas of α.*

A *thread* is a finite or infinite sequence of formulas weakly following the FL-subformula relation, that is, a sequence $t = (\alpha_i)_{i<K}$ for $K \leq \omega$ such that for all $i + 1 < K$, either $\alpha_{i+1} = \alpha_i$ or α_{i+1} is an immediate FL-subformula of α_i. In the second case, the thread is said to *progress at i*. Of the infinite threads, we are primarily interested in those that progress infinitely often. These we call *progressing* threads.

The following present some important properties of threads.

▶ **Proposition 2.3.** $\mathrm{sig}(\alpha) = \mathsf{FL}(\alpha) \cap (\mathrm{Var} \cup \mathrm{Lit})$.

▶ **Proposition 2.4.** *If β is an FL-subformula of α then $\mathsf{FL}(\beta) \subset \mathsf{FL}(\alpha)$ and $\mathrm{sig}(\beta) \subset \mathrm{sig}(\alpha)$.*

▶ **Proposition 2.5.** *Let $(\alpha_i)_{i\leq k}$ be a finite thread and suppose $|\alpha_k| < |\alpha_i|$ for all $i < k$. Then there exists a thread $(\beta_i)_{i\leq k}$ such that $\beta_k \in \mathrm{Var}$ and $\alpha_i = \beta_i(\alpha_k/\beta_k)$ for all $i \leq k$.*

Proof. By induction on k. Let $(\alpha_i)_{i\leq k}$ be a finite thread and $|\alpha_k| < |\alpha_i|$ for all $i < k$. Let $k > 0$ and suppose $\alpha_0 = \sigma y\,\alpha'$. The induction hypothesis applied to the thread $(\alpha_1, \ldots, \alpha_k)$ yields a corresponding thread $(\beta_1, \ldots, \beta_k)$ such that $\alpha_i = \beta_i(\alpha_k/x)$ for all $1 \leq i \leq k$ where $x = \beta_k$. In particular, $\alpha'(\alpha_0/y) = \beta_1(\alpha_k/x)$. As $|\alpha_0| > |\alpha_k|$, there is a formula β' such that $\beta_1 = \beta'(\sigma y\,\beta'/y)$ and $\alpha' = \beta'(\alpha_k/x)$. Setting $\beta_0 = \sigma y\,\beta'$ completes the argument. The other cases of α_0 are straightforward. ◀

▶ **Proposition 2.6.** *Let $(\alpha_i)_{i<\omega}$ be a progressing thread. There is a unique formula α satisfying*

1. *$\alpha = \alpha_i$ for infinitely many i,*
2. *there exists $k < \omega$ such that $|\alpha| \leq |\alpha_i|$ for all $i > k$.*

Moreover, α is quantified, namely $\alpha = \sigma x\beta$ for some $\sigma \in \{\mu, \nu\}$, $x \in \mathrm{Var}$ and β.

▶ **Definition 2.7** (μ-/ν-thread)**.** *Let t be a progressing thread and $\alpha = \sigma x\beta$ the formula identified by Proposition 2.6. If $\sigma = \mu$ we refer to t as a μ-thread, otherwise as a ν-thread.*

2.2 Semantics

Formulas of the modal μ-calculus are evaluated against *pointed frames* (shortened to *frames*), tuples $F = \langle W, E, r, \lambda\rangle$ where $\langle W, E\rangle$ is a countable directed graph over a non-empty set W, $r \in W$ and $\lambda\colon W \to 2^{\mathrm{Cons}}$ is a function mapping a set of constants to each element of W. We call W the *carrier*, elements of W *worlds* and λ the *assignment*. The distinguished world r is called the *root* of F. Frames will be identified with their carriers, writing $u \in F$ to express that u is a world of F, and the assignment of F denoted λ_F. For a world $u \in F$, let $\langle F, u\rangle$ be the pointed frame $\langle W, E, u, \lambda\rangle$ with root u. For $u \in F$, define $F(u) = \{v \in W \mid (u, v) \in E\}$.

There are two (equivalent) semantics for the modal μ-calculus. One is a lifting of Tarski–Kripke semantics for modal logic to the quantifiers of L_μ. The other is to view satisfiability of L_μ formulas as an infinite game between two players, called the *model-checking game*. We will employ the latter as it aligns most closely to the realm of cyclic proofs; for more on either choice of semantics we refer the reader to [9, 8].

In game-semantics, formulas are conceived as specifying the rules of a game between two players, *Verifier* and *Refuter*, whose aim is to verify/refute claims of the kind 'F validates α'. In the case that $\alpha = \bigvee\Gamma$, for example, the rules stipulate that it is Verifier's turn and that her move is to select as the next position in the game a statement 'F validates γ' for some $\gamma \in \Gamma$. If Verifier is able to win the game from one such position, she will have confirmed that F is a model of γ and, therefore, that F is a model of α. Refuter's aim is to win the game regardless of the move Verifier makes.

Formally, the model checking game is represented as a directed graph whose vertices express the positions in the game. For a fixed closed formula α and pointed frame $\langle F, r\rangle$, a position is a pair (γ, v) for $\gamma \in \mathsf{FL}(\alpha)$ and $v \in F$. Each position is assigned an owner – Verifier or Refuter – and a set of available next moves, called *continuations*. From the initial position (α, r), a play of the game unfolds by the player owning the current position selecting the next from the set of continuations. A player who cannot make a move when required loses. For example, Verifier immediately wins in the position $(\bigwedge\emptyset, v)$ since there are no continuations and Refuter has ownership of the position.

Ownership of positions and admissible continuations are detailed in Table 1. The continuations from $(\Diamond\gamma, v)$ and $(\Box\gamma, v)$ are the positions (γ, w) for $w \in F(v)$. Verifier owns $(\Diamond\gamma, v)$ and Refuter owns $(\Box\gamma, v)$. For a formula $\alpha = \sigma x\beta$ there is just one continuation which is to move to its unfolding $\beta(\alpha/x)$. As the formula may increase in complexity, plays in the model-checking game can be infinite. The winning player of an infinite play is determined by the quantifiers which are unfolded infinitely often during the course of the play.

▶ **Definition 2.8** (Model checking game). *Let α be a closed formula and F a frame. The* model checking game *for (α, F) is the directed graph $G(\alpha, F)$ over vertices $\mathsf{FL}(\alpha) \times F$ with an edge from p to p' iff p' is a continuation of p per Table 1. Each position is assigned an* owner*, either* Verifier *or* Refuter*, detailed in the table.*

A play *of the model checking game $G(\alpha, F)$ is a maximal path through the graph starting from the position (α, r) where r is the root of F.*

Table 1 The model-checking game $G(\alpha, F)$.

Positions owned by Verifier		Positions owned by Refuter	
Position	Continuations	Position	Continuations
(a, v) for $a \notin \lambda_F(v)$	none	(a, v) for $a \in \lambda_F(v)$	none
$(\neg a, v)$ for $a \in \lambda_F(v)$	none	$(\neg a, v)$ for $a \notin \lambda_F(v)$	none
$(\bigvee\Gamma, v)$	$\{(\gamma, v) \mid \gamma \in \Gamma\}$	$(\bigwedge\Gamma, v)$	$\{(\gamma, v) \mid \gamma \in \Gamma\}$
$(\Diamond\alpha, v)$	$\{(\alpha, w) \mid w \in F(v)\}$	$(\Box\alpha, v)$	$\{(\alpha, w) \mid w \in F(v)\}$
$(\mu x\alpha, v)$	$\{(\alpha(\mu x\alpha/x), v)\}$	$(\nu x\alpha, v)$	$\{(\alpha(\nu x\alpha/x), v)\}$

A finite play must end in a position that has no continuations, namely positions (l, v) for $l \in \mathrm{Lit}$, $(\bigwedge\emptyset, v)$ and $(\bigvee\emptyset, v)$, as well as $(\Diamond\gamma, v)$ and $(\Box\gamma, v)$ if $F(v) = \emptyset$. In such cases we say that the play is *losing* for the player owning the final position and *winning* for the other player. An infinite play is an infinite sequence $p = (\alpha_i, v_i)_i$ of positions such that $(\alpha_i)_i$ is an infinite progressing thread. Definition 2.7 designates the thread as either μ or ν. Verifier is the winner of p iff the thread is ν.

The winning condition on plays is cofinal: for finite plays it depends only on the final position and for infinite plays is invariant under removing finite prefixes.

▶ **Definition 2.9** (Strategy). *A* strategy for Verifier (Refuter) *in $G(\alpha, \langle F, r\rangle)$ is a tree $S = \langle T, \rho\rangle$ over positions in the game with root labelled by (α, r) and such that for every $u \in T$,*

1. *if $\rho(u)$ is a position owned by Refuter (resp. Verifier) then every continuation from $\rho(u)$ is the label of a successor of u in T.*
2. *if $\rho(u)$ is a position owned by Verifier (resp. Refuter) and the set of continuations from $\rho(u)$ is non-empty then u has a successor in T labelled by a continuation of $\rho(u)$.*

A play is consistent *with S if the play enumerates a path through S. A strategy for Verifier (Refuter) is* winning *if every consistent play is winning for Verifier (Refuter).*

For a closed formula α, we set $\langle\!|\alpha|\!\rangle$ to be the set of pointed frames F for which there exists a winning strategy for Verifier in $G(\alpha, F)$. This definition gives rise to the following properties showing that the game semantics agrees with the Tarski–Kripke semantics.

▶ **Proposition 2.10.** *The following hold for every $a \in \mathrm{Cons}$, $\alpha \in \mathsf{L}_\mu$, cedent Γ and quantifier σ.*

1. $\langle\!|a|\!\rangle = \{\langle F, r\rangle \mid a \in \lambda_F(r)\}$ *and* $\langle\!|\neg a|\!\rangle = \{\langle F, r\rangle \mid a \notin \lambda_F(r)\}$.
2. $\langle\!|\bigwedge\Gamma|\!\rangle = \bigcap_{\gamma\in\Gamma}\langle\!|\gamma|\!\rangle$ *and* $\langle\!|\bigvee\Gamma|\!\rangle = \bigcup_{\gamma\in\Gamma}\langle\!|\gamma|\!\rangle$.
3. $\langle\!|\Diamond\alpha|\!\rangle = \{\langle F, r\rangle \mid \exists u \in F(r)\ \langle F, u\rangle \in \langle\!|\alpha|\!\rangle\}$ *and* $\langle\!|\Box\alpha|\!\rangle = \{\langle F, r\rangle \mid \forall u \in F(r)\ \langle F, u\rangle \in \langle\!|\alpha|\!\rangle\}$.

All that is missing from the above is a characterisation of quantified formulas $\sigma x\, \alpha$. Let $\mathcal{F}$ be the set of frames. For a variable x and $U \subset \mathcal{F}$ of frames, let $\langle\!|\alpha|\!\rangle_{x\mapsto U}$ be the set of frames F for which Verifier has a winning strategy in the modified model-checking game $G(\alpha, F)$ where positions (x, u) are winning for Verifier iff $\langle F, u\rangle \in U$. The next result establishes the equivalence between game and Tarski–Kripke semantics.

▶ **Proposition 2.11.** *For every closed formula $\mu x\,\alpha$, $\langle\!|\mu x\,\alpha|\!\rangle = \bigcap\{U \subset \mathcal{F} \mid \langle\!|\alpha|\!\rangle_{x\mapsto U} \subset U\}$ and $\langle\!|\nu x\,\alpha|\!\rangle = \bigcup\{U \subset \mathcal{F} \mid U \subset \langle\!|\alpha|\!\rangle_{x\mapsto U}\}$.*

Proof sketch. Let S be a winning strategy for Verifier in $G(\mu x\,\alpha, F)$. Let $M \subseteq S$ be the set of vertices at which $\mu x\,\alpha$ occurs which, by assumption, is a well-founded subtree of S. Via transfinite induction on this tree it can be shown that if $\langle\!|\alpha|\!\rangle_{x\mapsto U} \subset U$ then $F \in U$. So $\langle\!|\mu x\,\alpha|\!\rangle \subset U$. As $\langle\!|\alpha|\!\rangle_{x\mapsto \langle\!|\mu x\,\alpha|\!\rangle} \subset \langle\!|\mu x\,\alpha|\!\rangle$, the first claim is complete. The second claim follows a dual argument: If $U \subset \langle\!|\alpha|\!\rangle_{x\mapsto U}$ then, by repeated application of the inclusion, a winning strategy in $G(\nu x\,\alpha, F)$ can be defined for every $F \in U$. ◀

In the model-checking game, negation acts as duality: exchanging ownership of all positions in $G(\alpha, F)$ is simply the game $G(\alpha^\perp, F)$ up to the naming of positions. Thus, winning strategies for Verifier in $G(\alpha, F)$ transfer to winning strategies for Refuter in $G(\alpha^\perp, F)$ and vice versa. The semantic properties above yield determinacy of the model-checking game: in every instance $G(\alpha, F)$ (exactly) one of the two players has a winning strategy:

▶ **Corollary 2.12** (Determinacy). *For every closed α, $\langle\!|\alpha^\perp|\!\rangle = \mathcal{F} \setminus \langle\!|\alpha|\!\rangle$.*

▶ **Definition 2.13** (Validity, equivalence). *A closed formula α is* satisfiable *if $\langle\!|\alpha|\!\rangle \neq \emptyset$ and* unsatisfiable *otherwise. We call α* valid *if $\langle\!|\alpha|\!\rangle$ comprises all frames. If $\langle\!|\alpha|\!\rangle = \langle\!|\beta|\!\rangle$ we say that α and β are* equivalent *and write $\alpha \equiv \beta$.*

2.3 Important Fragments

A variable x is *guarded* in α if every occurrence of x in α is under the scope of a modal operator. A formula is *guarded* if for every direct subformula $\sigma x\,\alpha$ the variable x is guarded in α. Replacing 'direct' in the above by 'FL-' yields an equivalent definition. Another equivalent formulation can be given in terms of cyclic threads:

▶ **Proposition 2.14** (Guarded fragment). *A formula α_0 is guarded iff for every finite thread $(\alpha_i)_{i\leq k}$, if $\alpha_l = \alpha_k$ for some $l < k$ and the thread progresses between l and k, then for some $j \in [l, k)$ and $\triangle \in \{\Box, \Diamond\}$ it is the case that $\alpha_j = \triangle\alpha_{j+1}$.*

We define two closure operations on sets of formulas.

▶ **Definition 2.15** (Closure operation). *For a set of formulas F, define $Clos(F)$ to be the least set $C \subset \mathsf{L}_\mu$ satisfying*

1. *$F \subset C$,*
2. *If $\Gamma \subset C$ then $\{\bigwedge\Gamma, \bigvee\Gamma\} \subset C$,*
3. *If $\alpha \in C$ then $\{\Box\alpha, \Diamond\alpha\} \subset C$,*
4. *If $\alpha, \beta \in C$ and $x \in \mathrm{Var}$ then $\alpha(\beta/x) \in C$.*

For $\sigma \in \{\mu, \nu\}$ define $Clos^\sigma(F)$ to be the least set $C \subset \mathsf{L}_\mu$ satisfying above plus:

5. *If $\alpha \in C$ then $\sigma x\alpha \in C$.*

Note, in particular, that $Clos(F)$ (as well as $Clos^\sigma(F)$) is closed under substitution. So, for instance, $\mu x(x \vee \nu y(a \wedge y)) \in Clos(\{\mu x(x \vee z), \nu y(a \wedge y)\}) \subset Clos^\mu(\{x \vee z, \nu y(a \wedge y)\})$, whereas $\mu x(x \vee \nu y(x \wedge y)) \in Clos^\mu(\{x \vee z, \nu y(x \wedge y)\}) \setminus Clos(\{\mu x(x \vee z), \nu y(x \wedge y)\})$.

The closure operation determines a natural quantifier hierarchy of formulas.

▶ **Definition 2.16** (Niwiński quantifier hierarchy). *The sets $\Pi_n, \Sigma_n \subset \mathsf{L}_\mu$ are defined by mutual recursion:*

- $\Sigma_0 = \Pi_0$ *is the set of quantifier-free formulas,*
- $\Sigma_{n+1} = Clos^\mu(\Pi_n)$,
- $\Pi_{n+1} = Clos^\nu(\Sigma_n)$.

Formulas in $Clos(\Sigma_1 \cup \Pi_1)$ are called alternation-free.

▶ **Proposition 2.17** ([17]). *Every formula is equivalent to a guarded formula in the same quantifier classes.*

The winning condition of the model-checking game is trivial for Σ_1 and Π_1 formulas:

▶ **Lemma 2.18.** *Verifier wins no infinite plays in $G(\alpha, F)$ if $\alpha \in \Sigma_1$ and wins all infinite plays if $\alpha \in \Pi_1$.*

The following characterisation is straightforward.

▶ **Lemma 2.19.** $\Sigma_{n+1} \cap \Pi_{n+1} = Clos(\Sigma_n \cup \Pi_n)$.

A formula in $\Sigma_{n+1} \setminus Clos(\Sigma_n \cup \Pi_n)$ is called *genuine* Σ_{n+1}. Similarly, a formula in $\Pi_{n+1} \setminus Clos(\Sigma_n \cup \Pi_n)$ is *genuine* Π_{n+1}.

▶ **Definition 2.20** (Ambiguous classes). *For a set of formulas F, define $F^{\equiv}$ to be the set of formulas equivalent to some formula in F. The* ambiguous class at level n *is the set* $\Delta_n = \Sigma_n^{\equiv} \cap \Pi_n^{\equiv}$.

The counting of the ambiguous classes conventionally begins at the first level, so Δ_1 is called the *first* ambiguous class, Δ_2 the *second* and so on. From Lemma 2.19 we have that $Clos(\Sigma_n \cap \Pi_n)^{\equiv} \subset \Delta_{n+1}$ for all n. The two sets turn out to be identical for $n \in \{0, 1\}$. If α is equivalent to both a Σ_1 and Π_1 formula then α is equivalent to a quantifier-free formula, and if α is equivalent to formulas in Σ_2 and Π_2, then α is equivalent to a formula in the intersection $\Sigma_2 \cap \Pi_2$ [7]. Both results are corollaries of Craig interpolation, proven in Sections 4 (for $n = 0$) and 6 (for $n = 1$). A seminal result in the modal μ-calculus, established by Santocanale and Arnold [28], is that the comparison is strict at all other levels: for every $n > 2$ there exists a formula in Δ_n which is not equivalent to any formula in $(\Sigma_n \cap \Pi_n)^{\equiv}$.

$$\frac{\Theta, \Gamma \Rightarrow \Delta}{\bigwedge\Theta, \Gamma \Rightarrow \Delta}\bigwedge\mathsf{L} \qquad \frac{\alpha, \Gamma \Rightarrow \Delta \quad \text{for all } \alpha \in \Theta}{\bigvee\Theta, \Gamma \Rightarrow \Delta}\bigvee\mathsf{L} \qquad \frac{\alpha(\sigma x\, \alpha/x), \Gamma \Rightarrow \Delta}{\sigma x\, \alpha, \Gamma \Rightarrow \Delta}\sigma\mathsf{L}$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \text{for all } \alpha \in \Theta}{\Gamma \Rightarrow \Delta, \bigwedge\Theta}\bigwedge\mathsf{R} \qquad \frac{\Gamma \Rightarrow \Delta, \Theta}{\Gamma \Rightarrow \Delta, \bigvee\Theta}\bigvee\mathsf{R} \qquad \frac{\Gamma \Rightarrow \Delta, \alpha(\sigma x\, \alpha/x)}{\Gamma \Rightarrow \Delta, \sigma x\, \alpha}\sigma\mathsf{R}$$

$$\frac{\alpha, \Gamma \Rightarrow \Delta}{\Gamma', \Diamond\alpha, \Box\Gamma \Rightarrow \Diamond\Delta, \Delta'}\Diamond\mathsf{L} \qquad \frac{\Gamma \Rightarrow \Delta, \alpha}{\Gamma', \Box\Gamma \Rightarrow \Diamond\Delta, \Box\alpha, \Delta'}\Box\mathsf{R}$$

$$\frac{}{\Gamma, l \Rightarrow l, \Delta}\mathsf{id} \qquad \frac{}{\Gamma, a, \neg a \Rightarrow \Delta}\mathsf{idL} \qquad \frac{}{\Gamma \Rightarrow \Delta, a, \neg a}\mathsf{idR}$$

$$\frac{}{\Gamma \Rightarrow \Delta}\mathsf{b} \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\mathsf{c}$$

Figure 1 Basic rules of inference. Rules denoted $\star\mathsf{L}/\star\mathsf{R}$ are called *left*/*right* rules respectively. Rules b and c are *structural* rules; others are *logical* rules. The rules $\Box\mathsf{R}$ and $\Diamond\mathsf{L}$ are, collectively, the *modal* rules. Instances of b and c are called *buds* and *companions* respectively.

3 Cyclic Derivations

We introduce a sequent calculus for the modal μ-calculus. The calculus derives expressions $\Gamma \Rightarrow \Delta$, called *sequents*, where Γ and Δ are finite sets of closed and guarded formulas, henceforth referred to as *cedents*. We refer to Γ as the *left* cedent of $\Gamma \Rightarrow \Delta$ and to Δ as the *right* cedent. In the context of sequents, formulas are identified with singleton sets and union is expressed by comma, so $\Gamma, \Delta = \Gamma \cup \Delta$ and $\Gamma, \alpha = \Gamma, \{\alpha\}$.

The sequent $\Gamma \Rightarrow \Delta$ is *valid* if $(\!|\bigwedge\Gamma|\!) \subset (\!|\bigvee\Delta|\!)$. That is, for every frame $\langle F, r\rangle$, if Verifier has a winning strategy from (γ, r) for each $\gamma \in \Gamma$, then she has a winning strategy from (δ, r) for some $\delta \in \Delta$. Corollary 2.12 implies that the sequents $\Gamma \Rightarrow \Delta$ and $\Gamma, \Delta^{\perp} \Rightarrow$ are equi-valid: one is valid iff both are.

Figure 1 presents the rules of our sequent calculus, named $\mathsf{K}\mu$. Dropping the structural and quantifier rules yields a natural sound and complete sequent calculus for basic modal logic K. A sequent which has been derived via the rule b will be called a *bud*. In the presence of b the calculus is clearly unsound, but incomplete without the rule. A sound and complete calculus for sequents in $\mathsf{K}\mu$ will be obtained by carefully constraining, in a purely syntactic manner, how buds may be utilised within a proof. In other words, the calculus of rules from Figure 1 should not be conceived of as a *proof* system per se, but as a system of *derivations* of which some will constitute *proofs*.

Before presenting the calculus in detail, we fix some basic concepts and notation.

▶ **Definition 3.1** (The sequent calculus $\mathsf{K}\mu$). *The sequent calculus* $\mathsf{K}\mu$ *is specified by the rules in Figure 1. For each rule, the sequent below the line is the* conclusion *and the sequents above are the* premises. *An instance of the structural rule* b *or* c *is called, respectively, a* bud *or* companion.

$$\dfrac{\dfrac{\dfrac{}{\Rightarrow \alpha, \beta, \alpha \vee \beta}\,\mathsf{b}}{\Rightarrow \alpha, \beta, \alpha \vee \beta}\,\bigvee\mathsf{R}}{\Rightarrow \alpha, \beta, \alpha \vee \beta}\,\mathsf{c}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a, \alpha \Rightarrow a}\,\mathsf{id}}{a \wedge \alpha \Rightarrow a}\,\bigwedge\mathsf{L}}{b, \Box(a \wedge \alpha) \Rightarrow \Box a, \Diamond\beta}\,\Box\mathsf{R} \quad \dfrac{\dfrac{\dfrac{}{b, \alpha \Rightarrow \beta}\,\mathsf{b}}{b \wedge \alpha \Rightarrow \beta}\,\bigwedge\mathsf{L}}{b, \Diamond(b \wedge \alpha) \Rightarrow \Box a, \Diamond\beta}\,\Diamond\mathsf{L}}{b, \Box(a \wedge \alpha) \vee \Diamond(b \wedge \alpha) \Rightarrow \Box a, \Diamond\beta}\,\bigvee\mathsf{L}}{b, \Box(a \wedge \alpha) \vee \Diamond(b \wedge \alpha) \Rightarrow \Box a \vee \Diamond\beta}\,\bigvee\mathsf{R}}{b, \Box(a \wedge \alpha) \vee \Diamond(b \wedge \alpha) \Rightarrow \beta}\,\mu\mathsf{R}}{b, \alpha \Rightarrow \beta}\,\mu\mathsf{L}}{b, \alpha \Rightarrow \beta}\,\mathsf{c}$$

Figure 2 Two examples of derivations. In the right-hand derivation, $\alpha = \mu x.\, \Box(a \wedge x) \vee \Diamond(b \wedge x)$ and $\beta = \mu x(\Box a \vee \Diamond x)$. The left-hand derivation is not a cyclic derivation because of the lack of a modal rule. The right-hand derivation qualifies as a cyclic derivation and, moreover, as a cyclic proof.

▶ **Definition 3.2** (Derivation). *A* derivation *in a sequent calculus is a tree over sequents and rules of the calculus such that every vertex and its immediate successors are the conclusion and premises respectively of the associated rule. The sequent labelling the root vertex is the* end-sequent *of the derivation.*

Relevant to subsequent definitions is the implicit constraint that every vertex of a derivation is a correct application of a rule of inference. In particular, leaves must be assigned rules with zero premises, such as id, idL, idR, b, or the specific instances of $\bigvee\mathsf{L}$ and $\bigwedge\mathsf{R}$ introducing the formulas $\bot = \bigvee\emptyset$ and $\top = \bigwedge\emptyset$. Such rule instances are called *initial*. The rules b and c are collectively called *structural* rules; all others rules are *logical*. The rules $\Box\mathsf{R}$ and $\Diamond\mathsf{L}$ are *modal* rules, and the rules id, idL and idR are *identity* rules.

In each instance of a logical rule one or more formulas in the conclusion sequent are distinguished and called the *principal* formula(s). This is the distinguished formula $\bigwedge\Theta$ in the conclusion of $\bigwedge\mathsf{L}$ and $\bigwedge\mathsf{R}$, the formula $\sigma x\, \alpha$ in $\sigma\mathsf{L}$ and $\sigma\mathsf{R}$, the formula $\Diamond\alpha$ in the left cedent of $\Diamond\mathsf{L}$ and $\Box\alpha$ in the right cedent of $\Box\mathsf{R}$ and, finally, the marked literals in the three identity rules. The *active* formulas are the formulas in the premise(s) of each rule that correspond to the principal formula: all formulas in Θ for $\bigwedge\mathsf{L}$, the formula $\alpha(\sigma x\, \alpha/x)$ in the premise of $\sigma\mathsf{L}$, the formula α in the left cedent of the premise of $\Diamond\mathsf{L}$, etc. All other formulas in a rule are called *side formulas*. Given a vertex v of a derivation, a formula is *principal* at v if it is the principal formula of the rule at v, and is *active* if it is an active formula in the sense of the rule at the parent vertex (to which v is a premise).

Finite derivations without buds are sound, in the sense that their end-sequent must be valid. The set of finite bud-free derivations is, however, incomplete as the rules of the calculus do not distinguish between the quantifiers μ and ν. If a formula α admits a bud-free derivation then so does the result of exchanging any ν for μ, and vice versa, in α. In particular, the valid sequent $\Rightarrow \nu x \Box x$ has no bud-free derivation.

A sound and complete calculus is obtained by permitting some (but not all) *infinite* derivations as proofs. The symmetry of μ and ν is broken by imposing a restriction on their occurrences along any infinite path. The valid formula $\nu x \Box x$ admits an infinite derivation by continually alternating applications of $\nu\mathsf{R}$ and $\Box\mathsf{R}$, and the invalid formula $\mu x \Box x$ has a derivation given by continually alternating $\mu\mathsf{R}$ and $\Box\mathsf{R}$. The infinitely many applications of the rule $\nu\mathsf{R}$ will qualify the former derivation as a 'proof' but not the latter.

In fact, for the modal μ-calculus (and many other modal/temporal logics), it is sufficient to treat only derivations which ultimately repeat themselves. These derivations will be called 'cyclic'. The special rules b and c provide a syntax for describing cyclic derivations. Given a derivation D, let $\mathrm{bud}(D)$ and $\mathrm{comp}(D)$ be the set of buds and, respectively, companions.

▶ **Definition 3.3** (Cyclic derivation)**.** *A* cyclic derivation *is a finite derivation D equipped with a function $\mathsf{c}\colon \mathrm{bud}(D) \to \mathrm{comp}(D)$ mapping each bud b to a companion $b^{\mathsf{c}} <_D b$ satisfying:*

1. *the sequents at b and b^{c} are identical,*
2. *there is an application of a modal rule on the path from b^{c} to b.*

We refer to b^{c} as the companion to *b.*

▶ **Example 3.4.** Figure 2 presents two derivations. The left-hand tree is a derivation because of the set-based perspective of sequents that allows the principal formula of $\bigvee\mathsf{R}$, namely $\alpha \vee \beta$, to occur also as a side formula. Although the single bud is repeating the root sequent and the latter is a companion, the tree is not a *cyclic* derivation because there is no modal rule between the (only) companion and bud. In contrast, the right-hand derivation in Figure 2 is a cyclic derivation: The companion to the single bud is also the root and there is a modal rule ($\Diamond\mathsf{L}$) between the two vertices. ⌟

Let D be a cyclic derivation. As the sequent at every bud in D is identical to that of its companion, any bud b of D could be extended by inserting a copy of the subderivation rooted at b^{c}. The result would be another cyclic derivation of the same sequent, which we call an *unfolding* of D. Repeatedly applying this operation so that every bud is eventually unfolded and removing superfluous companion rules, produces a bud-free derivation which we call the *unravelling* of D and denoted D_ω. If D has at least one bud, the unravelling of D will be an infinite derivation.

The branches of a cyclic derivation are all finite by assumption. We also want to consider sequences of vertices that describe branches of the *unravelling* of the cyclic derivation. We call these *paths* where a path through a cyclic derivation D is a finite or infinite sequence of vertices $(u_i)_{i<K}$ such that for every i, if $i+1 < K$ then either u_{i+1} is an immediate successor of u_i or u_i is a bud and $u_{i+1} = (u_i)^{\mathsf{c}}$ is the associated companion.

An important relation that we will make extensive use of later is a connectedness relation between companions taking into account the process of unravelling. We introduce a relation $\lhd_D \subset \mathrm{comp}(D) \times D$ defined by $u \lhd_D v$ iff $u \in \mathrm{comp}(D)$ is a companion and $u <_D v <_D w$ for some $w \in \mathrm{bud}(D)$ such that $u = w^{\mathsf{c}}$.

3.1 Cyclic Proofs at the First Level

Attending to sequents at the first level of the quantifier hierarchy, that is, $\Gamma \Rightarrow \Delta$ where $\Gamma \cup \Delta \subset \Sigma_1 \cup \Pi_1$, a simple syntactic constraint isolates a class of derivations that are sound and complete for the fragment. These are the derivations for which every bud is a repetition of a sequent on the same branch as the bud and, moreover, the bud contains either a Σ_1 formula in the left cedent or a Π_1 formula in the right cedent. The fact that a bud must be a repeated sequent means that a proof for this fragment can be conceived of as an infinite derivation where in place of the bud the derivation loops back to its repeat. Unsurprisingly, such derivations will be called *cyclic proofs*.

We treat the notion of cyclic proof for sequents at the first level in detail in this section. This will be sufficient to obtain various Lyndon and Uniform Interpolation theorems for sequents and formulas at the first quantifier level, covered in Sections 4 and 5. Sequents beyond the first level necessitate a more careful definition of proof to ensure both soundness and completeness. The same concept of a cyclic derivation will be used but the condition on when a cyclic derivation is a *proof* is no longer localised to a simple property on each bud. Rather, we must consider the various infinite paths through the cyclic derivation and the formula threads that each path carries. A definition of proofhood appropriate for the second level and higher is the topic of Section 6.

For sequents at the first level of the quantifier hierarchy, the sound derivations can be readily carved out based on the formulas featured in buds and companions.

▶ **Definition 3.5** (Cyclic proofs at the first level)**.** *Let $\Gamma \Rightarrow \Delta$ be a sequent at the first level and D a cyclic derivation of $\Gamma \Rightarrow \Delta$. Let $\Gamma_u \Rightarrow \Delta_u$ denote the sequent at vertex $u \in D$. A bud u is:*
Left successful *if Γ_u contains a genuine Σ_1 formula.*
Right successful *if Δ_u contains a genuine Π_1 formula.*
Successful *if u is left or right successful.*
A cyclic proof *is a finite cyclic derivation D for which every bud is successful. If there exists a cyclic proof of $\Gamma \Rightarrow \Delta$, we write $\vdash \Gamma \Rightarrow \Delta$.*

▶ **Example 3.6.** The right-hand derivation in Figure 2 is a cyclic proof. The end-sequent is in $\Pi_1 \cup \Sigma_1$. As the single bud contains a genuine Σ_1 formula in the left cedent (α), it is successful. As the left-hand derivation in that figure is not a cyclic derivation, neither is it a proof. Figure 3 presents a cyclic derivation demonstrating the necessity of requiring that the Π_1/Σ_1 formula in the bud is *genuine*. ⌟

Duality of the sequent calculus yields

▶ **Proposition 3.7.** *$\vdash \Gamma \Rightarrow \Delta$ iff $\vdash \Rightarrow \Gamma^{\perp}, \Delta$, iff $\vdash \Gamma, \Delta^{\perp} \Rightarrow$.*

Weakening is the rule that allows deriving sequents $\Gamma', \Gamma \Rightarrow \Delta, \Delta'$ from $\Gamma \Rightarrow \Delta$. We denote the rule by w. The sequent calculus $\mathsf{K}\mu$ implicitly incorporates weakening at initial sequents and in applications of the modal rules but it does not allow uses of the rule more generally.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Rightarrow \Box\bot, \Box\mu x(\Box\bot \vee \Box x)}\,\mathsf{b}}{\Rightarrow \Box\bot \vee \Box\mu x(\Box\bot \vee \Box x)}\,\bigvee\mathsf{R}}{\Rightarrow \mu x(\Box\bot \vee \Box x)}\,\mu\mathsf{R}}{\Rightarrow \Box\bot, \Box\mu x(\Box\bot \vee \Box x)}\,\Box\mathsf{R}}{\Rightarrow \Box\bot, \Box\mu x(\Box\bot \vee \Box x)}\,\mathsf{c}$$

Figure 3 A cyclic derivation of an invalid sequent. The single bud of the derivation contains a Π_1 formula ($\Box\bot$) but no *genuine* Π_1 formula.

On occasion it is convenient to assume that the rule is explicitly included in the calculus. Indeed, the addition of the rule to $\mathsf{K}\mu$, while keeping the definition of cyclic derivation/proof otherwise unchanged, proves the same sequents as $\mathsf{K}\mu$ itself. The proof of this fact proceeds by 'pushing' each application of weakening upwards through the cyclic derivation (permuting with the logical rules) until it is absorbed by a modal or identity rule. A subtlety arises in uses of weakening below buds or companions. These are treated by unfolding the derivation once and continuing the process. We leave the full proof as an exercise.

▶ **Proposition 3.8.** *A sequent admits a cyclic proof in* $\mathsf{K}\mu + \mathsf{w}$ *iff it does so in* $\mathsf{K}\mu$.

We prove soundness and completeness for this notion of proof with respect to the game semantics of Section 2. The proof of soundness, that conclusions of cyclic proofs are valid, proceeds by relating strategies in the model-checking game to paths through a cyclic derivation and the success condition on buds to winning conditions.

▶ **Proposition 3.9** (Soundness). *For* $\Gamma \cup \Delta \subset \Sigma_1 \cup \Pi_1$, *if* $\vdash \Gamma \Rightarrow \Delta$ *then* $\Gamma \Rightarrow \Delta$ *is valid.*

Proof. Determinacy (Corollary 2.12) and duality (Proposition 3.7) imply that it suffices to treat the case that $\Delta = \emptyset$. Let $\Gamma \subset \Sigma_1 \cup \Pi_1$ and let D be a cyclic derivation of $\Gamma \Rightarrow$. Let F be any frame. We show that if there exists a winning Verifier strategy in the model checking game $G(\bigwedge\Gamma, F)$, then D is not a proof in the sense of Definition 3.5. Let S be any Verifier strategy in this game. We call a set $X \subset S$ a *bar* if no two elements of X lie on the same path through S.

Under the assumption that S is a winning strategy, we show it is possible to define a path $(u_i)_i$ through D, a sequence of finite bars $(X_i)_i$ and a sequence of worlds $(w_i)_i$ in F satisfying

1. The set of positions in X_i is $\{(\gamma, w_i) \mid \gamma \in \Gamma_i\}$ where $\Gamma_i \Rightarrow$ is the sequent at u_i.
2. Every position in X_{i+1} is either in X_i or is an available move from a position in X_i.
3. For every $i > 0$ there exists $j > i$ such that $X_i \cap X_j = \emptyset$.

The root of D is associated the bar X_0 of vertices of S immediately above the root and w_0 is the single world assigned to these positions. As the root of S is the position $(\bigwedge\Gamma, w_0)$, the first two conditions are met. Note that property 3 is a consequence of the first two since every path through D contains infinitely many vertices labelled by modal rules.

$$L \subset \mathrm{Lit}\ \frac{\gamma_1, \Delta \Rightarrow \quad \cdots \quad \gamma_n, \Delta \Rightarrow \quad \Delta \Rightarrow}{\Diamond\gamma_1, \ldots, \Diamond\gamma_n, \Box\Delta, L \Rightarrow}\ \Diamond^*\mathsf{L}$$

Figure 4 The 'branching' modality rule ($\Diamond^*\mathsf{L}$) used to obtain templates. Applications of this rule in the calculus are subject to the constraint that no other rule of the calculus has the same conclusion. In particular, L is a consistent set of literals. The rightmost premise of $\Diamond^*\mathsf{L}$ is the *minor* premise; other premises are *major*.

Suppose the finite sequences $(u_i)_{i\leq k}$, $(X_i)_{i\leq k}$ and $(w_i)_{i\leq k}$ have been defined. If u_k is a companion, set u_{k+1} to be the premise; if u_k is a bud, u_{k+1} is the associated companion. In either case, $X_{k+1} = X_k$ and $w_{k+1} = w_k$.

If the rule at u_k is $\bigvee\mathsf{L}$ with principal formula $\bigvee\Delta$ then there is a position $(\bigvee\Delta, w_k)$ in the bar X_k. Since S is a winning, $\Delta \neq \emptyset$ and S has a position (δ, w_k) for some $\delta \in \Delta$ labelling the immediate successor of a vertex in X_k. Select u_{k+1} to be the successor of u_k whose active formula is δ, let $w_{k+1} = w_k$ and X_{k+1} be the bar exchanging the position $(\bigvee\Delta, w_k)$ for (δ, w_k) in X_k. The other non-modal rules are similar. Note, u_k cannot be an identity rule because in that case there will be positions (a, w_k) and $(\neg a, w_k)$ in S in contradiction with S winning.

Suppose the rule at u_k is a modal rule, namely $\Diamond\mathsf{L}$, with principal formula $\Diamond\gamma$. Let $\gamma, \Delta \Rightarrow$ be the sequent of the premise, so $\Box\Delta$ is a subset of the left cedent at u_k. By assumption, the positions $(\Diamond\gamma, w_k)$ and $(\Box\delta, w_k)$ for each $\delta \in \Delta$ label vertices in X_k. From the first, and the assumption that S is winning, we find a world w_{k+1} such that (γ, w_{k+1}) is the next position in S after $(\Diamond\gamma, w_k)$. We also find the positions (δ, w_{k+1}) among the immediate successors of the vertices in X_k. Setting X_{k+1} to be the corresponding vertices in S, satisfies the two conditions.

Let $X = \bigcup_i X_i$. This is a finitely branching subtree of S. Condition 3 provides that X is infinite and so X has an infinite branch by König's Lemma. Now consider a bud b of D which occurs infinitely often in the path $(u_i)_i$. Each formula at b necessarily appears infinitely often along some infinite path of X. As S is winning for Verifier, these formulas must all be Π_1, meaning that b is not successful. ◀

3.2 Proof Search

Completeness of cyclic proofs is shown via a proof-search argument: Given a sequent we define a cyclic derivation within a variant of $\mathsf{K}\mu$ and show that the derivation either contains a cyclic proof of the sequent or encodes a witness of its invalidity.

The proof-search calculus comprises the structural and left rules of $\mathsf{K}\mu$ with a branching modal rule in place of $\Diamond\mathsf{L}$. The special modal rule, $\Diamond^*\mathsf{L}$, is presented in Figure 4. A set of L of literals is *inconsistent* if $\{a, \neg a\} \subset L$ for some constant a, and is *consistent* otherwise. The conclusion of the branching modal rule must be a sequent of the form $\Diamond\Gamma, \Box\Delta, L \Rightarrow$ where $L \subset \mathrm{Lit}$ is a consistent set of literals. It has as a premise the sequent $\gamma, \Delta \Rightarrow$ for every $\gamma \in \Gamma$, called the *major* premises, and a final premise with the sequent $\Delta \Rightarrow$ alone, called the *minor*

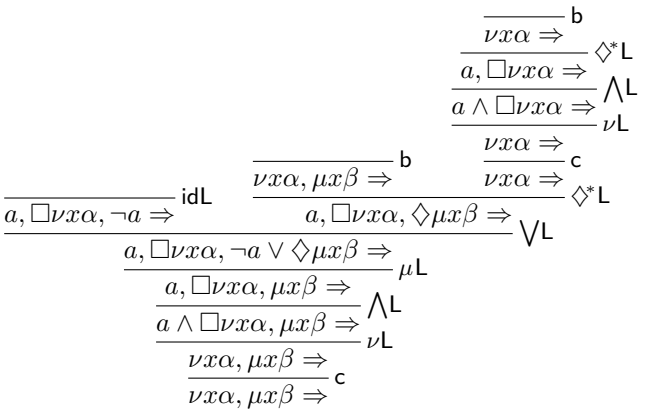


Figure 5 A template for $\{\nu x\,\alpha, \mu x\,\beta\}$ where $\alpha = a \wedge \Box x$ and $\beta = \alpha^{\perp}$. The central bud is successful as it contains a genuine Σ_1 formula, whereas the rightmost bud is unsuccessful. Pruning the lower instance of $\Diamond^*\mathsf{L}$ to its (only) major premise witnesses provability of the endsequent. Any other ordering of the logical rules results in the same sequents at modal rules.

premise. The minor premise plays no role in the proof-search beyond ensuring that instances of modal rules always contain at least one premise, relevant if $\Gamma = \emptyset$. Note, however, that if $\Delta = \emptyset$ then the minor premise will be the empty sequent $\Rightarrow$, which can be 'derived' via a degenerate application of $\Diamond^*\mathsf{L}$ with both conclusion and (single) premise empty.

We call the sequent calculus described above the *template calculus* and a (finite) cyclic derivation in the calculus a *template*. Figure 5 presents an example of a template.

▶ **Definition 3.10** (Template calculus). *The* template calculus *is the sequent calculus comprising the structural rules and logical left rules of* $\mathsf{K}\mu$*, with the modal rule* $\Diamond\mathsf{L}$ *replaced by the rule* $\Diamond^*\mathsf{L}$ *in Figure 4.*

A template *is a cyclic derivation* T *in this calculus such that between each bud and its associated companion there is an instance of the modal rule* $\Diamond^*\mathsf{L}$*. If the end sequent of* T *is* $\Gamma \Rightarrow$*, we refer to* T *as a* template for Γ *or an* α-template *if* $\Gamma = \{\alpha\}$.

Let T be a derivation in the template calculus. A *pruning* of T is a subtree P obtained by discarding all but one premise of each modal rule (and the entire subtree above a removed vertex). If the pruning always selects a major premise of an undeleted modal rule $\Diamond^*\mathsf{L}$ then by labelling these rules as $\Diamond\mathsf{L}$ the pruning can be conceived as a derivation in $\mathsf{K}\mu$. If T is a template then any pruning to a $\mathsf{K}\mu$ derivation will be a cyclic derivation.

▶ **Proposition 3.11.** *Let* T *be a template for* $\Gamma \subset \Pi_1 \cup \Sigma_1$*. If* $\Gamma \Rightarrow$ *is valid then* T *admits a pruning to a cyclic proof.*

Proof. Let T be a template for $\Gamma \subset \Sigma_1 \cup \Pi_1$. We recursively label each vertex of T as either *successful* or *unsuccessful*. A bud is successful iff it fulfils the criterion of Definition 3.5, namely the left cedent contains a genuine Σ_1 formula. A vertex labelled by $\bigvee\mathsf{L}$ is successful iff all of its premises are successful and a vertex labelled by $\Diamond^*\mathsf{L}$ is successful iff at least one major premise is successful. Initial sequents are always successful. All other rules are unary and are successful iff the single premise is successful.

If the root of T is successful then there is pruning of T to a cyclic proof. For the purpose of the claim we may assume that the root of T is unsuccessful. It suffices to show that $\langle\!|\bigwedge\Gamma|\!\rangle \neq \emptyset$, from which it follows that $\Gamma \Rightarrow$ is not valid. Let U be a subtree of T comprising the unsuccessful vertices but with only one successor chosen for any instance of $\bigvee\mathsf{L}$. By recursion through U we construct a frame $F = \langle W, E, r, \lambda\rangle$ and a winning strategy for Verifier in the game $G(\bigwedge\Gamma, F)$. The root of U is associated the frame comprising a single root world r with empty assignment and the Verifier strategy consists of just the starting position $(\bigwedge\Gamma, w_r)$ and the next available moves $\{(\gamma, w_r) \mid \gamma \in \Gamma\}$. For non-modal rules, the current world is unchanged from conclusion to premise and the leaf in the strategy corresponding to the principal formulas is extended by one step. In all cases but disjunction, this is straightforward. If the principal formula is a disjunction $\bigvee\Delta$ then, as the present vertex of U is unsuccessful, a premise of the rule is in U and the strategy is extended by selecting the active formula for this premise. Upon reaching a modal rule, we know that all major premises are present in U. Let $\Diamond\Gamma, \Box\Delta, L$ be the left cedent at this vertex of T and let w be the world which has been associated to this vertex. The first task is to specify the assignment at w which we take to be the set of constants present in the sequent, i.e., $\lambda(w) = L \cap \mathrm{Cons}$. This definition is well-formed as every world created in F gets associated to exactly one conclusion of a modal rule. Next, the frame is extended by adding a fresh world, w_γ, and edge (w, w_γ) to F for each $\gamma \in \Gamma$ and associating the world w_γ to the major premise whose active formula is γ. The extension of the Verifier strategy is the obvious one: positions $(\Diamond\gamma, w)$ in S are continued to (γ, w_γ), a position $(\Box\delta, w)$ is extended by all available moves $\{(\delta, w_\gamma) \mid \gamma \in \Gamma\}$ and positions (l, w) (for $l \in L$) are terminating in the strategy.

Once the recursive construction of F and S has reached all buds of U, the intended frame and strategy is completed by identifying leaves of F with the world associated to the companion in T and likewise for S. Consider any play consistent with S. If the play is finite, the final position was created from a modal rule in U and, by design, the play is winning for Verifier. If the play is infinite then it infinitely often contains a formula in an unsuccessful bud of T. As such formulas are all Π_1, the plays are winning for Verifier. ◀

▶ **Proposition 3.12** (Completeness). *For $\Gamma \cup \Delta \subset \Sigma_1 \cup \Pi_1$, if $\Gamma \Rightarrow \Delta$ is valid then $\vdash \Gamma \Rightarrow \Delta$ with a cyclic proof of height $\leq 2^{|\mathsf{FL}(\Gamma)|+|\mathsf{FL}(\Delta)|+1}$.*

Proof. By Proposition 3.11 it suffices to confirm that a template for Γ exists with the required height. Let us call a logical rule of $\mathsf{K}\mu$ *conservative* if its principal formula is not also a side formula. For the logical rules in Figure 1, the conservative instances are those where the

principal formula of a left/right rule is not also a side in the same cedent. Thus, of the two following instances of $\mu\mathsf{L}$, the first is conservative if $\mu x\,\alpha \notin \Gamma$ and the second is an example of a non-conservative instance of the same rule regardless of Γ.

$$\frac{\Gamma, \alpha(\mu x\,\alpha/x) \Rightarrow \Delta}{\Gamma, \mu x\,\alpha \Rightarrow \Delta}\,\mu\mathsf{L} \qquad \frac{\Gamma, \mu x\,\alpha, \alpha(\mu x\,\alpha/x) \Rightarrow \Delta}{\Gamma, \mu x\,\alpha \Rightarrow \Delta}\,\mu\mathsf{L}$$

Starting with root $\Gamma \Rightarrow$, systematically apply conservative instances of non-modal (left) rules in any order. Each application of a conservative non-modal rule reduces a particular complexity measure on cedents given by $|\Gamma|_{\Diamond^*\mathsf{L}} = \sum_{\gamma\in\Gamma} |\gamma|_{\Diamond^*\mathsf{L}}$ where $|\gamma|_{\Diamond^*\mathsf{L}}$ is defined as $|\gamma|$ except that $|\Box\delta|_{\Diamond^*\mathsf{L}} = |\Diamond\delta|_{\Diamond^*\mathsf{L}} = 0$ for all formulas δ. The only non-obvious case is that of the quantifiers, where we recall that sequents comprise only guarded formulas and that if $\mu x\,\alpha$ is locally guarded then $|\alpha(\mu x\,\alpha/x)|_{\Diamond^*\mathsf{L}} < |\mu x\,\alpha|_{\Diamond^*\mathsf{L}}$.

Thus, the above process concludes in a finite derivation in non-modal left rules wherein every leaf is a sequent of the form $\Diamond\Theta, \Box\Delta, L \Rightarrow$ for $L \subset \mathrm{Lit}$. If L is inconsistent, this sequent is an instance of idL; otherwise, apply the branching modal rule $\Diamond^*\mathsf{L}$ and repeat the process from each premise. As every sequent produced will be a subset of the Fischer–Ladner closure of Γ, which is finite, every branch will eventually repeat. Closing the derivation at the first repeated sequent as a bud and inserting an instance of the companion rule yields a template. The bound on height arises from the observation that after the insertion of companions, each branch of the template duplicates a sequent along it at most once, except for the bud which is duplicated twice. ◀

Combining the above we obtain:

▶ **Theorem 3.13.** *For $\Gamma, \Delta \subset \Pi_1 \cup \Sigma_1$, the following are equivalent*

1. $\Gamma \Rightarrow \Delta$ *is valid.*
2. $\vdash \Gamma \Rightarrow \Delta$ *is witnessed by a cyclic proof with height* $\leq 2^{|\mathsf{FL}(\Gamma)|+|\mathsf{FL}(\Delta)|+1}$.

The arguments underlying the proof of soundness and completeness are not constrained to the first quantifier level. Given a cyclic derivation D of a sequent $\Gamma \Rightarrow$, the soundness argument translates Verifier strategies in a game $G(\bigwedge\Gamma, F)$ for some F into paths through D that, in a precise sense, carries plays consistent with the strategy. If the strategy is winning for Verifier then the path will be infinite. Dually, the proof of Proposition 3.11 describes a translation of certain subtrees of templates into Verifier strategies over certain frames, with a similar correspondence between paths through the derivation and plays consistent with the strategy. The role of the syntactic restriction on sequents is to link the *winning* strategies to cyclic *proofs*. In both proofs, the success condition on the buds of a cyclic proof ensures, via Lemma 2.18, that the described Verifier strategies are winning.

Two notable corollaries to Theorem 3.13 arise from specialising the form of the sequent.

▶ **Corollary 3.14.** *For $\alpha \in \Pi_1$ and $\beta \in \Sigma_1$, $\alpha \Rightarrow \beta$ is valid iff there exists a finite bud-free derivation of $\alpha \Rightarrow \beta$ of height at most* $2^{|\mathsf{FL}(\alpha)|+|\mathsf{FL}(\beta)|+1}$.

Proof. For $\alpha \in \Pi_1$ and $\beta \in \Sigma_1$, a cyclic proof of $\alpha \Rightarrow \beta$ is, by requirement, bud-free. ◀

▶ **Corollary 3.15.** *For $\alpha \in \Sigma_1$ and $\beta \in \Pi_1$, $\alpha \Rightarrow \beta$ is valid iff there exists a cyclic derivation of $\alpha \Rightarrow \beta$ of height at most $2^{|\mathsf{FL}(\alpha)|+|\mathsf{FL}(\beta)|+1}$.*

Proof. Similar to Corollary 3.14, but noting that for $\alpha \in \Sigma_1$ and $\beta \in \Pi_1$, every cyclic derivation is a cyclic proof. ◀

4 Lyndon Interpolation at the First Level

Via the simple cyclic proofs introduced in the previous section, Craig and Lyndon interpolation can be established for the first level of the modal μ-calculus, namely for sequents built from $\Sigma_1 \cup \Pi_1$. Lyndon interpolation is the property that if $\vdash \alpha \Rightarrow \beta$ then $\vdash \alpha \Rightarrow \iota$ and $\vdash \iota \Rightarrow \beta$ for some ι with $\mathrm{sig}(\iota) \subset \mathrm{sig}(\alpha) \cap \mathrm{sig}(\beta)$. The formula ι is called the *interpolant*. Recall that $\mathrm{sig}(\alpha)$ – the *signature* of α – records the polarity (negated or unnegated) of the constants occurring within α. In particular, for α closed $\mathrm{sig}(\alpha) = \mathsf{FL}(\alpha) \cap \mathrm{Lit}$.

The construction of interpolants directly generalises the proof-theoretic method for modal logic K presented in Chapter 4. That argument recursively ascribes an interpolant to each sequent in a finite proof, starting at leaves and progressing towards the root through the modal and other logical rules. The desired Craig/Lyndon interpolant is the formula assigned to the root sequent.

The quantifier rules $\sigma\mathsf{L}$ and $\sigma\mathsf{R}$ do not affect the construction of interpolants as the signature is unaltered from premise and conclusion. That is, an interpolant chosen for the premise of such a rule is already an interpolant for the conclusion. As we shall see, it is the relaxation to *cyclic* proofs that presents the main hurdle to lifting interpolation from K to $\mathsf{K}\mu$.

4.1 Lyndon Candidates

Lifting interpolation from bud-free to cyclic proofs is not immediate. To visualise the difficultly, consider the cyclic proof of the sequent $b, \mu x.\, \Box(x \wedge a) \vee \Diamond(b \wedge x) \Rightarrow \nu x(\Box a \vee \Diamond x)$ from Figure 2, repeated in Figure 6. To directly apply the recursive construction from Chapter 4 one must first assign an 'interpolant' to the single bud $b, \alpha \Rightarrow \beta$ before building from this an interpolant for the root. Let z denote a hypothetical interpolant of the bud sequent. Following the standard construction leads to the formula $\Box a \vee \Diamond z$ as an interpolant for the sequent at the root (cf. left-hand derivation in Figure 6, ignoring the companion vertex). As bud and end-sequent are identical, it follows that each of z, $\Box a \vee \Diamond z$, $\Box a \vee \Diamond(\Box a \vee \Diamond z)$, etc. are all interpolants for $b, \alpha \Rightarrow \beta$. The dependency on z can be removed by finding a formula $\iota \equiv \Box a \vee \Diamond \iota$. That is, we can hope that an interpolant of the companion will be either $\mu z(\Box a \vee \Diamond z)$ or $\nu z(\Box a \vee \Diamond z)$ with the choice of quantifier determined by properties of the cyclic proof as a whole. For this specific example, the reader can confirm that both formulas are (Lyndon) interpolants, though this is not in general the case.

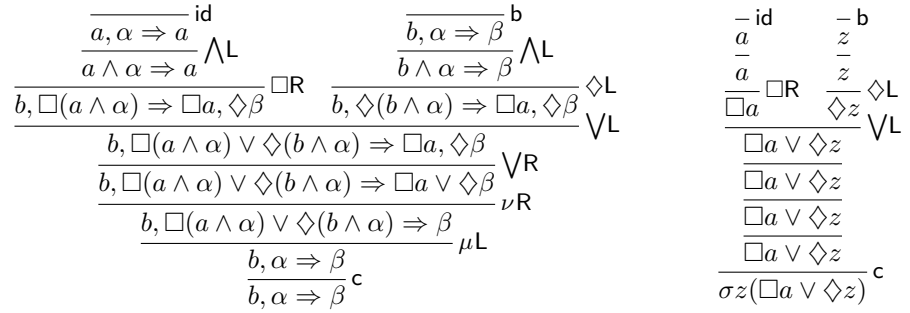


Figure 6 The cyclic proof from Figure 2 (left) and the formula associated to each vertex following Construction 1 for an arbitrary marking (right), with the root formula being the Lyndon candidate induced by this derivation. As before, $\alpha = \mu x(\Box(a \wedge x) \vee \Diamond(b \wedge x))$ and $\beta = \nu x(\Box a \vee \Diamond x)$.

The introduction of quantifiers into interpolants can be identified with companion vertices, meaning that the construction remains recursive in the given cyclic proof. The process begins at buds which are assigned variables so that buds with a common companion are given the same variable. Interpolants for other vertices are chosen to mirror the structure of the derivation, with quantifiers being introduced by companions, illustrated in Figure 6.

For reasons that will become clear, the choice of quantifier will be specified as an input to the construction of interpolants along with the cyclic derivation. Define a *marked (cyclic) derivation* to be a pair (D, σ) where D is a (cyclic) derivation and $\sigma\colon \mathrm{comp}(D) \to \{\mu, \nu\}$ is an assignment of a quantifier (either μ or ν) to each companion. The function σ will be referred to as a *marking* of D. The construction of a 'candidate' Lyndon interpolant from a marked cyclic derivation is formalised by the following.

▶ **Construction 1.** *Let (D, σ) be a marked cyclic derivation. Let $v \mapsto x_v\colon \mathrm{comp}(D) \to \mathrm{Var}$ be an injection of companion vertices in D into variables. In the following we write σ_v for $\sigma(v)$. If $v \in D$ has just one immediate successor, we denote it by v^+. Each $v \in D$ is associated a formula ι_v defined by recursion on D:*

Bud: *$\iota_v = x_{v^c}$, namely the variable associated to the bud's companion.*

Logical Rule:

- **id.** *$\iota_v = l$ where l is the principal literal.*
- **idL.** *$\iota_v = \bot$.*
- **idR.** *$\iota_v = \top$.*
- **$\bigvee$L.** *$\iota_v = \bigvee\{\iota_w \mid w \in \mathrm{Suc}_D(v)\}$.*
- **$\bigwedge$R.** *$\iota_v = \bigwedge\{\iota_w \mid w \in \mathrm{Suc}_D(v)\}$.*
- **$\Diamond$L.** *$\iota_v = \Diamond \iota_{v^+}$.*
- **$\Box$R.** *$\iota_v = \Box \iota_{v^+}$.*
- ***Any other logical rule.*** *$\iota_v = \iota_{v^+}$.*

Companion: *$\iota_v = \sigma_v x_v \, \iota_{v^+}$.*

The Lyndon candidate induced by (D, σ) *is the formula* $\iota_{(D,\sigma)} = \iota_r$ *where* r *is the root of* D.

We begin with some observations concerning the structure of Lyndon candidates. Let (D, σ) be any marked cyclic derivation.

▶ **Lemma 4.1.** $|\iota_{(D,\sigma)}| \leq h(D)$.

▶ **Lemma 4.2.** *If the endsequent of* D *is* $\Gamma \Rightarrow \Delta$ *then* $\mathrm{sig}(\iota_{(D,\sigma)}) \subset \mathrm{sig}(\Gamma) \cap \mathrm{sig}(\Delta)$.

▶ **Lemma 4.3.**

1. *If* $\mathrm{comp}(D) = \emptyset$ *then* $\iota_{(D,\sigma)}$ *is quantifier-free.*
2. *If* $\sigma(u) = \mu$ *for every* $u \in \mathrm{comp}(D)$ *then* $\iota \in \Sigma_1$.
3. *If* $\sigma(u) = \nu$ *for every* $u \in \mathrm{comp}(D)$ *then* $\iota \in \Pi_1$.

All three lemmas above are immediate consequences of Construction 1. The following generalisation of Lemma 4.3 will be useful in Section 6.

▶ **Lemma 4.4.**

1. *If* $\sigma(u) = \sigma(v)$ *whenever* $u \lhd_D v \in \mathrm{comp}(D)$, *then* $\iota_{(D,\sigma)}$ *is alternation-free.*
2. *If for every sequence* $u_0 \lhd_D \cdots \lhd_D u_k \in \mathrm{comp}(D)$ *we have* $|\{i < k \mid \sigma(u_i) \neq \sigma(u_{i+1})\}| \leq n$, *then* $\iota_{(D,\sigma)} \in \mathit{Clos}(\Sigma_{n+1} \cup \Pi_{n+1})$.

To validate that a candidate interpolant actually interpolates the intended sequent, it remains to build the witnessing proofs. This step also follows the proof-theoretic method presented in Chapter 4 and Chapter 5 whereby the derivation is modified by inserting in place of either the left or right cedent a subformula of the candidate interpolant. From a cyclic derivation of $\alpha \Rightarrow \beta$ and a candidate interpolant $\iota = \iota_{(D,\sigma)}$ derived from this derivation we can, without difficulty, obtain cyclic derivations for the sequents $\alpha \Rightarrow \iota$ and $\iota \Rightarrow \beta$.

Whether the constructed derivations are cyclic *proofs* depends on properties of the marking. Under the constant marking $u \mapsto \nu$, the sequent $\alpha \Rightarrow \iota$ always admits a cyclic proof, whereas provability of $\iota \Rightarrow \beta$ is trivialised by the opposite marking $u \mapsto \mu$. Finding a compromise that works for *both* sequents is the main hurdle to proving interpolation, though this problem manifests only for sequents beyond the first quantifier level.

The next two results describe the transformation in detail. As mentioned, the idea is to replace in each sequent of a cyclic proof $P \vdash \alpha \Rightarrow \beta$ the left (resp. right) cedent by the candidate interpolant associated to that vertex. Taken literally, however, this will not yield a derivation because the interpolants associated to the premise and conclusion of a companion cannot be linked by a rule of the calculus. The former will be a formula ι_u with a free variable x_v whereas the latter will be a quantified formula $\iota_v = \sigma_v x_v \, \iota_u$. But as the reader has likely guessed, the 'real' interpolant associated to the premise of a companion should be the unfolding of ι_v, namely $\iota_u(\iota_v/x_v)$. The necessary substitutions are expressed as follows.

▶ **Definition 4.5** (Lyndon candidate at a vertex)**.** *Let (D, σ) be a marked cyclic derivation with root $\Gamma \Rightarrow \Delta$ and let ι_v be the formula associated to v by Construction 1. Let $C_{\lhd v} = \{u \in \mathrm{comp}(D) \mid u \lhd_D v\}$ be the companion vertices strictly below $v \in D$ relative to the ordering $\lhd_D$. For each $v \in D$, the* Lyndon candidate at v *is the formula $\hat{\iota}_v$ defined as*

$$\hat{\iota}_v = \iota_v(\iota_{c_1}/x_{c_1}) \cdots (\iota_{c_n}/x_{c_n}) \quad \text{where } c_1 < \cdots < c_n < v \text{ enumerate the elements of } C_{\lhd v}.$$

The reader can confirm that $\hat{\iota}_r = \iota_r = \iota_{(D,\sigma)}$ where r is the root of D and that $\hat{\iota}_v$ is a closed formula for every v.

The next results describe the cyclic derivations obtained by combining a cyclic proof with the candidate interpolants.

▶ **Lemma 4.6.** *Let (D, σ) be a marked cyclic derivation with root $\alpha \Rightarrow \beta$ and let $\Gamma_v \Rightarrow \Delta_v$ denote the sequent labelling vertex $v \in D$. Let $\hat{\iota}_v$ be the Lyndon candidate at v. There exist cyclic derivations of*

1. *$\alpha \Rightarrow \iota_{(D,\sigma)}$ with buds $\{\Gamma_w \Rightarrow \hat{\iota}_w \mid w \in \mathrm{comp}(D)\}$.*
2. *$\iota_{(D,\sigma)} \Rightarrow \beta$ with buds $\{\hat{\iota}_w \Rightarrow \Delta_w \mid w \in \mathrm{comp}(D)\}$.*

Lemma 4.6 is a special case of the following refinement. Although the next result violates the assumption that sequents comprise closed formulas only, we take it as implicit that the variables in the statement are considered as fresh constants.

▶ **Proposition 4.7.** *Let (D, σ) be a cyclic derivation as per Lemma 4.6 and fix a vertex v. Let $C_{\lhd v} = \{w \in \mathrm{comp}(D) \mid w \lhd_D v\}$ and $C_{\geq v} = \{w \in \mathrm{comp}(D) \mid v \leq w\}$. There exist finite derivations of:*

1. *$\Gamma_v \Rightarrow \iota_v$ from initial sequents $\{\Gamma_w \Rightarrow x_w \mid w \in C_{\lhd v}\}$ and $\{\Gamma_w \Rightarrow \hat{\iota}_w \mid w \in C_{\geq v}\}$, wherein buds of the latter kind each have a companion.*
2. *$\iota_v \Rightarrow \Delta_v$ from initial sequents $\{x_w \Rightarrow \Delta_w \mid w \in C_{\lhd v}\}$ and $\{\hat{\iota}_w \Rightarrow \Delta_w \mid w \in C_{\geq v}\}$, wherein buds of the latter kind each have a companion.*

Proof. By induction on v. If v is a bud, then $v^c \in C_{\lhd v}$ and the claim is trivial. If v is an identity sequent, so are $\Gamma_v \Rightarrow \iota_v$ and $\iota_v \Rightarrow \Delta_v$. Suppose that $v \in \mathrm{comp}(D)$ and let v^+ be the immediate successor of v. We treat case 1. By the induction hypothesis, there is a finite derivation of $\Gamma_{v^+} \Rightarrow \iota_{v^+}$ from initial sequents

$$\{\Gamma_w \Rightarrow x_w \mid w \in C_{\lhd v^+}\} \cup \{\Gamma_w \Rightarrow \iota_w \mid w \in C_{\geq v^+}\}.$$

Note, $C_{\lhd v^+} = C_{\lhd v} \cup \{v\}$. Uniformly substituting $\iota_v = \sigma_v x_v \, \iota_{v^+}$ for x_v in this derivation yields a finite derivation of $\Gamma_v \Rightarrow \iota_{v^+}(\iota_v/x_v)$ from initial sequents $\{\Gamma_w \Rightarrow w \mid w \in C_{\lhd v}\}$ and $\{\Gamma_w \Rightarrow \iota_w \mid w \in C_{\geq v}\}$. Appending an instance of the rule $\sigma_v\mathsf{R}$ to the root of this derivation followed by the companion rule c yields a derivation of $\Gamma_v \Rightarrow \iota_v$ from the desired initial sequents. The remaining cases are straightforward. ◀

4.2 Lyndon Interpolation and Applications

Considering derivations with no buds, Lemmas 4.1 and 4.6 imply our first interpolation result for the modal μ-calculus:

▶ **Theorem 4.8** (Quantifier-free Lyndon Interpolation). *Suppose* $\vdash \alpha \Rightarrow \beta$ *with* $\alpha \in \Pi_1$ *and* $\beta \in \Sigma_1$. *There exists a quantifier-free formula* ι *such that*
1. $\mathrm{sig}(\iota) \subset \mathrm{sig}(\alpha) \cap \mathrm{sig}(\beta)$.
2. $\vdash \alpha \Rightarrow \iota$ *and* $\vdash \iota \Rightarrow \beta$.
3. $|\iota| \leq 2^{|\mathsf{FL}(\alpha)|+|\mathsf{FL}(\beta)|+1}$.

Proof. Let D be a cyclic proof witnessing $\vdash \alpha \Rightarrow \beta$. This proof, by virtue of the assumption on α and β, contains no buds and, hence, we may assume that it has no companions. Let $\iota = \iota_{(D,\sigma)}$ where σ is the empty marking. Lemma 4.2 confirms that ι is quantifier-free and satisfies the desired restriction on signatures. Lemma 4.6 describes cyclic derivations of the sequents $\alpha \Rightarrow \iota$ and $\iota \Rightarrow \beta$. As D contains no companions, the derivations are bud-free (finite) proofs. The bound is based on size of derivation is a consequence of Corollary 3.14 and Lemma 4.1. ◀

As the quantifier-free formulas are both Π_1 and Σ_1, Theorem 4.8 yields

▶ **Corollary 4.9.** *Modal logic* K *has the Lyndon interpolation property.*

Reading Theorem 4.8 as a separation result yields

▶ **Corollary 4.10** (Π_1-separation). *Let* $\alpha, \beta \in \Pi_1$. *If* $\langle\!|\alpha|\!\rangle \cap \langle\!|\beta|\!\rangle = \emptyset$ *then there exists a quantifier-free* ι *such that* $\langle\!|\alpha|\!\rangle \subset \langle\!|\iota|\!\rangle$ *and* $\langle\!|\iota|\!\rangle \cap \langle\!|\beta|\!\rangle = \emptyset$.

Proof. Let $\alpha, \beta \subset \Pi_1$ be such that $\langle\!|\alpha|\!\rangle \cap \langle\!|\beta|\!\rangle = \emptyset$. Then $\beta^\perp \in \Sigma_1$ and $\langle\!|\alpha|\!\rangle \subset \langle\!|\beta^\perp|\!\rangle$ by Corollary 2.12. By completeness, $\vdash \alpha \Rightarrow \beta^\perp$ and Theorem 4.8 yields a quantifier-free formula ι fulfilling the requirements. ◀

A special case of the previous corollary is:

▶ **Corollary 4.11** (Δ_1-collapse). $\Delta_1 = \Delta_0$. *That is, every* Δ_1*-formula is equivalent to a quantifier-free formula.*

Proof. By Corollary 4.10: Suppose $\alpha \in \Pi_1$ and $\alpha^\perp \equiv \beta \in \Pi_1$. Since $\langle\!|\alpha|\!\rangle \cap \langle\!|\alpha^\perp|\!\rangle = \emptyset$ there is a formula $\iota \in \Delta_0$ such that $\langle\!|\alpha|\!\rangle \subset \langle\!|\iota|\!\rangle \subset \langle\!|\alpha|\!\rangle$. ◀

The next results treat the other scenarios of interpolating sequents at the first level.

▶ **Theorem 4.12** (Σ_1-Lyndon Interpolation). *Suppose* $\vdash \alpha \Rightarrow \beta$ *with* $\alpha, \beta \in \Sigma_1$. *There exists a* Σ_1 *formula* ι *such that*
1. $\mathrm{sig}(\iota) \subset \mathrm{sig}(\alpha) \cap \mathrm{sig}(\beta)$.
2. $\vdash \alpha \Rightarrow \iota$ *and* $\vdash \iota \Rightarrow \beta$.

Proof. Let (D, σ) be a marked cyclic proof of $\vdash \alpha \Rightarrow \beta$ where $\sigma_v = \mu$ for all companions. Let $\iota = \iota_{(D,\sigma)}$ be the candidate interpolant and let D_1 and D_2 be the cyclic derivations of $\alpha \Rightarrow \iota$ and $\iota \Rightarrow \beta$ respectively given by Lemma 4.6. We observe that these derivations are cyclic proofs. A bud $v \in \mathrm{bud}(D_1)$ will be labelled by a sequent $\Gamma_w \Rightarrow \hat{\iota}_w$ where $w \in \mathrm{comp}(D)$. As D is a proof but $\beta \in \Sigma_1$, Γ_w contains a genuine Σ_1 formula. So v is left successful. Similarly, every bud v of D_2 is left successful as, by construction, $\hat{\iota}_w$ is genuine Σ_1 for $w \in \mathrm{comp}(D)$. ◀

▶ **Corollary 4.13** (Π_1-Lyndon Interpolation). *Suppose $\vdash \alpha \Rightarrow \beta$ with $\alpha, \beta \in \Pi_1$. There exists a Π_1 formula ι such that*

1. $\mathrm{sig}(\iota) \subset \mathrm{sig}(\alpha) \cap \mathrm{sig}(\beta)$.
2. $\vdash \alpha \Rightarrow \iota$ *and* $\vdash \iota \Rightarrow \beta$.

One configuration not covered by the above results is for sequents $\alpha \Rightarrow \beta$ where $\alpha \in \Sigma_1$ and $\beta \in \Pi_1$. From Theorem 3.13 we know that such a sequent is valid iff it admits a cyclic proof. The difficulty with this scenario, however, is that the constructed interpolant will not in general be Σ_1 or Π_1. This happens in the case of a derivation D with three companions $u < v_1, v_2$ where the left cedent at v_1 is empty and the right cedent at v_2 is empty. For the construction above to work, the chosen marking would need to send v_1 to ν and v_2 to μ, resulting in a candidate interpolant for u in neither Σ_1 nor Π_1. However, a simple modification of the construction achieves the aim.

▶ **Theorem 4.14.** *Suppose $\vdash \alpha \Rightarrow \beta$ with $\alpha \in \Sigma_1$ and $\beta \in \Pi_1$. There exists a Σ_1 formula ι such that*

1. $\mathrm{sig}(\iota) \subset \mathrm{sig}(\alpha) \cap \mathrm{sig}(\beta)$.
2. $\vdash \alpha \Rightarrow \iota$ *and* $\vdash \iota \Rightarrow \beta$.

Proof. Let ι be the Lyndon candidate induced by D with the trivial marking $\sigma \colon u \mapsto \mu$ but with the alteration that a vertex with empty left cedent is assigned the candidate $\iota_v = \top$. ◀

5 Uniform Interpolation at the First Level

The method of constructing Lyndon interpolants can be adapted to generate *uniform* interpolants based on observations made by Pitts [26] (see also Chapter 5 and Chapter 9). The idea is to explore, via proof-search, all ways in which a given formula α can influence a proof of a conditional sequent of the form $\alpha \Rightarrow \beta$ and encode this information as a single formula of L_μ. The evolution of left cedents in cyclic derivations is captured precisely by the template calculus of Section 3.1. This section demonstrates the encoding of a template T for a formula $\alpha \in \Sigma_1 \cup \Pi_1$ as a formula $\iota \in \Pi_1$ which uniformly interpolates α against all Π_1 formulas. The general uniform interpolation property is addressed in Section 6.

The template calculus, recall, is the sequent calculus of structural and left logical rules of $\mathsf{K}\mu$ with the modal rule $\Diamond\mathsf{L}$ replaced by a branching form:

$$\frac{\gamma_1, \Delta \Rightarrow \quad \cdots \quad \gamma_n, \Delta \Rightarrow \quad \Delta \Rightarrow}{\Diamond\gamma_1, \ldots, \Diamond\gamma_n, \Box\Delta, L \Rightarrow} \Diamond^*\mathsf{L}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a,\alpha\Rightarrow}\,\mathsf{b}\quad\dfrac{}{\alpha\Rightarrow}\,\mathsf{b}}{a,\Diamond a,\Box\alpha\Rightarrow}\,\Diamond^*\mathsf{L}}{a,\Diamond a\wedge\Box\alpha\Rightarrow}\,\bigwedge\mathsf{L}}{a,\alpha\Rightarrow}\,\nu\mathsf{L}}{a,\alpha\Rightarrow}\,\mathsf{c}\qquad\dfrac{}{\alpha\Rightarrow}\,\mathsf{b}}{\dfrac{\dfrac{\dfrac{\Diamond a,\Box\alpha\Rightarrow}{\Diamond a\wedge\Box\alpha\Rightarrow}\,\bigwedge\mathsf{L}}{\alpha\Rightarrow}\,\nu\mathsf{L}}{\alpha\Rightarrow}\,\mathsf{c}}\,\Diamond^*\mathsf{L}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{x}\,\mathsf{b}\quad\overline{y}\,\mathsf{b}}{\Diamond x\wedge\Box(x\vee y)}\,\Diamond^*\mathsf{L}}{\Diamond x\wedge\Box(x\vee y)}}{\Diamond x\wedge\Box(x\vee y)}}{\Diamond x\wedge\Box(x\vee y)}}{\nu x(\Diamond x\wedge\Box(x\vee y))}\,\mathsf{c}\qquad\overline{y}\,\mathsf{b}}{\dfrac{\dfrac{\dfrac{\dfrac{\Diamond\gamma\wedge\Box(y\vee\gamma)}{\Diamond\gamma\wedge\Box(y\vee\gamma)}}{\Diamond\gamma\wedge\Box(y\vee\gamma)}}{\Diamond\gamma\wedge\Box(y\vee\gamma)}}{\nu y(\Box\gamma\vee\Box(y\vee\gamma))}\,\mathsf{c}}\,\Diamond^*\mathsf{L}$$

Figure 7 An α-template where $\alpha=\nu x(\Diamond a\wedge\Box x)$ and the uniform interpolant relative to Π_1 formulas and the empty signature; $\gamma=\nu x(\Diamond x\wedge\Box(x\vee y))$.

and subject to the constraint that L is a consistent set of literals. The formulas $\Diamond\gamma_1,\ldots,\Diamond\gamma_n$ are considered the principal formulas of the rule and the corresponding premises $\gamma_i,\Delta\Rightarrow$ are the *major* premises. The final premise $\Delta\Rightarrow$ with no active formula is the *minor* premise of the rule. A cyclic derivation in the template calculus is called a *template*. We recall also the concept of a *pruning* of a template T, which is any subtree of T given by removing all but one premise of each instance of $\Diamond^*\mathsf{L}$ in T and any vertex above a removed premise.

Unlike our use of the template calculus in Section 3, the minor premise plays an important role in what follows. We view a template for a cedent Γ as a record of the (potential) dependence on Γ of an arbitrary cyclic proof with endsequent $\Gamma\Rightarrow\Delta$. In the simplest scenario a cyclic proof D of $\Gamma\Rightarrow\Delta$ can be seen as carving out a pruning of the fixed template T. Instances of $\Diamond\mathsf{L}$ in D pick out a major premise of $\Diamond^*\mathsf{L}$ and the rule $\Box\mathsf{R}$ corresponds to the minor premise. In this light, the proof-search argument of Proposition 3.11 corresponds to the specific case $\Delta=\emptyset$.

Templates can vary in two essential ways. Obviously, two templates may employ different instances of logical rules to derive the same sequent. They could, however, always agree on the choice and instance of logical rules yet differ in respect to which vertices/sequents are buds or which, of potentially many, repetitions are allocated as companions. In other words, two templates could have identical unravellings yet differ as finite derivations.

Changing the order of non-modal rules within a template will never change the instances of $\Diamond^*\mathsf{L}$. As the structure of the logical rules does not affect the success condition, we can readily assume that any two templates for the same cedent agree modulo the selection of buds and companions. In other words, we can implicitly assume that all templates unravel to the same infinite derivation in the template calculus.

A similar observation holds for cyclic derivations in $\mathsf{K}\mu$. The order of non-modal rules is irrelevant, and we can assume that instances of a modal rule always have a maximally decomposed conclusion (each cedent having the form $L,\Box\Gamma,\Diamond\Delta$ for $L\subset\mathrm{Lit}$ consistent) and

maximal choice of side formulas, though the choice of principal formula is permitted to vary between instances. These assumptions allow a cyclic derivation in $\mathsf{K}\mu$ to be related to a pruning of the template along the lines of Proposition 3.11. The statement of the connection relies on one further operation on derivations. For a detailed account, we refer the reader to [4]; a different approach to the same problem is presented in [5].

The *(left) projection* of a $\mathsf{K}\mu$-derivation D is the tree D_{L} in which the right cedent at every vertex has been removed along with any vertex that is the conclusion of a non-modal right rule. We tacitly assume that the root of D is a left rule (otherwise D_{L} may not be a tree), but such an assumption is harmless.

▶ **Assumption 1** (Canonicity of derivations)**.** *For any Γ-template T and bud-free derivation D with endsequent $\Gamma' \Rightarrow \Delta$ where $\Gamma' \subseteq \Gamma$, D_{L} is a pruning of the unravelling of T modulo the removal of some non-modal and structural rules from the latter.*

5.1 Uniform Candidates

The construction of a candidate uniform interpolant from a template follows the same procedure as that of Lyndon interpolants, with the only alteration in the treatment of the branching modal rule. A *marked template* is a pair (T, σ) where T is a template and σ, called the *marking*, is a function from the set of companions of T into $\{\mu, \nu\}$.

▶ **Construction 2.** *Fix a signature S. Let (T, σ) be a marked cyclic derivation and let $v \mapsto x_v \colon \mathrm{comp}(D) \to \mathrm{Var}$ be an injection of companion vertices in D into variables. We write σ_v for $\sigma(v)$. Let Φ_v be the left cedent at vertex $v \in T$. Each vertex $v \in T$ is associated a formula ι_v defined by recursion on T, with the first applicable case taking precedence:*

Empty sequent: *If $\Phi_v = \emptyset$ then $\iota_v = \top$.*

Bud: *$\iota_v = x_{v^c}$, namely the variable associated to the bud's companion.*

Identity: *$\iota_v = \bot$*

Modal Rule: *Let $v_0, \ldots, v_n$ enumerate the immediate successors of v with the minor premise last. Define*

$$\iota_v = \bigwedge \Gamma \text{ where } \Gamma = (\Phi_v \cap S) \cup \left\{ \Diamond \iota_{v_i} \;\middle|\; i < n \right\} \cup \left\{ \Box(\iota_0 \vee \cdots \vee \iota_n) \right\}.$$

Logical Rule $\bigvee\mathsf{L}$: *$\iota_v = \bigvee\{\iota_w \mid w \in \mathrm{Suc}_T(v)\}$.*

Any other logical rule: *$\iota_v = \iota_{v^+}$.*

Companion: *$\iota_v = \sigma_v x_v \, \iota_{v^+}$.*

The uniform candidate induced by *(T, σ, S) is the formula $\iota_{(T,\sigma,S)} = \iota_r$ where r is the root of T.* ⌟

▶ **Example 5.1.** Figure 7 presents an example of Construction 2 to a template for the Π_1 formula $\nu x(\Diamond a \wedge \Box x)$. ⌟

The verification that certain templates induce uniform interpolants boils down to properties of the marking of T. Given a marked α-template (T, σ), a cyclic derivation of $\alpha \Rightarrow \iota_{(T,\sigma,S)}$ is readily available, given by inserting into T a substituted form of the formula ι_v as the right cedent at each vertex v. The derivation will be a cyclic proof provided that sufficient companions of T are assigned the quantifier ν (as these induce right successful buds). Furthermore, the template induces a transformation on cyclic derivation of sequents $\alpha \Rightarrow \beta$ into derivations of $\iota_{(T,\sigma,S)} \Rightarrow \beta$ by a similar process, subject to constraints on the signature of β. That the transformation preserves the proofhood condition depends on sufficiently many companions being marked μ (inducing left successful buds). Uniform interpolation relies on finding templates whose marking satisfy these competing conditions.

As in the case of Lyndon candidates, uniform candidates mirror the structure of the derivation – the template – from which they are constructed. If the marking assigns ν to all companions, for instance, the constructed candidate will be Π_1. Indeed, Lemmas 4.1 to 4.4 carry over to the realm of uniform candidates without alteration.

The substitutional instances of the candidate formulas are defined in the same way as for candidate Lyndon interpolants. Let (T, σ) be a marked template. The notation Φ_u is generally used for the left cedent at vertex $u \in T$.

▶ **Definition 5.2** (Uniform candidate at a vertex). *Let ι_v be the formula associated to vertex v by Construction 2. Let $C_{\lhd v} = \{w \in \mathrm{comp}(T) \mid w \lhd_T v\}$ be the companions strictly below $v \in T$ relative to the relation $\lhd_T$. For each $v \in T$, the* uniform candidate at v *is the formula*

$$\hat{\iota}_v = \iota_v(\iota_{c_1}/x_{c_1}) \cdots (\iota_{c_n}/x_{c_n}) \quad \textit{where } c_1 < \cdots < c_n < v \textit{ enumerate the elements of } C_{\lhd v}.$$

The reader can confirm that $\hat{\iota}_r = \iota_r = \iota_{(T,\sigma,S)}$ where r is the root of D and that $\hat{\iota}_v$ is a closed formula for every v.

We begin with correspondents to the Interpolation Lemma 4.6.

▶ **Lemma 5.3** (First Template Lemma). *Let (T, σ) be a marked template for Γ and let Φ_u be the left cedent at $v \in T$. For every signature S there is a cyclic derivation of $\Gamma \Rightarrow \iota_{(T,\sigma,S)}$ in $\mathsf{K}\mu + \mathsf{w}$ with buds labelled by sequents $\Phi_u \Rightarrow \hat{\iota}_u$ for $u \in \mathrm{bud}(T)$.*

Proof. The cyclic derivation of $\Gamma \Rightarrow \iota_{(T,\sigma,S)}$ is constructed by recursion through the template. Let T' be the template T in which the sequent at each vertex $v \in T$ is replaced by $\Phi_v \Rightarrow \hat{\iota}_v$. The root of T' is labelled $\Gamma \Rightarrow \iota_{(T,\sigma,S)}$. To turn T' into a cyclic derivation, we maintain the same bud-companion association and merely insert sufficient right rules between the vertices of T' and replace occurrences of $\Diamond^*\mathsf{L}$ by finite derivations. Thus, a bud/companion $\Phi_b \Rightarrow$ in T becomes a bud/companion $\Phi_b \Rightarrow \hat{\iota}_b$ in T' and an identity sequent $\Phi_v \Rightarrow$ becomes $\Phi_v \Rightarrow \bot$. If $\Phi_v = \emptyset$, then v becomes the provable sequent $\Phi_v \Rightarrow \top$ and the vertices above v are discarded. The transformation in the case of logical rules is straightforward and omitted. This leaves only the modal rule $\Diamond^*\mathsf{L}$ and companions to consider. Suppose the modal rule occurs at a vertex v. Let $v_0, \ldots, v_n$ enumerate the immediate successors of v with the minor premise

last. Let $L = \Phi_v \cap S$. The construction defines

$$\hat{\iota}_v = \bigwedge \Gamma \text{ where } \Gamma = L \cup \{\Diamond \hat{\iota}_{v_i} \mid i < n\} \cup \{\Box(\hat{\iota}_{v_0} \vee \cdots \vee \hat{\iota}_{v_n})\}.$$

The vertex v is expanded into the derivation

$$\dfrac{\dfrac{}{\Phi_v \Rightarrow l}\,\mathsf{id} \quad \text{for } l \in L \qquad \dfrac{\dfrac{\Phi_{v_i} \Rightarrow \hat{\iota}_{v_i}}{\Phi_v \Rightarrow \Diamond \hat{\iota}_{v_i}}\,\Diamond\mathsf{L} \quad \text{for } i < n \qquad \dfrac{\dfrac{\Phi_{v_n} \Rightarrow \hat{\iota}_{v_n}}{\Phi_v \Rightarrow \hat{\iota}_{v_0} \vee \cdots \vee \hat{\iota}_{v_n}}\,\mathsf{w},\bigvee\mathsf{R}}{\Phi_v \Rightarrow \Box(\hat{\iota}_{v_0} \vee \cdots \vee \hat{\iota}_{v_n})}\,\Box\mathsf{R}}{\Phi_v \Rightarrow \hat{\iota}_v}\,\bigwedge\mathsf{R}$$

The remaining case is the companion. Then $\hat{\iota}_v = \sigma_v x_v\, \beta$ for some formula β, and $\hat{\iota}_{v^+} = \beta(\hat{\iota}_v/x_v)$. Inserting an instance of $\sigma_v\mathsf{R}$ above the companion suffices. As bud and companions are associated the same sequent, a cyclic derivation has been constructed. ◀

▶ **Lemma 5.4** (Second Template Lemma). *For any signature S and formula Δ with $\mathrm{sig}(\Delta) \subset S$, given a cyclic derivation D of $\alpha \Rightarrow \Delta$ there exists a cyclic derivation of $\iota_{(T,\sigma,S)} \Rightarrow \Delta$ whose buds (and companions) are sequents of the form $\hat{\iota}_u \Rightarrow \Theta$ such that $\Phi_u \Rightarrow \Theta$ is a bud of D.*

Proof. This argument closely follows that of the Lyndon Interpolation Lemma 4.6. The main complication is the treatment of modal rules.

Let D be a cyclic derivation of $\alpha \Rightarrow \Delta$ and consider its unravelling to a bud-free derivation which we name D_ω. Let $\Phi_u \Rightarrow$ be the sequent at $u \in T$ and $\Gamma_u \Rightarrow \Delta_u$ the sequent at $u \in D_\omega$. The idea is to replace the left cedent in each vertex of D_ω by an appropriately chosen candidate and close off the infinite branches of D_ω to obtain a cyclic derivation. Assumption 1 ensures that the left projection of D_ω aligns with those occurring in the template.

We begin by associating to each vertex $u \in D_\omega$ a set $V_u \subset T$ of template vertices which identifies the candidate interpolants $I_u = \{\hat{\iota}_u \mid u \in V_u\}$ that can act in the place of the left cedent of u. The assignment of template vertices to D_ω is the naive choice determined by compatibility of left cedents: $V_u = \{v \in T \mid \Gamma_u \subset \Phi_v\}$. We now show how to build a bud-free derivation of $\hat{\iota}_v \Rightarrow \Delta_u$ for each $v \in V_u$. Let $S = \{\hat{\iota}_v \Rightarrow \Delta_u \mid v \in V_u\}$ be the collection of such sequents. The construction proceeds co-recursively in the sense that we associate to each sequent $\hat{\iota}_v \Rightarrow \Delta_u$ from S a finite derivation $D_{u,v}$ whose buds are sequents in S. Patching together sufficiently many such derivations will yield a cyclic derivation of the desired form.

Let $v \in V_u$. There are four distinct cases to consider based on the rule at $u \in D_\omega$.

Non-modal right rule. These rules do not alter the left cedent and we simply maintain the right rule employed. If, for example, the rule at u is $\bigwedge\mathsf{R}$ with principal formula $\bigwedge\{\delta_1, \ldots, \delta_n\} \in \Delta_u$, then we have $v \in V_{u'}$ for each successor $u' \in \mathrm{Suc}_D(u)$ and $D_{u,v}$ is the following finite derivation for appropriate choice of Δ.

$$\left.\dfrac{\dfrac{}{\hat{\iota}_v \Rightarrow \delta_1, \Delta}\,\mathsf{b} \quad \cdots \quad \dfrac{}{\hat{\iota}_v \Rightarrow \delta_n, \Delta}\,\mathsf{b}}{\hat{\iota}_v \Rightarrow \Delta_u}\,\bigvee\mathsf{L}\right\} = D_{u,v}$$

Non-modal left rule. Let $\gamma \in \Gamma_u$ be the principal formula at u. If this is also the principal formula at $v \in T$ then $D_{u,v}$ is formed analogously to the previous case as the single left rule with corresponding buds. For instance, in the case that $\gamma = \bigvee\{\gamma_1, \ldots, \gamma_n\}$ then $\hat{\iota}_v = \bigvee\{\hat{\iota}_w \mid w \in \mathrm{Suc}_T(v)\}$ and

$$\left.\dfrac{\dfrac{}{\hat{\iota}_{v_1} \Rightarrow \Delta_u}\,\mathsf{b} \quad \cdots \quad \dfrac{}{\hat{\iota}_{v_n} \Rightarrow \Delta_u}\,\mathsf{b}}{\hat{\iota}_v \Rightarrow \Delta_u}\bigwedge\mathsf{R}\right\} = D_{u,v}$$

The buds each correspond to a sequent in S. However, it may be that γ is not principal at v. First, we may assume that v is not a bud of T as the companion is labelled by the same sequent. Assumption 1 ensures that Γ_u is a subset of each cedent above v until a point that γ is principal. Moreover, only (left) logical rules occur in this portion of the template. So we let $D_{u,v}$ be this finite derivation mimicking the rules that alter the candidate interpolants ending at the point that γ is decomposed.

Identity sequent. The left or right identity rules are covered in the previous cases. So assume that the cedents Δ_u and Φ_v contain a literal l in common. As $\Delta_u \subset \mathsf{FL}(\Delta)$, it follows that $l \in S$. By the previous case we can assume that v is a modal rule, so $\hat{\iota}_u = \bigwedge\Gamma$ with $l \in \Gamma$ and $\hat{\iota}_u \Rightarrow \Delta_u$ has a straightforward (finite) proof.

The final case is that u is a modal rule. As above, we may assume that v is also the conclusion of a modal rule in T, whence

$$\hat{\iota}_v = \bigwedge\Gamma \text{ where } \Gamma = L \cup \left\{\Diamond\iota_{v_i} \;\middle|\; i < n\right\} \cup \left\{\Box(\iota_{v_0} \vee \cdots \vee \iota_{v_n})\right\}$$

where $v_0, \ldots, v_n$ enumerate the successors of v (in T) with v_n the minor premise and L is some set of literals. If the rule at u is the left modal rule ($\Diamond\mathsf{L}$), this rule is carried over to $D_{u,v}$ with the premise u^+ being matched to the major premise v_i such that $\Gamma_{u^+} \subset \Phi_{v_i}$. If, however, the rule at u is the right modal rule ($\Box\mathsf{R}$), then we appeal to the fact that $\Gamma_{u^+} \subset \Phi_{v_i}$ for all $i \leq n$ to form the disjunction $\bigvee\{\hat{\iota}_{v_0}, \ldots, \hat{\iota}_{v_n}\}$ before introducing the modal operator. The two cases each induce one of the following derivations for $D_{u,v}$:

$$\dfrac{\dfrac{\dfrac{}{\hat{\iota}_{v_i} \Rightarrow \Delta_{u^+}}\,\mathsf{b}}{\ldots, \Diamond\hat{\iota}_{v_i}, \ldots \Rightarrow \Delta_u}\Diamond\mathsf{L}}{\hat{\iota}_v \Rightarrow \Delta_u}\bigwedge\mathsf{L} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\hat{\iota}_{v_0} \Rightarrow \Delta_{u^+}}\,\mathsf{b} \quad \cdots \quad \dfrac{}{\hat{\iota}_{v_n} \Rightarrow \Delta_{u^+}}\,\mathsf{b}}{\hat{\iota}_{v_0} \vee \cdots \vee \hat{\iota}_{v_n} \Rightarrow \Delta_{u^+}}\bigvee\mathsf{L}}{\ldots, \Box(\hat{\iota}_{v_0} \vee \cdots \vee \hat{\iota}_{v_n}) \Rightarrow \Delta_u}\Box\mathsf{R}}{\hat{\iota}_v \Rightarrow \Delta_u}\bigwedge\mathsf{L}$$

With all cases of $D_{u,v}$ defined, let E be the derivation with root $\iota_{(T,S)} \Rightarrow \Delta$ given by repeatedly pasting at every bud the next applicable partial derivation $D_{u,v}$. This derivation has preserved all companion rules from D_ω. Cutting off each branch at the first repeated companion completes the construction. ◀

5.2 Uniform Interpolation and Applications

From the two template lemmas, we obtain our first uniform interpolation result.

▶ **Theorem 5.5** (Π_1 Uniform Interpolation). *For every signature S and formula $\alpha \in \Pi_1 \cup \Sigma_1$ there exists $\iota \in \Pi_1$ satisfying*

1. $\mathrm{sig}(\iota) \subset S \cap \mathrm{sig}(\alpha)$ *and* $|\iota| \leq 2^{|\mathsf{FL}(\alpha)|+1}$.
2. $\vdash \alpha \Rightarrow \iota$.
3. *for all* $\beta \in \Pi_1$ *with* $\mathrm{sig}(\beta) \subset S$, *if* $\vdash \alpha \Rightarrow \beta$ *then* $\vdash \iota \Rightarrow \beta$.

As we are targeting a Π_1 interpolant it is natural to consider the candidate $\iota_{(T,\sigma,S)}$ for an α-template in which every companion is marked ν. This choice guarantees condition 2 by Lemma 5.3 as right cedents in the generated cyclic derivation always comprise Π_1 formulas. This candidate, however, will not necessarily satisfy the third condition. It will fail, for example, if $\alpha = \mu x \Diamond x$: The natural template for α induces the Π_1 candidate $\iota = \nu x(\Diamond x \wedge \Box(\alpha \vee \top))$, which is equivalent to $\nu x \Diamond x$. Letting $\beta = \bot$, we have $\vdash \alpha \Rightarrow \beta$ but not $\vdash \iota \Rightarrow \beta$ as the latter sequent is not valid. The construction fails in this case because the sequent α is unsatisfiable, yet ι is satisfiable. Selecting $\bot$ for ι, indeed, provides a trivial uniform interpolant.

The solution is to 'correct' the candidate interpolant by pre-emptively setting the subformula assigned to $v \in T$ to be $\bot$ if $\Phi_v \Rightarrow$ is provable. Recall that provability can be checked from within the template alone by Theorem 3.11. We did not incorporate this step into Construction 2 because it is only necessary for constraining the uniform candidate to a particular quantifier class.

Proof. Fix a signature S and formula $\alpha \in \Pi_1 \cup \Sigma_1$. Let T be an α-template in which buds are the first repeat on each path and let $\sigma\colon \mathrm{comp}(T) \to \{\nu\}$ be the trivial marking assigning ν to all companions. By construction, $\iota_{(T,\sigma,S)} \in \Pi_1$ and $|\iota_{(T,\sigma,S)}| \leq 2^{|\mathsf{FL}(\alpha)|+1}$. Let Φ_u be the left cedent of $u \in T$ and $\hat{\iota}_u$ be the uniform candidate assigned to $u \in T$. We choose ι to be the modification of $\iota_{(T,\sigma,S)}$ given by replacing any subformula ι_u such that $\Phi_u \Rightarrow$ is provable by the formula $\bot$. This change can be carried out by inspection of the template alone (cf. Theorem 3.11). Clearly, $\iota \in \Pi_1$ and $\mathrm{sig}(\iota) \subset S \cap \mathrm{sig}(\alpha)$ and $|\iota| \leq |\iota_{(T,\sigma,S)}|$.

The First Template Lemma provides a cyclic derivation of $\alpha \Rightarrow \iota_{(T,\sigma,S)}$ in which all buds are sequents of the form $\Phi_u \Rightarrow \hat{\iota}_u$ for $u \in \mathrm{bud}(T)$. As each such formula $\hat{\iota}_u$ is (genuine) Π_1 all buds are right successful. The corresponding cyclic derivation of $\alpha \Rightarrow \iota$ is also a proof because if a bud is (now) labelled by $\Phi_u \Rightarrow \bot$ it is because $\Phi_u \Rightarrow$ is provable. Thus, $\vdash \alpha \Rightarrow \iota$.

For the final condition, fix $\beta \in \Pi_1$ and suppose D is a cyclic proof of $\alpha \Rightarrow \beta$. Consider the cyclic derivation of $\iota_{(T,\sigma,S)} \Rightarrow \beta$ described by the Second Template Lemma. A corresponding cyclic derivation D' of $\iota \Rightarrow \beta$ is readily obtained by simply closing every sequent with $\bot$ in the left cedent as an initial sequent. We claim that D' is a cyclic proof. Let $u \in \mathrm{bud}(D')$ be a bud of D' which is not right successful. As β is Π_1, the right cedent of u must be empty. Let $v \leq u$ be the first vertex on the path to u with empty right cedent. This vertex corresponds to a vertex $v' \in D$ of the form $\Gamma' \Rightarrow$. As D is a proof, the definition of ι means that the sequent at v in D' happens to be $\bot \Rightarrow$. ◀

Because the definition of signature takes into account polarities of variables, Theorem 5.5 actually establishes uniform *Lyndon* interpolation for the class of Σ_1 formulas:

▶ **Corollary 5.6.** *The Π_1 fragment of the modal μ-calculus has the right uniform Lyndon interpolation property. By duality, the Σ_1 fragment has the left uniform Lyndon interpolation property.*

▶ **Corollary 5.7.** *Every $\alpha \in \Sigma_1$ can be computably associated a formula $\alpha_\Pi \in \Pi_1$ with $|\alpha_\Pi| \leq 2^{|\mathsf{FL}(\alpha)|+1}$ such that $\alpha \in \Delta_1$ iff $\alpha \equiv \alpha_\Pi$.*

Proof. Fix $\alpha \in \Sigma_1$ and let $S = \mathrm{Lit}$ be the universal signature. Set α_Π to be the uniform interpolant for α given by Theorem 5.5. In particular, $\langle\!|\alpha|\!\rangle \subset \langle\!|\alpha_\Pi|\!\rangle$ by soundness. Since $\alpha_\Pi \in \Pi_1$, the right to left direction is immediate. So suppose $\alpha \equiv \beta$ for $\beta \in \Pi_1$. Then $\vdash \alpha \Rightarrow \beta$ by completeness and $\vdash \alpha_\Pi \Rightarrow \beta$ by Theorem 5.5. So $\alpha \equiv \alpha_\Pi$. ◀

In the previous section we confirmed, via the construction of Lyndon interpolants, that every Δ_1 formula is equivalent to a quantifier-free formula (Corollary 4.11). Combining that observation with the characterisation of Δ_1 above yields a decidability result first observed by Martin Otto [25].

▶ **Corollary 5.8** (Otto's Theorem for Σ_1)**.** *The problem of determining whether an arbitrary Σ_1-formula is equivalent to a quantifier-free formula is decidable.*

We will obtain the full version of Otto's Theorem, lifting the restriction to Σ_1, as a corollary of interpolation for formulas of arbitrary complexity, treated in the next section.

6 Cyclic Proofs and Interpolation

We present a general notion of cyclic proof that is sound and complete for all sequents in L_μ. The point of departure from the account in Definition 3 is that notion of proof can longer be expressed by a local condition on bud/companion pairs. Rather, proofhood in the general case is a *global* property determined by a condition on the *infinite* paths through a cyclic derivation.

The next definition expresses the idea of following the (potentially infinite) decomposition of a formula through a cyclic derivation.

▶ **Definition 6.1.** *Let D be a cyclic derivation. A* left *(resp.* right*)* occurrence *in D is a pair $f = (\alpha, v)$ where v is a vertex of D and α is a formula in the left (right) cedent at v. A* left trace *in D is an infinite sequence $t = (\alpha_i, v_i)_{i<\omega}$ of left occurrences such that for all $i < \omega$,*

1. *$(v_i)_i$ is a path through D,*
2. *if α_i is principal at v then α_{i+1} is an active formula at v_{i+1},*
3. *if α_i is not principal and v_i is not a modal rule then $\alpha_{i+1} = \alpha_i$,*
4. *if α_i is not principal and v_i is a modal rule then $\Box\alpha_{i+1} = \alpha_i$.*

A sequence of right occurrences fulfilling the same criteria but with $\Diamond$ *in place of* $\Box$ *in the final clause is called a* right trace. *The path* $(v_i)_i$ *is called the* carrier *of* t.

A simple argument, appealing to (weak) König's Lemma, shows that every path through a cyclic derivation is the carrier of at least one (left or right) trace. A cyclic derivation, recall, must contain a modal rule between every bud-companion pair. As a consequence, the formulas along a trace never stabilise:

▶ **Lemma 6.2.** *Let* $t = (\alpha_i, v_i)_{i<\omega}$ *be a left or right trace. The sequence of formulas* $(\alpha_i)_i$ *is a progressing thread.*

We can now isolate the cyclic proofs.

▶ **Definition 6.3** (Good trace)**.** *Let* t *be a (left or right) trace through a cyclic derivation and let* τ *be the progressing thread carried by* t. *We call* t good *if it is a left trace and* τ *is* μ *or a right trace and* τ *is* ν.

▶ **Definition 6.4** (Cyclic proof)**.** *A cyclic derivation* D *is a* cyclic proof *iff every infinite path through* P *is the carrier of a good trace.*

Definition 6.4 agrees with the previous notion of cyclic proofs for endsequents in the first quantifier level: An infinite path through a cyclic proof according to Definition 3.5 must be such that either every vertex along it is left-successful, or every vertex is right-successful. Weak König's Lemma implies that the path carries either a left trace of genuine Σ_1 formulas or a right trace of genuine Π_1 formulas. Both examples are good traces. In the general setting, checking whether a cyclic derivation fulfils the condition of Definition 6.4 is decidable and, for many logics including the modal μ-calculus, is PSPACE-complete [6].

Soundness and completeness of cyclic proofs is established by the same arguments as before observing that the connection between paths through derivations and strategies extends to one between traces and plays. Soundness in particular is a simple refinement of the proof of Proposition 3.9.

▶ **Lemma 6.5** (Duality)**.** $\vdash \Gamma \Rightarrow \Delta$ *iff* $\vdash \Rightarrow \Delta, \Gamma^{\perp}$, *iff* $\Gamma, \Delta^{\perp} \Rightarrow$.

▶ **Theorem 6.6** (Soundness)**.** $\Gamma \Rightarrow \Delta$ *is valid if* $\vdash \Gamma \Rightarrow \Delta$.

Proof. Suppose D is a cyclic derivation of $\Gamma \Rightarrow$ and S is a winning Verifier strategy in $G(\bigwedge\Gamma, F)$. The proof of Proposition 3.9 determines an infinite path $P = (u_i)_i$ through D such that every left trace carried by P can be associated a play consistent with S that carries the same thread ignoring consecutive repetitions. As S is winning for Verifier, P does not carry a good trace, hence D is not a cyclic proof. ◀

Completeness is not so straightforward. Although the definition of a template applies in the general setting, without an analogue of the *unsuccessful* vertices a lifting of Proposition 3.11 is not immediately available. We return to completeness at the end of Section 6.1.

6.1 A Normal Form for Cyclic Derivations

In order to ensure the candidate constructions from previous sections work at higher levels of the quantifier hierarchy, it is necessary to impose constraints on the structure of cyclic proofs. This is accomplished through a normal-form theorem for cyclic derivations that shows that every cyclic proof can be represented as a marked derivation where the marking encapsulates the existence of good threads. The use of marked derivations in this way first appeared in [15] (conceived of as a 'colouring' of companions). Our use of them is more abstract and motivated by properties of cyclic systems known as *reset systems* developed in [1, 4, 16, 30, 19]. A normal form theorem similar to the one below is implicit in the syntactic proofs of Lyndon and uniform interpolation in [3, 4].

Let P be a path through a cyclic derivation. By $\mathrm{Inf}(P)$ we denote the set of vertices that appear infinitely often along P:

$$\mathrm{Inf}(P) = \{u \in D \mid u = P(i) \text{ for infinitely many } i\}.$$

The set $\mathrm{Inf}(P)$ is non-empty and has a least element, which must be a companion. The least vertex is denoted $\min(P)$.

▶ **Theorem 6.7** (Normal form theorem). *For every cyclic derivation in* $\mathsf{K}\mu$ *or the template calculus there exists a cyclic derivation* D *in the same calculus with the same unravelling which admits a marking* σ *such that any path* P *through* D *carries a good left trace iff* $\sigma(\min(P)) = \mu$.

The theorem is due to Janin and Walukiewicz who proved the result for a variation of our sequent calculus $\mathsf{K}\mu$ [15, Lemma 4.5]. Their proof appeals to standard results in automata theory (see, e.g., [14]). From a fixed cyclic derivation (or template) D one obtains, via McNaughton's theorem [23], a deterministic parity stream automaton $\mathcal{A}$ over the alphabet of sequents and rules which accepts an infinite sequence in the alphabet iff the sequences enumerates a path through D and the path carries a good left trace. Running $\mathcal{A}$ against the unravelling of D gives an assignment of states to vertices in D_ω which can be refolded into cyclic derivation with the desired property.

A completely syntactic proof of the normal form theorem can be given by elaborating the sequent calculus for $\mathsf{K}\mu$. Fix a countably infinite set of symbols, called names. Each sequent in D and formula within the sequent is annotated by a finite sequence of names recording the quantifiers encountered in the finite trace from the root to the formula. Aside from the logical rules inherited from D a new structural rule – the *reset* rule – is added which, read bottom-up, discards superfluous information from all annotations in the sequent. Sequent calculi incorporating annotations and a reset mechanism are called *reset systems*. The reset rule is the main component which ensures that the unravelling of D can be annotated by employing only finitely many names and finitely many distinct annotated sequents. As the annotation mechanism is deterministic, the annotated form of D_ω can be realised as the unravelling of a cyclic derivation in the reset calculus which validates Theorem 6.7. A reset

calculus for the modal μ-calculus was introduced in [16] for satisfiability and dualised to validity in [30]. The connection to normal forms was realised in [2, Thm IV.10].

The automata- and proof-theoretic approaches to the normal form theorem are closely related. The first reset calculus for the modal μ-calculus was inspired by the structural operations underpinning the Safra construction for determinising stream automata [16]. Alternative reset systems arise from considering different determinisation algorithms for Büchi and parity automata [12]. A general method of associating reset systems to abstract notion of cyclic derivation is explored in [19].

▶ **Definition 6.8.** *A marked derivation which satisfies the property in Theorem 6.7 is said to be in* normal form *or, simply,* normal.

In a normal cyclic derivation, the existence of good left traces on a path is determined by the marking of the lowest companion that occurs infinitely often. If a companion v is marked by μ then *every* path that visits v infinitely often and companions below v only finitely often is the carrier of a good left trace. If v is marked by ν then *no* such path carries a good left trace.

The normal form property does not state anything about *right* traces. Under the knowledge that the derivation is a proof a symmetric condition arises:

▶ **Corollary 6.9.** *Suppose D is a normal cyclic proof. Then any path P through D carries a good right trace iff $\sigma(\min(P)) = \nu$.*

For endsequents from $\Sigma_1 \cup \Pi_1$, the normal form theorem is vacuous. Given a derivation of a sequent $\Gamma \Rightarrow \Delta$ in this fragment, mark a companion u as μ iff the left cedent at u contains a genuine Σ_1 formula. Then (D, σ) is in normal form. Normal derivations at the first level have the further property that the marking is constant along path segments: if $u \lhd_D v$ for $u, v \in \mathrm{comp}(D)$ then $\sigma(u) = \sigma(v)$. Lemma 4.4 shows that this property alone is sufficient to ensure that the Lyndon candidate constructed from D is in the class $\mathit{Clos}(\Sigma_1 \cup \Pi_1)$.

The normal form theorem for templates is sufficient to prove completeness, starting with a lifting of Proposition 3.11:

▶ **Proposition 6.10.** *Let T be a normal template for Γ. If $\Gamma \Rightarrow$ is valid then T admits a pruning to a cyclic proof.*

Proof. Let $\Gamma \Rightarrow$ be any sequent and (T, σ) a template for Γ in normal form. Say that a bud of T is *successful* iff its companion is marked μ, and a non-bud vertex is *successful* if it is either a modal rule and a major premise is successful, or a non-modal rule and all premises are successful. Let U be the set of unsuccessful vertices.

We assume that $\Gamma \Rightarrow$ does not admit a cyclic proof. In particular, no cyclic derivation can be witnessed as a subtree of the successful vertices in T as, by the definition of normal form, any such derivation would be a proof. Hence, U contains the root of T. From U we recursively construct a frame $\langle F, r\rangle$ and Verifier strategy S in $G(\bigwedge\Gamma, F)$ with the property

that every infinite play consistent with S corresponds to a left trace through some path of U. By the choice of U, all left traces carry ν-threads, so S is winning for Verifier. ◀

Completeness and decidability is a corollary of Theorem 6.7 and Proposition 6.10:

▶ **Theorem 6.11** (Completeness). *If $\Gamma \Rightarrow \Delta$ is valid, then $\vdash \Gamma \Rightarrow \Delta$.*

As the construction of normal forms is computable, Proposition 6.10 implies

▶ **Corollary 6.12.** *Provability in $\mathsf{K}\mu$ is decidable.*

6.2 Lyndon Interpolation for the Full Language

The normal form theorem provides a direct path to Lyndon interpolants:

▶ **Theorem 6.13** (Lyndon Interpolation for μ-calculus). *Suppose $\vdash \alpha \Rightarrow \beta$ for $\alpha, \beta \in \mathsf{L}_\mu$. There exists a formula ι such that*
1. $\mathrm{sig}(\iota) \subset \mathrm{sig}(\alpha) \cap \mathrm{sig}(\beta)$,
2. $\vdash \alpha \Rightarrow \iota$ *and* $\vdash \iota \Rightarrow \beta$.

Proof. Fix a marked normal cyclic proof (D, σ) and let ι be the induced Lyndon candidate following Construction 1. Condition 1 of the theorem holds by definition. We outline the argument for $\vdash \alpha \Rightarrow \iota$, with the other half being symmetric. Let $\Gamma_u \Rightarrow \Delta_u$ be the sequent labelling $u \in D$.

By Lemma 4.6, $\vdash \alpha \Rightarrow \iota$ admits a cyclic derivation D' where buds (and companions) are the sequents $\Gamma_w \Rightarrow \hat{\iota}_w$ for $w \in \mathrm{comp}(D)$. The proof of Proposition 4.7 presents D' as, essentially, the cyclic derivation D with the right cedent at $w \in D$ replaced by the formula $\hat{\iota}_w$. Any path P through D' corresponds to a path through D that carries exactly the same left traces. Let C be the companions of D visited infinitely often by that path. Furthermore, P carries exactly one right trace, whose infinitely occurring quantified formulas are precisely $\{\hat{\iota}_w \mid w \in C\}$. The shortest such formula is, by design, $\hat{\iota}_u$ for $u = \min P$. If P does not carry a good left trace, then $\sigma(u) = \nu$ and $\hat{\iota}_u = \nu x_u \zeta$ for some ζ. But then P carries a good right trace. ◀

We can establish a refinement of the result in case of the second quantifier level. For this we require an observation:

▶ **Proposition 6.14.** *Let $\Gamma \subset \Pi_2$ and $\Delta \subset \Sigma_2$. If (D, σ) is a normal proof of $\Gamma \Rightarrow \Delta$ then for all $u, v \in \mathrm{comp}(D)$, if $u \lhd_D v$ then $\sigma(u) = \sigma(v)$.*

Proof. Let $u, v \in \mathrm{bud}(D)$ such that $u^{\mathsf{c}} < v^{\mathsf{c}} < u$. Suppose $\sigma(u^{\mathsf{c}}) = \nu$. Let k exceed the number of formulas in the left cedent at v and consider the path P which alternates between the bud u and $K + 1$ cycles at v:

$$P \coloneqq u^{\mathsf{c}} \ldots \underbrace{v v^{\mathsf{c}} \ldots v v^{\mathsf{c}}}_{k} \ldots u u^{\mathsf{c}} \ldots \underbrace{v v^{\mathsf{c}} \ldots v v^{\mathsf{c}}}_{k} \ldots u u^{\mathsf{c}} \ldots$$

By design, $\min P = u^{\mathsf{c}}$. Since $\sigma(u^{\mathsf{c}}) = \nu$, P carries a good right trace. From some point on, this trace never encounters the quantifier μ as the main connective. Also, by the choice of k, the trace must repeat a formula at v before the path returns next to u. As a consequence, the path which cycles between v^{c} and v only also carries a good right trace and $\sigma(v^{\mathsf{c}}) = \nu$ by normal form. The case of $\sigma(u^{\mathsf{c}}) = \mu$ is symmetric. ◀

The proof above breaks if $\Gamma \not\subset \Pi_2$: the good (right) trace carried by P could, for example, encounter a formula $\nu x\, \alpha$ in each path segment $u^{\mathsf{c}} \dots v$ and strictly longer formulas $\mu y\, \beta$ within the 'local' cycles $v^{\mathsf{c}} \dots v$. In that case the marking would satisfy $\sigma(u) = \nu$ but $\sigma(v) = \mu$.

▶ **Theorem 6.15** (Lyndon interpolation at the second level). *Suppose that* $\vdash \alpha \Rightarrow \beta$ *for* $\alpha \in \Pi_2$ *and* $\beta \in \Sigma_2$. *There exists an alternation-free formula* ι *such that*

1. $\mathrm{sig}(\iota) \subset \mathrm{sig}(\alpha) \cap \mathrm{sig}(\beta)$,
2. $\vdash \alpha \Rightarrow \iota$ *and* $\vdash \iota \Rightarrow \beta$.

Proof. By Corollary 6.9 we may assume a normal proof (D, σ) of $\alpha \Rightarrow \beta$. The associated Lyndon candidate $\iota_{(D,\sigma)}$ is, by the proof of Theorem 6.13, a Lyndon interpolant for $\alpha \Rightarrow \beta$. Lemma 4.4 and Proposition 6.14 show that $\iota_{(D,\sigma)} \in \mathit{Clos}(\Pi_1 \cap \Sigma_1)$. ◀

Interpolation for the second level has two interesting corollaries. The first result was shown by Marti and Venema [22] via a novel annotated cyclic calculus for alternation-free fragment.

▶ **Corollary 6.16.** *The alternation-free fragment of* L_μ *admits Lyndon interpolation.*

The second corollary is a modal rendering of Rabin's definability theorem for weak monadic second order logic [27], originally transferred to the μ-calculus by Arnold and Niwiński [7].

▶ **Corollary 6.17.** *Every* Δ_2 *formula is equivalent to an alternation-free formula in the same signature. In particular,* $\Delta_2 = (\Pi_2 \cap \Sigma_2)^{\equiv}$.

We conclude our voyage into Lyndon interpolation with two results on the higher ambiguous classes. Both results are due to Arnold and Santocanale [28]. The first is a negative result:

▶ **Theorem 6.18.** *For every* $n > 2$, $(\Pi_n \cup \Sigma_n)^{\equiv} \subsetneq \Delta_n$.

The proof of Theorem 6.18 is beyond the scope of this exposition. The second result from [28] is a refinement of Theorem 6.15 that can be lifted to higher levels (see Corollary 6.21 below). The theorem appeals to an important syntactic fragment of L_μ.

▶ **Definition 6.19** (Disjunctive formula). *A formula* α *is* disjunctive *if for every conjunction* $\bigwedge\Gamma \in \mathsf{FL}(\alpha)$ *there exists* $L \subset \mathrm{Lit}$ *and* $\Delta \subset \Theta \subset \mathsf{FL}(\alpha)$ *such that* $\Gamma = L \cup \Diamond\Delta \cup \Box\bigvee\Theta$. *The set of disjunctive formulas is denoted* $\mathbb{D}$.

In disjunctive formulas, the use of conjunction is restricted to specific arrangements of modal statements. The unary modalities can be expressed disjunctively as $\Diamond\alpha \equiv \bigwedge\{\Diamond\alpha, \Diamond\top, \Box\bigvee\{\alpha, \top\}\}$ and $\Box\alpha \equiv \Box\bigvee\{\alpha\}$. The notion was introduced by Janin and Walukiewicz [15] who showed that every formula in the modal μ-calculus is equivalent to a disjunctive formula. Definition 6.19 is more permissive than the standard presentation which is obtained by requiring $\Theta = \Delta$. Our formulation has better proof-theoretic properties [5] and can be reduced to the Janin–Walukiewicz definition by noting that, for $\Delta \subset \Theta$ and $\Delta' = \Delta \cup \{\bigvee(\Theta \setminus \Delta)\}$,

$$\bigwedge\Diamond\Delta \wedge \Box\bigvee\Theta \equiv \left(\bigwedge\Diamond\Delta \wedge \Box\bigvee\Delta\right) \vee \left(\bigwedge\Diamond\Delta' \wedge \ldots\right).$$

We have already seen examples of disjunctive formulas: the uniform candidates in Construction 2. We explore the consequences of that observation in the next section.

▶ **Theorem 6.20** (Lyndon Interpolation for Disjunctive Sequents). *Suppose* $\vdash \alpha \Rightarrow \beta$ *where* $\alpha, \beta^\perp \in \Pi_{n+1} \cap \mathbb{D}$. *There exists* $\iota \in \mathit{Clos}(\Pi_n \cup \Sigma_n)$ *such that*

1. $\mathrm{sig}(\iota) \subset \mathrm{sig}(\alpha) \cap \mathrm{sig}(\beta)$,
2. $\vdash \alpha \Rightarrow \iota$ *and* $\vdash \iota \Rightarrow \beta$.

Proof sketch. The structure of disjunctive formulas is such that a cyclic proof of $\alpha \Rightarrow \beta$ with α and $\beta^\perp$ both disjunctive can be transformed into one in which every bud contains at most one formula in each cedent. We may further assume that the cyclic proof is normal and utilises conservative instances of rules only, in the sense of the proof of Proposition 3.11.

From the observations above and the assumption that $\alpha \in \Pi_{n+1}$ and $\beta \in \Sigma_{n+1}$ we deduce that in every chain of companions $u_0 \lhd u_1 \lhd \cdots \lhd u_k$ the sequence of markings $\sigma(u_0), \ldots, \sigma(u_k)$ has fewer than n alternations, whence the Lyndon interpolant induced by the cyclic proof is in $\mathit{Clos}(\Pi_n \cup \Sigma_n)$. ◀

▶ **Corollary 6.21** (Collapse of Disjunctive-Δ_n). *Suppose* $\alpha \in \mathbb{D} \cap \Pi_{n+1}$. *If* $\alpha^\perp$ *is equivalent to a formula in* $\mathbb{D} \cap \Pi_{n+1}$, *then* $\alpha \in \mathit{Clos}(\Sigma_n \cup \Pi_n)^{\equiv}$.

6.3 Uniform Interpolation for the Full Language

The normal form theorem for templates (Theorem 6.7) also offers a means to prove uniform interpolation for the full language of the modal μ-calculus:

▶ **Theorem 6.22** (Uniform Lyndon Interpolation). *For every signature* S *and formula* $\alpha \in \mathsf{L}_\mu$ *there exists a formula* ι *such that*

1. $\mathrm{sig}(\iota) \subset \mathrm{sig}(\alpha) \cap S$.
2. $\vdash \alpha \Rightarrow \iota$.
3. *For all* β, *if* $\mathrm{sig}(\beta) \subset S$ *and* $\vdash \alpha \Rightarrow \beta$, *then* $\vdash \iota \Rightarrow \beta$.

Proof. Fix S and α, and an α-template T. By Theorem 6.7 we may assume that (T, σ) is in normal form. Let $\iota = \iota_{(T,\sigma,S)}$ be the induced uniform candidate with $\hat{\iota}_u$ the candidate associated to $u \in T$. By design, $\mathrm{sig}(\iota) \subset \mathrm{sig}(\alpha) \cap S$. Lemmas 5.3 and 5.4 describe the cyclic derivations that witness conditions 2 and 3 and it suffices to confirm that these derivations are proofs. Concerning 2, recall that a cyclic derivation D_T of $\alpha \Rightarrow \iota$ is obtained by installing $\hat{\iota}_u$ as the right cedent of $u \in T$ and interspersing the derivation with right rules as needed. Let P be any path in this derivation and with $u = \min P$ the minimal companion visited infinitely often. Suppose that P does not carry a good left trace. So, $\sigma(u) = \nu$. But then $\hat{\iota}_u = \nu x_u \zeta$ for some ζ and, by design, $\hat{\iota}_u$ is the shortest formula along the single right trace carried by P. Hence, D_T is a cyclic proof.

The argument towards 3 is similar. Let $\mathrm{sig}(\beta) \subset S \cap \mathrm{sig}(\alpha)$. The proof of Lemma 5.4 describes the construction of a cyclic derivation D' with endsequent $\iota \Rightarrow \beta$ from a cyclic proof D of $\alpha \Rightarrow \beta$. Consider a path P through D'. The construction of D' was such that P is associated a path P_D through D and a path P_T through T such that (i) P and P_D carry the same right traces, (ii) P carries a good left trace if P_T does, and (iii) P_T carries every left trace of P_D. As D is a cyclic proof, P must carry a good trace. ◀

The candidate uniform interpolants obtained from Construction 2 are all disjunctive formulas (cf. Definition 6.19). An immediate corollary of our uniform interpolation theorem is the celebrated *disjunctive normal form theorem* for the modal μ-calculus, established in [15]:

▶ **Corollary 6.23** (Disjunctive normal form theorem)**.** *Every formula of the modal μ-calculus has an equivalent disjunctive formula in the same signature.*

Proof. Apply Theorem 6.22 for $S = \mathrm{sig}(\alpha)$. Then $\vdash \alpha \Rightarrow \iota$ (condition 2) and $\vdash \iota \Rightarrow \alpha$ (by condition 3). ◀

Corollary 5.7 gave a characterisation of the Σ_1 formulas that are equivalent to a Π_1 formula and, by extension, the Σ_1 formulas equivalent to quantifier-free formulas. The template approach to uniform interpolants lifts the characterisation to the whole calculus. The first step is a characterisation of Π_1-equivalence due to Küsters and Wilkie [18]. As a second corollary we obtain Otto's theorem on the decidability of quantifier-free equivalence [25].

▶ **Theorem 6.24** (Π_1-characterisation)**.** *For every $\alpha \in \mathsf{L}_\mu$ there is an effectively computed $\alpha_\Pi \in \Pi_1$ such that $\alpha \in \Pi_1^{\equiv}$ iff $\alpha \equiv \alpha_\Pi$.*

Proof. Let $S = \mathrm{Lit}$ be the universal signature. We carry out the proof of Theorem 6.22 with modification to candidates used in Theorem 5.5, from which the claim is an immediate consequence. ◀

▶ **Corollary 6.25** (Π_1-decidability)**.** *The problem of determining whether an arbitrary formula of the modal μ-calculus is equivalent to a Π_1 formula is decidable.*

Decidability of quantifier-free definability now extends to the full language:

▶ **Corollary 6.26** (Otto's theorem). *The problem of determining whether an arbitrary formula of the modal μ-calculus is equivalent to a quantifier-free formula is decidable.*

Proof. Define $\alpha_\Sigma = \neg(\neg\alpha)_\Pi \in \Sigma_1$. By the results above, $\langle\!\langle \alpha_\Sigma \rangle\!\rangle \subset \langle\!\langle \alpha \rangle\!\rangle \subset \langle\!\langle \alpha_\Pi \rangle\!\rangle$ and $\langle\!\langle \alpha \rangle\!\rangle$ is definable by a quantifier-free formula iff $\vdash \alpha_\Pi \Rightarrow \alpha_\Sigma$. ◀

The final result we present is an analogue to the automata-theoretic result stating that non-deterministic Büchi tree automata accept the same languages as alternating Büchi automata. From the perspective of interpolation, the result shows that the Lyndon interpolation theorems for the first and second quantifiers (Theorems 4.8 and 6.15) reduce to our interpolation theorem for disjunctive formulas (Theorem 6.20).

▶ **Theorem 6.27.** *Every Π_1 and Π_2 formula is equivalent to a Π_1 and Π_2 disjunctive formula respectively.*

Proof. We show the Π_2 case. Fix $\alpha \in \Pi_2$ and let (T, σ) be a normal α-template. Let ι be the uniform candidate for α with respect to $\mathrm{sig}(\alpha)$. In particular, $\iota \equiv \alpha$. As in Proposition 6.14, we have that for all $u \lhd v \in \mathrm{comp}(T)$, if $\sigma(u) = \mu$ then $\sigma(v) = \mu$. Thus, $\iota \in \Pi_2 \cap \mathbb{D}$. ◀

7 Discussion

From any cyclic derivation of a sequent $\alpha \Rightarrow \beta$, the cyclic Maehara method yields an explicit formula ι in the signature common to both α and β, and cyclic derivations of $\alpha \Rightarrow \iota$ and $\iota \Rightarrow \beta$. If the initial derivation happens to be a cyclic proof and in normal form then the resulting derivations are guaranteed to be cyclic proofs and ι a Lyndon interpolant of $\alpha \Rightarrow \beta$.

The structure of the interpolating formula is completely determined by the initial cyclic derivation of $\alpha \Rightarrow \beta$. In selected cases, sufficient bounds on the complexity of the interpolant can be given. If α and β both sit in the first non-trivial level of the quantifier hierarchy, a candidate interpolant can always be selected from the same level (Theorems 4.8 through 4.14). If α is Π_2 and β is Σ_2, the candidate formed from a normal proof will be alternation-free, i.e., in $\Pi_2 \cap \Sigma_2$ (Theorem 6.15). In particular, the alternation-free fragment of L_μ has the Lyndon interpolation property (Corollary 6.16) and is the largest level of the quantifier hierarchy with this property (Theorem 6.18; [28]). Similar observations, both positive and negative, can be made for the designation of uniform interpolants (e.g. Theorems 5.5 and 6.27).

The question naturally arises whether the syntactic method can be applied also to fragments of the μ-calculus corresponding to specific fixed point logics. In such settings the quantifier complexity of the induced interpolant is less important than the issue of whether the constructed interpolant can be expressed by (or adjusted to be) a formula in the chosen fixed point logic, as the latter is clearly required to conclude the Lyndon/uniform interpolation property. Gödel–Löb provability logic GL is one example in which the gap between construction and logic can be bridged. Shamkanov's syntactic proof of Lyndon interpolation [29] stems from the realisation of Gödel–Löb logic as a cyclic proof system over

the modal logic of transitive frames, K4. As the Maehara method supports the K4 modality, every GL-provable sequent admits a Lyndon interpolant in the μ-calculus over K4. Lyndon interpolation for GL is a corollary of the de Jong–Sambin–Bernardi fixed point theorem which establishes that the μ-calculus collapses over GL.

There are, however, many examples of logics that cannot internalise the structure of their cyclic proofs. Well-known fixed point logics that lack the Craig interpolation property include linear-time temporal logic, computation-tree logic [21] and the logic of common knowledge [31]. Each of these logics can be associated sound and complete analytic sequent calculi that support the cyclic Maehara construction of candidate interpolants. Lyndon interpolants for valid sequents are definable in the linear-time μ-calculus, multi-modal μ-calculus and μ-calculus over S5 respectively, but not always in the initial fixed point logic itself. This phenomenon was explored in detail in Chapter 10. A deeper analysis of the cyclic Maehara method can potentially contribute to characterising the necessary complexity of interpolants and algorithms for deciding their existence within specific fragments.

Acknowledgments

Both authors were supported by the Knut and Alice Wallenberg Foundation [2020.0199]. The first author was additionally supported by the Swedish Research Council [2022-01685] and the Dutch Research Council [OCENW.M20.048]. The authors would like to thank the referees for their suggestions, Sam van Gool and his students at ENS Saclay and Cárolos Laméris for eliminating many errors from this work.

References

1 Bahareh Afshari and Graham E. Leigh. Finitary proof systems for Kozen's μ. Oberwolfach preprints (owp) issn 1864-7596, Mathematisches Forschungsinstitut Oberwolfach, 2016.

2 Bahareh Afshari and Graham E. Leigh. Cut-free completeness for modal mu-calculus. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017.

3 Bahareh Afshari and Graham E. Leigh. Lyndon interpolation for modal μ-calculus. In Aybüke Özgün and Yulia Zinova, editors, *Language, Logic, and Computation*, pages 197–213, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-030-98479-3_10.

4 Bahareh Afshari, Graham E. Leigh, and Guillermo Menéndez Turata. Uniform interpolation from cyclic proofs: The case of modal mu-calculus. In Anupam Das and Sara Negri, editors, *Automated Reasoning with Analytic Tableaux and Related Methods. TABLEAUX 2021*, volume 12842 of *Lecture Notes in Computer Science*, Cham, 2021. Springer. doi:10.1007/978-3-030-86059-2_20.

5 Bahareh Afshari, Graham E. Leigh, and Guillermo Menèndez Turata. Demystifying μ. *Fundamenta Informaticae*, 194(2):12773, August 2025. doi:10.46298/fi.12773.

6 Bahareh Afshari and Dominik Wehr. Abstract cyclic proofs. *Mathematical Structures in Computer Science*, 34(7):552–577, August 2024. doi:10.1017/S0960129524000070.

7 André Arnold and Damian Niwiński. Fixed point characterization of Büchi automata on infinite trees. *J. Inf. Process. Cybern.*, 26(8-9):451–459, January 1991.

8 André Arnold and Damian Niwiński. *Rudiments of μ-calculus*, volume 146 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2001.

9 Julian Bradfield and Colin Stirling. 12 Modal mu-calculi. In *Studies in Logic and Practical Reasoning*, volume 3, pages 721–756. Elsevier, 2007. doi:10.1016/S1570-2464(07)80015-2.

10 Giovanna D'Agostino and Marco Hollenberg. Logical questions concerning the μ-calculus: Interpolation. Lyndon and Łos-Tarski. *The Journal of Symbolic Logic*, 65(1):310–332, 2000.

11 Christian Dax, Martin Hofmann, and Martin Lange. A proof system for the linear time μ-calculus. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, S. Arun-Kumar, and Naveen Garg, editors, *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *Lecture Notes in Computer Science*, pages 273–284. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. doi:10.1007/11944836_26.

12 Maurice Dekker, Johannes Kloibhofer, Johannes Marti, and Yde Venema. Proof systems for the modal μ-calculus obtained by determinizing automata. In Revantha Ramanayake and Josef Urban, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 242–259, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-43513-3_14.

13 Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, April 1979. Publisher: Elsevier BV. doi:10.1016/0022-0000(79)90046-1.

14 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. doi:10.1007/3-540-36387-4.

15 David Janin and Igor Walukiewicz. Automata for the modal μ-calculus and related results. In Jiří Wiedermann and Petr Hájek, editors, *Mathematical Foundations of Computer Science 1995: 20th International Symposium, MFCS '95 Prague, Czech Republic, August 28–September 1, 1995 Proceedings*, pages 552–562, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

16 Natthapong Jungteerapanich. A tableau system for the modal μ-calculus. In Martin Giese and Arild Waaler, editors, *Automated Reasoning with Analytic Tableaux and Related Methods, 18th International Conference, TABLEAUX 2009, Oslo, Norway, July 6-10, 2009. Proceedings*, volume 5607 of *Lecture Notes in Computer Science*, pages 220–234. Springer, 2009.

17 Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

18 Ralf Küsters and Thomas Wilke. Deciding the first level of the μ-calculus alternation hierarchy. In Manindra Agrawal and Anil Seth, editors, *Proceedings of the 22nd Conference Kanpur on Foundations of Software Technology and Theoretical Computer Science*. Springer-Verlag Berlin Heidelberg, December 2002.

19 Graham E. Leigh and Dominik Wehr. From GTC to Reset: Generating reset proof systems from cyclic proof systems. *Annals of Pure and Applied Logic*, 175(10):103485, December 2024. doi:10.1016/j.apal.2024.103485.

20 Shôji Maehara. Craig-interpolation theorem. *Sugaku*, 12(4):235–237, 1961.

21 Larisa Maksimova. Temporal logics of "the next" do not have the Beth property. *Journal of Applied Non-Classical Logics*, 1(1):73–76, January 1991. doi:10.1080/11663081.1991.10510772.

22 Johannes Marti and Yde Venema. Focus-style proof systems and interpolation for the alternation-free μ-calculus, April 2021. arXiv:2103.01671 [cs]. doi:10.48550/arXiv.2103.01671.

23 Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, October 1966. doi:10.1016/S0019-9958(66)80013-X.

24 Damian Niwinski and Igor Walukiewicz. Games for the mu-calculus. *Theoretical Computer Science*, 163(1&2):99–116, 1996.

25 Martin Otto. Eliminating recursion in the μ-calculus. In Gerhard Goos, Juris Hartmanis, Jan Van Leeuwen, Christoph Meinel, and Sophie Tison, editors, *STACS 99*, volume 1563, pages 531–540. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. Series Title: Lecture Notes in Computer Science. doi:10.1007/3-540-49116-3_50.

26 Andrew M. Pitts. On an interpretation of second order quantification in first order intuitionistic propositional logic. *J. Symb. Log.*, 57(1):33–52, 1992.

27 Michael O. Rabin. Weakly definable relations and special automata. In *Studies in Logic and the Foundations of Mathematics*, volume 59, pages 1–23. Elsevier, 1970. doi:10.1016/S0049-237X(08)71929-3.

28 Luigi Santocanale and André Arnold. Ambiguous classes in μ-calculi hierarchies. *Theoretical Computer Science*, 333(1-2):265–296, March 2005. doi:10.1016/j.tcs.2004.10.024.

29 Daniyar Shamkanov. Circular proofs for the Gödel-Löb provability logic. *Math. Notes*, 96:575–585, 2014.

30 Colin Stirling. A tableau proof system with names for modal mu-calculus. In Andrei Voronkov and Margarita V. Korovina, editors, *HOWARD-60: A Festschrift on the Occasion of Howard Barringer's 60th Birthday*, volume 42 of *EPiC Series in Computing*, pages 306–318. EasyChair, 2014.

31 Thomas Studer. Common knowledge does not have the Beth property. *Information Processing Letters*, 109(12):611–614, May 2009. doi:10.1016/j.ipl.2009.02.011.

Chapter 12

Interpolation with Automated First-Order Reasoning

Christoph Wernhard
University of Potsdam, Germany

Abstract

We consider interpolation from the viewpoint of fully automated theorem proving in first-order logic as a general core technique for mechanized knowledge processing. For Craig interpolation, our focus is on the two-stage approach, where first an essentially propositional ground interpolant is calculated that is then lifted to a quantified first-order formula. We discuss two possibilities to obtain a ground interpolant from a proof: with clausal tableaux, and with resolution. Established preprocessing techniques for first-order proving can also be applied for Craig interpolation if they are restricted in specific ways. Equality encodings from automated reasoning justify strengthened variations of Craig interpolation. Contributions to Craig interpolation that emerged from automated reasoning include variations for logics used in databases and logic programming. As an approach to uniform interpolation we introduce second-order quantifier elimination with examples and describe the basic algorithms DLS and SCAN.

Contents

1 Introduction 416
2 Notation and Preliminaries 419
3 Preprocessing: Skolemization, Clausification, Simplification 420
4 The Two-Stage Approach to Craig Interpolation in First-Order Logic . 424
5 Ground Interpolation with Clausal Tableaux 427
5.1 Clausal Tableaux – Proof Objects for Automated Reasoning 427
5.2 The Connection Tableau Calculus 428
5.3 The Regularity Restriction 430
5.4 The Hyper Property 430
5.5 Interpolant Calculation from a Clausal Tableau 432
5.6 The Two-Stage Approach: Overall Workflow 433



© Christoph Wernhard;
licensed under Creative Commons License CC BY 4.0
In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 12, pages 415–459. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.l

6 Ground Interpolation with Resolution 435
6.1 From a Deduction via a Deduction Tree to a Ground Deduction 435
6.2 Interpolant Calculation from a Ground Deduction 437
6.3 Deduction Trees and Clausal Tableaux 440
7 Craig-Lyndon Interpolation and Equality 442
8 Contributions of Automated Reasoning to Craig Interpolation 443
9 Second-Order Quantifier Elimination 445
9.1 Direct Methods – Ackermann's Lemma and the DLS Algorithm 447
9.2 Predicate Elimination with Resolution – The SCAN Algorithm 450
Acknowledgments 453
References 453

1 Introduction

In this chapter we approach interpolation from the viewpoint of fully automated theorem proving in first-order logic as a general core technique for mechanized knowledge processing. Craig interpolation [19, 20] fits quite naturally into this methodology centered around first-order proving: In first-order logic, if a formula entails another one, then there is a finite proof of this entailment. A Craig interpolant – a first-order formula that is semantically between the entailing formulas and syntactically within their shared vocabulary – can be calculated from the proof.

Automated reasoning gives us Craig interpolation in *mechanized* form. A first-order theorem prover searches for the underlying proof, which is then converted to an interpolant. This makes Craig interpolation available for practical applications. Novel variations where interpolants satisfy stronger syntactic constraints than just shared vocabulary can be explored with numerous and large problem instances. For proof search, we can benefit from decades of research in automated reasoning, manifested in the *Handbook of Automated Reasoning* [86] and conference series such as *Conference on Automated Deduction (CADE)* and *International Joint Conference on Automated Reasoning (IJCAR)*. We can utilize advanced highly-optimized systems and the *TPTP (Thousands of Problems for Theorem Provers) World* [94], a research infrastructure that includes a problem library, specified standard formats, software tools and data from prover evaluations.

Although taming proof search is a core objective of automated reasoning, it is less relevant for Craig interpolation, where we can start from a given proof, assuming it had already been found by some powerful system. However, a closer inspection of the regular winners of the annual *CADE ATP System Competition (CASC)*, *Vampire* [50] and *E* [89], reveals that in powerful configurations they do not output proof objects in some defined calculus. Applications that require actual proofs, such as hammers [13], which invoke automated systems on subproblems in an interactive setting, use a workflow where just lemmas are taken

from the powerful systems, to guide the search with a weaker system that builds proofs. A practical alternative is provided with *Prover9* [68], a fairly strong first-order prover that can output actual resolution proofs and represents the state of the art in about 2009.

Vampire, *E* and *Prover9* operate by maintaining clause sets, which grow through adding clauses inferred by calculi that can be described as resolution and equality processing with paramodulation or superposition [85, 84, 8]. Also vice-versa, equality inferences by superposition can be taken as basis, with resolution as a special case. There is a second tradition of fully automated provers, going back to Prawitz [80, 81], which operate by enumerating tableau-like proof structures in combination with unification of formulas. Model elimination [63], the connection method [11, 12], and clausal tableaux [55, 59, 57] are such methods. With respect to power for general proof search, this approach currently resides at the state of the art of around 2000 but for some applications and investigations it is still well suited, as demonstrated with the *leanCoP* family, e.g., [79, 46, 82, 78] and recent developments [83]. This approach, enumerating proof structures, inherently leads to well-defined proof objects. Moreover, due to the typical enumeration of structures by increasing size, proofs tend to be small. *CMProver* [101, 103], a system implementing this approach, thus provides a further practical way to obtain proof objects for Craig interpolation.

Uniform interpolation is in automated reasoning considered since the 1990s as *second-order quantifier elimination*. The approach is based on *equivalence*, computing for a given second-order formula an *equivalent* first-order formula. It continues a thread that has its roots – just like Craig interpolation [21, 22] – in the *elimination problem* considered in the *algebra of logic*, e.g., by Boole and Schröder. In early studies of first-order logic, elimination was applied to first-order formulas with just unary predicates and no function symbols as a decision procedure by Löwenheim, Skolem and Behmann. Behmann's presentation from 1922 [9] can be viewed as a modern computational method, using equivalence-preserving formula rewriting until innermost second-order subformulas have a shape that allows schematic elimination [100]. Ackermann studied the elimination problem on the basis of full first-order logic in depth and presented in 1935 numerous results [2, 3], including: a polarity-related elimination schema, known today as *Ackermann's lemma*, a variation of resolution, which, for certain formula classes, yields an infinite conjunction as elimination result, a form of Skolemization for predicates that, in certain cases, allows reduction to unary predicates, and the negative result that second-order quantifier elimination on the basis of first-order logic cannot succeed in general. Ackermann's results were reinvented and rediscovered in the early 1990s, leading to two algorithms that expanded into main families of modern elimination algorithms: First, the SCAN algorithm by Gabbay and Ohlbach [32], which eliminates predicates by a form of resolution. It had been implemented [77, 29] on the basis of the *OTTER* first-order prover [67], the predecessor of *Prover9* mentioned above. The presentation of SCAN [32] shows applications in different areas of computational logic and initiated the modern study of second-order quantifier elimination. Second, the DLS algorithm by Doherty, Łukaszewicz and Szałas [95, 24] which rewrites formulas to let Ackermann's lemma become applicable.

For the concepts and methods from *automated* proving some distinguishing aspects can be observed that seem to have their roots in the requirement to provide a basis for implementation but may also be useful in wider and abstract contexts. We sketch four of these.

Certain Forms of Simplicity. Formulas considered for proof search in automated proving are typically in clausal form. Skolemization makes explicit quantifier symbols dispensable. Variable instantiation is typically driven by most general unifiers (most general substitutions that make two terms or atoms identical) of pairs of atoms with the same predicate but in literals with complementary polarity. Herbrand's theorem is a common tool to justify completeness of calculi and to design calculi. In a sense it simplifies the problem of first-order proving by reducing unsatisfiability of a set of first-order clauses to propositional unsatisfiability of a finite set of clause instances. Robinson's resolution [85] has just a *a single rule*. Gentzen's **LK**, aiming to model human reasoning, has 19 rules. As traced by Bibel and Otten [12], the connection method emerged from **LK** in a series of "compressions", reducing the number of rules, via Schütte's one-sided GS [90, 11, 97] (three rules plus cut) until there are no more rules and only a generalization of the *axiom* property remains that characterizes validity. It superimposes a graph structure on the input formula that links *connections*, certain occurrences of literals with the same predicate but complementary polarity. A graph labeling represents the implicit involvement of multiple formula copies, which, under the most general substitution obtained by unifying atoms in connection instances, form a propositionally valid disjunction. Proof search shifts in the connection method from exploring possibilities to apply sequent rules to search for a certificate of validity based on structures superimposed on the formula.

Some Details Turn into Relevant Spaces of Possibilities. An implementation can reveal details that have effect on specific features of an output or on the resources required to obtain it. For Craig interpolation, we have, e.g., choices in proof search that may influence the size of the found proof and thus the size of the interpolant, and choices that have heuristic effect on the duration of the proof search. A subformula might have occurrences in each of the two interpolated formulas. For interpolant calculation it can be considered as belonging to either one, each possibility resulting in a different interpolant.

Approaching Problems and Special Logics with Encodings into Classical Logic. Systems of automated reasoning are complex and highly optimized, making internals hard to modify. Typically they process formulas of some general fundamental class, e.g., classical first-order logic (first-order provers) or classical propositional logic (SAT solvers). The most immediate approach to solve an application problem or to process some special logic is thus to *encode* the problem or the special logic in the machine-supported classical logic.

Robustness of Results. The passage of an abstractly specified method to an implementation typically goes along with a strengthening of the correctness because overlooked subtle aspects and omissions come to light. Claims about the method can be substantiated by experiments that relate it to the state of the art. For automated theorem proving this is well supported by the *TPTP World* with evaluation records from many provers and standardized prover interfaces.

Outline of the Chapter. After setting up preliminaries (Sect. 2), we address preprocessing for first-order proving, with specific constraints for Craig interpolation (Sect. 3). For Craig interpolation our focus is on the *two-stage approach*, where first an essentially propositional ground interpolant is calculated that is then *lifted* to a quantified first-order formula (Sect. 4). We discuss two possibilities to obtain a ground interpolant from a proof, with *clausal tableaux* (Sect. 5), and with *resolution* (Sect. 6). Equality encodings from automated reasoning justify strengthened variations of Craig interpolation (Sect. 7). Also further contributions to Craig interpolation emerged from automated reasoning (Sect. 8). We introduce second-order quantifier elimination with examples and describe the basic algorithms, *DLS* and *SCAN* (Sect. 9).

2 Notation and Preliminaries

Unless specially noted, we consider formulas of first-order logic without equality (which does not preclude incorporation of equality as an axiomatized predicate). The symbol $\models$ expresses entailment, $\equiv$ equivalence. A *negation normal form (NNF) formula* is a quantifier-free formula built up from *literals* (atoms or negated atoms), truth value constants $\top, \bot$, conjunction and disjunction. A *conjunctive normal form (CNF) formula* is an NNF formula that is a conjunction of disjunctions (*clauses*) of literals. A CNF formula is represented by a *set of clauses*. For clauses and sets of clauses the semantic notions of entailment and equivalence apply with respect to their universal closure, i.e., considering their free variables as governed by a universal quantifier. The complement of a literal L is denoted by $\overline{L}$. For a set F of clauses, the set of all literals in the clauses of F is denoted by $\mathcal{L}\mathit{iterals}(F)$, and the set of their complements by $\overline{\mathcal{L}\mathit{iterals}(F)}$.

An occurrence of a subformula in a formula has positive (negative) *polarity*, depending on whether it is in the scope of an even (odd) number of possibly implicit occurrences of negation. We call predicate and function symbols briefly *predicates* and *functions*. Constants are considered as 0-ary functions. Let F be a formula. $\mathcal{V}\mathit{ar}(F)$ is the set of its free individual variables, $\mathcal{F}\mathit{un}(F)$ the set of functions occurring in it, $\mathcal{C}\mathit{onst}(F)$ the set of constants among these, and $\mathcal{P}\mathit{red}^{\pm}(F)$ is the set of all polarity-predicate pairs $+p$ and $-p$ such that p is the predicate of an atom occurrence in F with the indicated polarity, positive by $+$ and negative by $-$. For example, $\mathcal{P}\mathit{red}^{\pm}((\neg \mathsf{p} \vee \mathsf{q}) \wedge (\neg \mathsf{q} \vee \mathsf{r})) = \{-\mathsf{p}, -\mathsf{q}, +\mathsf{q}, +\mathsf{r}\}$. $\mathcal{V}\mathit{oc}^{\pm}(F)$ is $\mathcal{P}\mathit{red}^{\pm}(F) \cup \mathcal{F}\mathit{un}(F) \cup \mathcal{V}\mathit{ar}(F)$. We also define $\mathcal{P}\mathit{red}(F)$ as the set of all predicates that occur in F and $\mathcal{V}\mathit{oc}(F)$ as $\mathcal{P}\mathit{red}(F) \cup \mathcal{F}\mathit{un}(F) \cup \mathcal{V}\mathit{ar}(F)$.

For second-order formulas F, $\mathcal{P}\mathit{red}^{\pm}(F)$ and $\mathcal{P}\mathit{red}(F)$ only contain predicates with *free* occurrences in F. We indicate second-order quantification over predicates with letters p, q, e.g, $\exists p\, F$, and over functions with letters f, g. The special case of quantification over a function that is a constant is just first-order quantification over the constant considered as individual variable. A *sentence* is a formula without free variables. A formula is *ground* if it is quantifier-free and has no free variables. We can now define the central notion of a *Craig-Lyndon* interpolant as follows.

▶ **Definition 1** (Craig-Lyndon Interpolant). *Let F, G be formulas such that $F \models G$. A* Craig-Lyndon interpolant *for F, G is a formula H such that $F \models H$, $H \models G$, and $\mathcal{V}oc^{\pm}(H) \subseteq \mathcal{V}oc^{\pm}(F) \cap \mathcal{V}oc^{\pm}(G)$.*

In the context of Craig-Lyndon interpolation for formulas F, G, we call $\mathcal{V}oc^{\pm}(F) \setminus \mathcal{V}oc^{\pm}(G)$ the *F-only symbols*, $\mathcal{V}oc^{\pm}(G) \setminus \mathcal{V}oc^{\pm}(F)$ the *G-only symbols*, and $\mathcal{V}oc^{\pm}(F) \cap \mathcal{V}oc^{\pm}(G)$ the *FG-shared symbols*. The perspective of validating an entailment $F \models G$ by showing unsatisfiability of $F \wedge \neg G$ is reflected in the notion of *Craig-Lyndon separator* for F, G, defined as Craig-Lyndon interpolant for $F, \neg G$. A *Craig interpolant* is defined like a Craig-Lyndon interpolant, but using $\mathcal{V}oc$ instead of $\mathcal{V}oc^{\pm}$ for the syntactic condition.

We represent a substitution σ of variables by terms as a set of assignments $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. The application of a substitution σ to a formula or term E is written $E\sigma$. We call $E\sigma$ an *instance* of E. If E is an instance of E' and E' is an instance of E, we say that both are *variants* of each other. A substitution is called *ground* if its range is a set of ground terms. We use the notation for substitutions also to express the substitution of a predicate by another one. To express the substitution of a predicate by a complex formula we generalize it as follows. Let F be a formula, let p be an n-ary predicate, let $\boldsymbol{x} = x_1, \ldots, x_n$ be distinct individual variables, and let G be a formula. Then $F\{p \mapsto \lambda \boldsymbol{x}.G\}$ denotes F with all occurrences $p(t_1, \ldots, t_n)$ of p replaced by the respective instance $G\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ of G. For example, $(p(\mathsf{a}) \wedge p(\mathsf{b}))\{p \mapsto \lambda x.(\mathsf{q}(x) \vee \mathsf{r}(x))\}$ stands for $(\mathsf{q}(\mathsf{a}) \vee \mathsf{r}(\mathsf{a})) \wedge (\mathsf{q}(\mathsf{b}) \vee \mathsf{r}(\mathsf{b}))$.

We use *tuples* of terms, variables, functions, and predicates for several purposes. For example, to abbreviate a nested quantification $\exists x_1 \ldots \exists x_n F$ with the same quantifier as a single quantification $\exists \boldsymbol{x} F$, with $\boldsymbol{x}$ representing the tuple $x_1 \ldots x_n$. As another example, we write an atom $p(t_1, \ldots, t_n)$ as $p(\boldsymbol{t})$ with $\boldsymbol{t}$ representing the tuple $t_1 \ldots t_n$ of argument terms. If the ordering and number of occurrences of members are irrelevant, a tuple can be identified with the set of its members. In such cases we use a permissive notation, where tuples can directly appear as arguments of set operations. For tuples $\boldsymbol{x}$ of variables we assume that the members are pairwise different. For tuples $\boldsymbol{t} = t_1 \ldots t_n$ and $\boldsymbol{s} = s_1 \ldots s_n$ of terms, we write the formula $\boldsymbol{t} = \boldsymbol{s}$ as shorthand for $t_1 = s_1 \wedge \ldots \wedge t_n = s_n$, and $\boldsymbol{t} \neq \boldsymbol{s}$ for $t_1 \neq s_1 \vee \ldots \vee t_n \neq s_n$. For formulas F, tuples $\boldsymbol{x} = x_1, \ldots, x_n$ of variables and $\boldsymbol{t} = t_1, \ldots, t_n$ of terms we write $F\{\boldsymbol{x} \mapsto \boldsymbol{t}\}$ as shorthand for $F\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$.

Tables 1–4 show equivalences EQ 1–EQ 30 that will be referenced and discussed in various contexts. The presentation is such that rewriting from the left to the right side is the more common case. Nevertheless, for some equivalences also rewriting in the converse direction has applications.

3 Preprocessing: Skolemization, Clausification, Simplification

Most core techniques of automated first-order provers operate on a set of clauses. The given first-order formula whose unsatisfiability is to be proven is thus *preprocessed* to a set of clauses which is *equi-satisfiable*, i.e., is unsatisfiable iff the original formula is so. Conceptually,

Table 1 Equivalences that are independent from quantification. They are to be considered modulo commutativity of $\wedge$ and $\vee$. EQ 2 and EQ 3 provide alternate ways to eliminate $\leftrightarrow$.

Eliminating Implication and Biconditional

(EQ 1) $F \to G \equiv \neg F \vee G$
(EQ 2) $F \leftrightarrow G \equiv (\neg F \vee G) \wedge (F \vee \neg G)$
(EQ 3) $F \leftrightarrow G \equiv (F \wedge G) \vee (\neg F \wedge \neg G)$

Converting to Negation Normal Form

(EQ 4) $\neg\neg F \equiv F$
(EQ 5) $\neg(F \wedge G) \equiv \neg F \vee \neg G$
(EQ 6) $\neg(F \vee G) \equiv \neg F \wedge \neg G$

Distributing

(EQ 7) $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$
(EQ 8) $F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$

Truth Value Simplification

(EQ 9) $F \wedge \top \equiv F$
(EQ 10) $F \wedge \bot \equiv \bot$
(EQ 11) $F \vee \top \equiv \top$
(EQ 12) $F \vee \bot \equiv F$
(EQ 13) $\neg\top \equiv \bot$
(EQ 14) $\neg\bot \equiv \top$

Table 2 Equivalences involving quantifiers. They apply to first- and second-order formulas F, G, and also to individual and second-order quantifiers as well as to combinations of these. The placeholder Q indicates that the equivalence holds for both quantifiers $Q \in \{\forall, \exists\}$.

Moving Negation over Quantifiers

(EQ 15) $\neg\forall x\, F \equiv \exists x\, \neg F$
(EQ 16) $\neg\exists x\, F \equiv \forall x\, \neg F$

Prenexing / Moving Quantifiers Inwards

(EQ 17) $\forall x\, F \wedge \forall x\, G \equiv \forall x\, (F \wedge G)$
(EQ 18) $\exists x\, F \vee \exists x\, G \equiv \exists x\, (F \vee G)$
(EQ 19) $Qx\, F \vee G \equiv Qx\, (F \vee G)$ if $x \notin \mathcal{V}oc(G)$
(EQ 20) $Qx\, F \wedge G \equiv Qx\, (F \wedge G)$ if $x \notin \mathcal{V}oc(G)$

Reordering Quantifiers

(EQ 21) $QxQy\, F \equiv QyQx\, F$

Eliminating Void Quantifiers

(EQ 22) $Qx\, F \equiv F$ if $x \notin \mathcal{V}oc(F)$

Table 3 Equivalences that justify pushing in and pulling out terms or formulas. Formulas F, G are first-order. For EQ 23 we assume that $\mathcal{V}oc(\boldsymbol{t}_i) \cap \boldsymbol{x} = \varnothing$, $i \in \{1, \ldots, n\}$, for EQ 24 and EQ 25 that $\mathcal{V}oc(\boldsymbol{t}) \cap \boldsymbol{x} = \varnothing$, and for EQ 26–EQ 28 that p does not occur in G and that variables free in G are not quantified in F.

Pushing In / Pulling Out Terms

(EQ 23) $\exists \boldsymbol{x}\, [(\bigvee_{i=1}^{n} \boldsymbol{x} = \boldsymbol{t}_i) \wedge F] \equiv \bigvee_{i=1}^{n} F\{\boldsymbol{x} \mapsto \boldsymbol{t}_i\}$
(EQ 24) $\forall \boldsymbol{x}\, (\boldsymbol{x} = \boldsymbol{t} \to F) \equiv F\{\boldsymbol{x} \mapsto \boldsymbol{t}\}$
(EQ 25) $\forall \boldsymbol{x}\, (F \to \boldsymbol{x} \neq \boldsymbol{t}) \equiv \neg F\{\boldsymbol{x} \mapsto \boldsymbol{t}\}$

Pushing In / Pulling Out Formulas, Ackermann's Lemma

(EQ 26) $\exists p\, [(\forall \boldsymbol{x}\, (p(\boldsymbol{x}) \leftrightarrow G) \wedge F] \equiv F\{p \mapsto \lambda \boldsymbol{x}.G\}$
(EQ 27) $\exists p\, [(\forall \boldsymbol{x}\, (p(\boldsymbol{x}) \to G) \wedge F] \equiv F\{p \mapsto \lambda \boldsymbol{x}.G\}$ if $-p \notin \mathcal{P}red^{\pm}(F)$
(EQ 28) $\exists p\, [(\forall \boldsymbol{x}\, (G \to p(\boldsymbol{x})) \wedge F] \equiv F\{p \mapsto \lambda \boldsymbol{x}.G\}$ if $+p \notin \mathcal{P}red^{\pm}(F)$

Table 4 Equivalences involving quantifier switching. Here f is a function such that $f \notin \mathcal{F}un(F)$, p, q are predicates such that $q \notin \mathcal{P}red(F)$, $\boldsymbol{x}, \boldsymbol{y}$ are tuples of variables, and y is a variable.

Second-Order Skolemization

(EQ 29) $\forall \boldsymbol{x} \exists y\, F \equiv \exists f \forall \boldsymbol{x} F\{y \mapsto f(\boldsymbol{x})\}$

Ackermann's Quantifier Switching

(EQ 30) $\forall \boldsymbol{x} \exists p\, F \equiv \exists q \forall \boldsymbol{x} F\{p \mapsto \lambda \boldsymbol{y}.q(\boldsymbol{y}\boldsymbol{x})\}$

this proceeds in the following steps: (1) *Prenexing*, bringing the formula into a form where a quantifier prefix is applied to a quantifier-free formula by rewriting with EQ 15–EQ 20. (2) *Skolemization*, eliminating existential variables with EQ 29. This yields a formula whose quantifier prefix starts with existential quantifiers over fresh functions, the *Skolem functions*. Unless the Skolem function is a constant, the quantifier is a second-order quantifier. (For determining unsatisfiability, the existential quantifier prefix over the Skolem functions can be dropped.) (3) *CNF transformation*, converting the formula to a set of clauses, by EQ 1–EQ 7. (4) *Simplification*, simplifying the set of clauses to an equi-satisfiable set.

Typically, these phases are performed interleaved. Skolemization, e.g., might be applied separately to subformulas obtained by shifting quantifiers with EQ 17–EQ 20 outwards as well as inwards [33, 75]. Distributing with EQ 7 at CNF transformation calls for simplifying formulas before they get duplicated. Blow-up through distributing can be completely avoided with *structure-preserving* (or *definitional*) *normal forms* [5] that are based on rewriting with EQ 26–EQ 28 from right to left before CNF transformation.

Simplification is important in automated reasoning and has many facets. The general idea is that the expensive core proving is accompanied by cheaper operations that do whatever can be done with few resources to make the problem easier. Simplifications are not just applied at preprocessing but also incorporated into the core proving. For example, the core operation of CDCL SAT solvers involves unit propagation, a propositional simplification. As another example, CDCL SAT solvers interrupt the core reasoning to simplify the computed lemmas (learned clauses) in *inprocessing* phases. Resolution provers can delete a clause if it is newly inferred and subsumed by a previously inferred clause (*forward subsumption*) and if it was previously inferred and is subsumed by a newly inferred clause (*backward subsumption*).

Simplifications for theorem proving fall into two broad categories, those that preserve equivalence and those that do not preserve equivalence but equi-satisfiability. Well-known simplifications of the first class are deletion of tautological clauses, deletion of subsumed clauses, replacing clauses by condensation, and replacing clauses by subsumption resolution [17, 7, 75]. In view of interpolation, we note that none of these introduces additional functions or predicates, which makes them straightforwardly applicable to each of the two interpolated formulas.

A well-known simplification of the second class is the deletion of a clause that contains a *pure* literal, a literal whose predicate does not occur with opposite polarity in the set of clauses. A closer looks shows that this does not just preserve equi-satisfiability, but, moreover, if F' is obtained from F by deleting a clause with a pure literal with predicate p, then $\exists p\, F \equiv \exists p\, F'$. Evidently, no additional functions or predicates are introduced. We can utilize such properties of simplifications that do not preserve equivalence but actually preserve more than just equi-satisfiability to justify their use in the preprocessing of interpolated formulas. The following proposition shows corresponding constraints that are suitable for Craig interpolation.

▶ **Proposition 2** (Simplifying Interpolated Formulas). *Let F, G, F', G' be first-order formulas such that*

$$\exists \boldsymbol{f}'\, F' \equiv \exists \boldsymbol{f} F,\ \forall \boldsymbol{g}'\, G' \equiv \forall \boldsymbol{g}\, G,\ \mathcal{V}oc(\exists \boldsymbol{f}'\, F') = \mathcal{V}oc(\exists \boldsymbol{f}\, F),\ \textit{and}\ \mathcal{V}oc(\forall \boldsymbol{f}'\, G') = \mathcal{V}oc(\forall \boldsymbol{g}\, G),$$

where $\boldsymbol{f}' = \mathcal{V}oc(F') \setminus \mathcal{V}oc(G')$, $\boldsymbol{g}' = \mathcal{V}oc(G') \setminus \mathcal{V}oc(F')$, $\boldsymbol{f} = \mathcal{V}oc(F) \setminus \mathcal{V}oc(G)$, and $\boldsymbol{g} = \mathcal{V}oc(G) \setminus \mathcal{V}oc(F)$. Then, a first-order formulas H is a Craig interpolant for F, G iff it is a Craig interpolant for F', G'.

The quantifications in this proposition are mixed, first- and second-order, over predicates, functions, and constants or variables. The proposition shows constraints for preprocessing given formulas F, G to simpler formulas F', G' that have the same Craig interpolants. Simplifications that match these constraints are, e.g., the mentioned deletion of a clause with a pure literal, unless the literal's predicate is FG-shared. At least in the propositional case also *blocked clause elimination* [53, 44, 47], which generalizes the purity deletion, is justified by Prop. 2, if the predicates of the blocking literals are not FG-shared. Moreover, since Prop. 2 stays unchanged if F, G and F', G' are switched, also cases of *addition* of blocked clauses [53] are justified. Justified transformations that enrich the set of predicates include conversion to structure-preserving normal forms, if the fresh predicates introduced in the conversion of F and of G are distinct. Also second-order quantifier elimination, if it does not eliminate FG-shared predicates, is justified. Modern SAT solvers [28] as well as some first-order provers, such as *Prover9*, use elimination in specific cases as simplifications. However, typically these systems provide no options for specifying predicates as protected from being eliminated.

For *Craig-Lyndon* interpolation, Prop. 2 can be relaxed by "quantification over predicates only in a specific polarity", which can be defined as follows.

▶ **Definition 3** (Polarity-Sensitive Predicate Quantification). *For second-order formulas F and predicates p define $\exists{+}p\, F$ and $\exists{-}p\, F$ as follows, where p' is a fresh predicate.*

$$\exists{+}p\, F \stackrel{\text{def}}{=} \exists p'\, (F\{p \mapsto p'\} \wedge \forall \boldsymbol{x}\, (p(\boldsymbol{x}) \to p'(\boldsymbol{x}))). \qquad \forall{-}p\, F \stackrel{\text{def}}{=} \neg\exists{+}p\, \neg F.$$
$$\exists{-}p\, F \stackrel{\text{def}}{=} \exists p'\, (F\{p \mapsto p'\} \wedge \forall \boldsymbol{x}\, (p'(\boldsymbol{x}) \to p(\boldsymbol{x}))). \qquad \forall{+}p\, F \stackrel{\text{def}}{=} \neg\exists{-}p\, \neg F.$$

The following examples illustrate polarity-sensitive predicate quantification by showing the expansion into conventional predicate quantification and an equivalent first-order formula, obtained by second-order quantifier elimination, which is discussed in Sect. 9.

▶ **Example 4.**

(i) $\exists{-}\mathsf{p}\,(\mathsf{p}(\mathsf{a}) \wedge \neg\mathsf{p}(\mathsf{b})) \equiv \exists p'\,(p'(\mathsf{a}) \wedge \neg p'(\mathsf{b}) \wedge \forall x(p'(x) \to \mathsf{p}(x))) \equiv \mathsf{a} \neq \mathsf{b} \wedge \mathsf{p}(\mathsf{a}).$

(ii)
$$\begin{aligned} & \exists{-}\mathsf{q}\,(\forall x(\mathsf{p}(x) \to \mathsf{q}(x)) \wedge \forall x\,(\mathsf{q}(x) \to \mathsf{r}(x))) \\ \equiv\ & \exists q'\,(\forall x\,(\mathsf{p}(x) \to q'(x)) \wedge \forall x\,(q'(x) \to \mathsf{r}(x)) \wedge \forall x\,(q'(x) \to \mathsf{q}(x))) \\ \equiv\ & \forall x(\mathsf{p}(x) \to (\mathsf{q}(x) \wedge \mathsf{r}(x))). \end{aligned}$$

⌟

Vice versa, conventional predicate quantification can be expressed by polarity-sensitive quantification: $\exists p\, F \equiv \exists{+}p \exists{-}p\, F$. Proposition 2 can be adapted to Craig-Lyndon interpolation by considering predicate quantification as polarity-sensitive and using $\mathcal{V}oc^{\pm}$ in place of $\mathcal{V}oc$.

4 The Two-Stage Approach to Craig Interpolation in First-Order Logic

Herbrand's theorem, due to Herbrand in 1930 [40], is a central tool in automated first-order theorem proving, where it is commonly stated in the following form.

▶ **Theorem 5** (Herbrand's Theorem). *A (possibly infinite) set F of clauses is unsatisfiable iff there is a finite set of ground instances of clauses of F.*

▶ **Example 6.** Let $F = \{\mathsf{p}(\mathsf{a}),\ \neg\mathsf{p}(x) \vee \mathsf{pp}(\mathsf{f}(x)),\ \neg\mathsf{p}(\mathsf{f}(\mathsf{f}(\mathsf{a})))\}$. Then $\{\mathsf{p}(\mathsf{a}),\ (\neg\mathsf{p}(x) \vee \mathsf{pp}(\mathsf{f}(x)))\{x \mapsto \mathsf{a}\},\ (\neg\mathsf{p}(x) \vee \mathsf{pp}(\mathsf{f}(x)))\{x \mapsto \mathsf{f}(\mathsf{a})\},\ \neg\mathsf{p}(\mathsf{f}(\mathsf{f}(\mathsf{a})))\}$ is an unsatisfiable set of ground instances of F. ⌟

To obtain an unsatisfiable set of ground clauses according to Herbrand's theorem it is sufficient to instantiate with terms built from functions in the given set of clauses and, if there is no constant among these, an additional constant c_0.

Given the conversion of a first-order formula to a set of clauses as discussed in Sect. 3, Herbrand's theorem tells us that behind a first-order proof there is a propositional proof, with ground atoms in the role of propositional variables. The proof object obtained from a theorem prover can represent such a ground proof. This suggests to calculate a Craig-Lyndon interpolant for first-order formulas in *two stages*: (1) Calculating a Craig-Lyndon interpolant from a ground proof with propositional techniques; (2) Interpolant lifting, that is replacing ground terms with quantified variables.

A corresponding two-stage proof of the Craig interpolation property of first-order logic was given in the 1960s by Kreisel and Krivine [52]. Harrison [39] presents a version of their proof that is adapted to automated reasoning, explicitly referring to Skolemization and Herbrand's theorem. In automated reasoning, the two-stage approach was introduced in 1995 with Huang's paper [43] on interpolation with resolution, which was rediscovered in the 2010s by Bonacina and Johansson [15], who coined the name *two-stage approach*. They prove a limited form of interpolant lifting that applies just to constants in contrast to compound terms. Baaz and Leitsch [6] prove a general form, which they call *abstraction*, on the basis of a natural deduction calculus.

We present here a general form of interpolant lifting from [104], where it is proven on the basis of Skolemization and Herbrand's theorem. The relationships that allow a ground formula H_{GRD} to be lifted to a Craig interpolant for first-order formulas F, G are captured by the notion of *interpolant lifting base*. Given a lifting base, i.e., if the specified relationships hold, a first-order Craig interpolant for F, G can be constructed from H_{GRD} by replacing certain occurrences of terms with variables and prepending a certain first-order quantifier prefix upon these variables. The Craig interpolation property for first-order logic follows since if $F \models G$, then an abstract construction ensures existence of a lifting base. Clausal tableaux and resolution deductions provide proofs that can straightforwardly be viewed *as if* they were constructed by the abstract method. This makes the two-stage approach applicable with

different underlying first-order calculi for ground interpolation, where we will discuss clausal tableaux (Sect. 5) and resolution (Sect. 6).

We now specify *interpolant lifting base* formally. W.l.o.g. we assume that the interpolated first-order formulas F, G are *sentences*: To interpolate F, G with free variables, we first replace these by dedicated constants and finally replace the constants in the obtained interpolant by the corresponding variables.

▶ **Definition 7** (Interpolant Lifting Base). *An* interpolant lifting base *is a tuple*

$$\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{GRD} \rangle,$$

where F, G are first-order sentences, $\boldsymbol{f}, \boldsymbol{g}$ are disjoint tuples of distinct function symbols, H_{GRD} (the subscript GRD suggesting ground*) is a ground formula such that there exist quantifier-free formulas $F_{EXP}(\boldsymbol{u}), G_{EXP}(\boldsymbol{v})$ (the subscript EXP suggesting* expansion*) with disjoint tuples of free variables $\boldsymbol{u}, \boldsymbol{v}$, respectively, and a ground substitution η with the following properties.*

(1) $F \models \exists \boldsymbol{f} \forall \boldsymbol{u}\, F_{EXP}(\boldsymbol{u})$.
(2) $\mathcal{P}red^{\pm}(F_{EXP}(\boldsymbol{u})) \subseteq \mathcal{P}red^{\pm}(F)$.
(3) $\mathcal{F}un(F_{EXP}(\boldsymbol{u})) \subseteq (\mathcal{F}un(F) \cap \mathcal{F}un(G)) \cup \boldsymbol{f}$.
(4) $\mathcal{F}un(F) \cap \boldsymbol{g} = \varnothing$.

(1′) $\forall \boldsymbol{g} \exists \boldsymbol{v}\, G_{EXP}(\boldsymbol{v}) \models G$.
(2′) $\mathcal{P}red^{\pm}(G_{EXP}(\boldsymbol{v})) \subseteq \mathcal{P}red^{\pm}(G)$.
(3′) $\mathcal{F}un(G_{EXP}(\boldsymbol{v})) \subseteq (\mathcal{F}un(F) \cap \mathcal{F}un(G)) \cup \boldsymbol{g}$.
(4′) $\mathcal{F}un(G) \cap \boldsymbol{f} = \varnothing$.

(5) $\mathcal{D}om(\eta) = \boldsymbol{u} \cup \boldsymbol{v}$.
(6) $\mathcal{F}un(\mathcal{R}ng(\eta)) \subseteq \mathcal{F}un(F_{EXP}(\boldsymbol{u})) \cup \mathcal{F}un(G_{EXP}(\boldsymbol{v})) \cup \{c_0\}$, *where c_0 is a constant in $\boldsymbol{fg}$.*
(7) *H_{GRD} is a Craig-Lyndon interpolant for $F_{EXP}(\boldsymbol{u})\eta$ and $G_{EXP}(\boldsymbol{v})\eta$.*

For first-order sentences F, G such that $F \models G$ we can construct an interpolant lifting base as follows. Apply prenexing, Skolemization and CNF transformation independently to each of $F, \neg G$ to obtain formulas $\exists \boldsymbol{f}' \forall \boldsymbol{u}'\, M'(\boldsymbol{u}')$, $\exists \boldsymbol{g}' \forall \boldsymbol{v}'\, N'(\boldsymbol{v}')$ that are equivalent to $F, \neg G$, respectively, where $\boldsymbol{f}', \boldsymbol{g}'$ are the introduced Skolem functions and $M'(\boldsymbol{u}'), N'(\boldsymbol{v}')$ are sets of clauses, with free variables $\boldsymbol{u}', \boldsymbol{v}'$. Since $M'(\boldsymbol{u}') \cup N'(\boldsymbol{v}')$ is unsatisfiable, by Herbrand's theorem there is an unsatisfiable finite set of ground instances of clauses from $M'(\boldsymbol{u}') \cup N'(\boldsymbol{v}')$. This set of ground instances can contain different instances of the same clause in $M'(\boldsymbol{u}') \cup N'(\boldsymbol{v}')$. Thus, this set can be considered as obtained in two steps, by first creating *copies* of clauses in $M'(\boldsymbol{u}') \cup N'(\boldsymbol{v}')$, one copy for each ground instance, where a *copy* of a clause is a variant with fresh variables, and, second, applying a single ground substitution to the set of copies. Let $M_{\text{EXP}}(\boldsymbol{u})$ ($N_{\text{EXP}}(\boldsymbol{v})$) with free variables $\boldsymbol{u}$ ($\boldsymbol{v}$) be the set of these copies and let η be the ground substitution. The range of η is a set of terms built from functions in $M'(\boldsymbol{u}') \cup N'(\boldsymbol{v}')$ and, if there is no constant among these, a fresh constant c_0. Let $\boldsymbol{f}$ ($\boldsymbol{g}$) be the union of the Skolem functions $\boldsymbol{f}'$ ($\boldsymbol{g}'$) introduced for F (G) and the F-only (G-only) functions. In case a fresh c_0 was introduced, add it either to $\boldsymbol{f}$ or to $\boldsymbol{g}$. Let $F_{\text{EXP}}(\boldsymbol{u}) = M_{\text{EXP}}(\boldsymbol{u})$, let $G_{\text{EXP}}(\boldsymbol{v}) = \neg N_{\text{EXP}}(\boldsymbol{v})$, and let H_{GRD} be a ground Craig-Lyndon interpolant for $F_{\text{EXP}}(\boldsymbol{u})\eta, G_{\text{EXP}}(\boldsymbol{v})\eta$. Then $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}} \rangle$ is an interpolant lifting base.

Neither the "copy expansions" $F_{\text{EXP}}(\boldsymbol{u}), G_{\text{EXP}}(\boldsymbol{v})$, nor the ground substitution η have to be actually constructed for obtaining an interpolant lifting base. Just their *existence* is required.

▶ **Example 8.** Each of the following examples shows a lifting base $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}}\rangle$ together with suitable $F_{\text{EXP}}(u), G_{\text{EXP}}(v), \eta$. Introductory comments show specific properties.

(i) $\boldsymbol{f}$ contains a non-constant; members of $\boldsymbol{f}$ and of $\boldsymbol{g}$ occur in H_{GRD}: $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}}\rangle = \langle \forall x\, \mathsf{p}(x, \mathsf{f}(x)),\ \exists x\, \mathsf{p}(\mathsf{g}, x),\ \mathsf{f},\ \mathsf{g},\ \mathsf{p}(\mathsf{g}, \mathsf{f}(\mathsf{g}))\rangle$, with $F_{\text{EXP}}(u) = \mathsf{p}(u, \mathsf{f}(u))$, $G_{\text{EXP}}(v) = \mathsf{p}(\mathsf{g}, v)$, $\eta = \{u \mapsto \mathsf{g}, v \mapsto \mathsf{f}(\mathsf{g})\}$. It holds that $F_{\text{EXP}}(u)\eta = G_{\text{EXP}}(v)\eta = H_{\text{GRD}} = \mathsf{p}(\mathsf{g}, \mathsf{f}(\mathsf{g}))$.

(ii) A member of $\boldsymbol{f}$ (i.e., f_2) is a Skolem function: $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}}\rangle = \langle \forall x \exists y\, \mathsf{p}(x, y, \mathsf{f}_1),\ \exists x \exists y\, \mathsf{p}(\mathsf{g}, x, y),\ \mathsf{f}_1\mathsf{f}_2,\ \mathsf{g},\ \mathsf{p}(\mathsf{g}, \mathsf{f}_2(\mathsf{g}), \mathsf{f}_1)\rangle$, with $F_{\text{EXP}}(u) = \mathsf{p}(u, \mathsf{f}_2(u), \mathsf{f}_1)$, $G_{\text{EXP}}(v_1, v_2) = \mathsf{p}(\mathsf{g}, v_1, v_2)$, $\eta = \{u \mapsto \mathsf{g}, v_1 \mapsto \mathsf{f}_2(\mathsf{g}), v_2 \mapsto \mathsf{f}_1\}$. It holds that $F_{\text{EXP}}(u)\eta = G_{\text{EXP}}(v_1, v_2)\eta = H_{\text{GRD}} = \mathsf{p}(\mathsf{g}, \mathsf{f}_2(\mathsf{g}), \mathsf{f}_1)$.

(iii) $F_{\text{EXP}}(u_1, u_2)$ is a conjunction of different variants of the quantifier-free inner formula $\mathsf{p}(x, \mathsf{f})$ of F: $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}}\rangle = \langle \forall x\, \mathsf{p}(x, \mathsf{f}),\ \exists x\, \mathsf{p}(\mathsf{g}_1, x) \wedge \exists x\, \mathsf{p}(\mathsf{g}_2, x),\ \mathsf{f},\ \mathsf{g}_1\mathsf{g}_2,\ \mathsf{p}(\mathsf{g}_1, \mathsf{f}) \wedge \mathsf{p}(\mathsf{g}_2, \mathsf{f})\rangle$, with $F_{\text{EXP}}(u_1, u_2) = \mathsf{p}(u_1, \mathsf{f}) \wedge \mathsf{p}(u_2, \mathsf{f})$, $G_{\text{EXP}}(v) = \mathsf{p}(\mathsf{g}_1, v) \wedge \mathsf{p}(\mathsf{g}_2, v)$, $\eta = \{u_1 \mapsto \mathsf{g}_1, u_2 \mapsto \mathsf{g}_2, v \mapsto \mathsf{f}\}$. It holds that $F_{\text{EXP}}(u_1, u_2)\eta = G_{\text{EXP}}(v)\eta = H_{\text{GRD}} = \mathsf{p}(\mathsf{g}_1, \mathsf{f}) \wedge \mathsf{p}(\mathsf{g}_2, \mathsf{f})$.

(iv) Formulas F, G extend those of Example (8.i) by literals with predicates q, r that occur in only one of F, G, and a second function symbol in G. Differently from the previous three cases, the ground interpolant is not an instance of F_{EXP} and G_{EXP}. $\langle F = \forall x\, \mathsf{p}(x, \mathsf{f}(x)) \wedge \forall x \forall y\, \mathsf{q}(\mathsf{f}(x), y),\ G = \exists x\, (\mathsf{p}(\mathsf{g}_1, x) \vee \mathsf{r}(\mathsf{g}_2(x))),\ \boldsymbol{f} = \mathsf{f},\ \boldsymbol{g} = \mathsf{g}_1, \mathsf{g}_2,\ H_{\text{GRD}} = \mathsf{p}(\mathsf{g}_1, \mathsf{f}(\mathsf{g}_1))\rangle$, with $F_{\text{EXP}}(u_1, u_2, u_3) = \mathsf{p}(u_1, \mathsf{f}(u_1)) \wedge \mathsf{q}(\mathsf{f}(u_2), u_3)$, $G_{\text{EXP}}(v) = \mathsf{p}(\mathsf{g}_1, v) \vee \mathsf{r}(\mathsf{g}_2(v))$, $\eta = \{u_1 \mapsto \mathsf{g}_1, v \mapsto \mathsf{f}(\mathsf{g}_1), u_2 \mapsto \mathsf{g}_2(\mathsf{f}(\mathsf{g}_1)), u_3 \mapsto \mathsf{g}_1\}$. It holds that $F_{\text{EXP}}(u_1, u_2, u_3)\eta = \mathsf{p}(\mathsf{g}_1, \mathsf{f}(\mathsf{g}_1)) \wedge \mathsf{q}(\mathsf{f}(\mathsf{g}_2(\mathsf{f}(\mathsf{g}_1))), \mathsf{g}_1)$ and $G_{\text{EXP}}(v)\eta = \mathsf{p}(\mathsf{g}_1, \mathsf{f}(\mathsf{g}_1)) \vee \mathsf{r}(\mathsf{g}_2(\mathsf{f}(\mathsf{g}_1)))$. Other values of F_{EXP}, G_{EXP} and η are also possible. For example, u_2, u_3 could be merged with u_1, or η could assign u_2, u_3 to other ground terms. ⌟

Given a lifting base $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}}\rangle$ for first-order sentences F, G such that $F \models G$, we can construct a Craig-Lyndon interpolant for F, G as specified with Theorem 9 below. Its statement needs additional notation. We write $s \lhd t$ to express that s is a strict subterm of t. If $\boldsymbol{f}$ is a set or sequence of functions, then an $\boldsymbol{f}$-term is a term whose outermost symbol is in $\boldsymbol{f}$. An occurrence of an $\boldsymbol{f}$-term in a formula F is *$\boldsymbol{f}$-maximal* if it is not within an occurrence of another $\boldsymbol{f}$-term. If σ is an injective variable substitution whose range is a set of ground $\boldsymbol{f}$-terms, then $F\langle\sigma^{-1}\rangle$ denotes F with all $\boldsymbol{f}$-maximal occurrences of terms t in the range of σ replaced by the variable that is mapped by σ to t. As an example, let $\boldsymbol{f} = \mathsf{fg}$, let $F = \mathsf{p}(\mathsf{h}(\mathsf{f}(\mathsf{a}), \mathsf{g}(\mathsf{f}(\mathsf{a}))))$ and let $\sigma = \{x \mapsto \mathsf{f}(\mathsf{a}),\ y \mapsto \mathsf{g}(\mathsf{f}(\mathsf{a}))\}$. Then $F\langle\sigma^{-1}\rangle = \mathsf{p}(\mathsf{h}(x, y))$. We can now state the theorem that specifies the variable introduction into ground interpolants.

▶ **Theorem 9** (Interpolant Lifting). *Let $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{GRD}\rangle$ be an interpolant lifting base. Let $\{t_1, \ldots, t_n\}$ be the set of the $\boldsymbol{fg}$-terms with an $\boldsymbol{fg}$-maximal occurrence in H_{GRD}, ordered such that if $t_i \lhd t_j$, then $i < j$. Let $\{v_1, \ldots, v_n\}$ be a set of fresh variables and let σ be the injective substitution $\sigma \stackrel{\text{def}}{=} \{v_i \mapsto t_i \mid i \in \{1, \ldots, n\}\}$. For $i \in \{1, \ldots, n\}$ let $Q_i \stackrel{\text{def}}{=} \exists$ if $v_i\sigma$ is an $\boldsymbol{f}$-term and $Q_i \stackrel{\text{def}}{=} \forall$ otherwise, that is, if $v_i\sigma$ is a $\boldsymbol{g}$-term. Then*

$$H = Q_1 v_1 \ldots Q_n v_n\, H_{GRD}\langle\sigma^{-1}\rangle$$

is a Craig-Lyndon interpolant for F, G.

Theorem 9 shows the construction of a first-order sentence H from a given ground formula H_{GRD} and sets, $\boldsymbol{f}, \boldsymbol{g}$, of function symbols. Sentence H is obtained from H_{GRD} by replacing $\boldsymbol{fg}$-maximal occurrences of $\boldsymbol{f}$-terms and $\boldsymbol{g}$-terms with variables, and prepending a quantifier prefix over these. Variables replacing an $\boldsymbol{f}$-term ($\boldsymbol{g}$-term) are existential (universal), and whenever variables x, y replace terms s, t, respectively, such that $s \lhd t$, then the quantifier over x precedes that over y. The obtained H is a Craig-Lyndon interpolant for the first-order sentences F, G, provided $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}} \rangle$ is an interpolant lifting base.

▶ **Example 10.** Consider the interpolant lifting bases from Example 8. Respective Craig-Lyndon interpolants according to Theorem 9 are as follows. For (8.i) and for (8.iv): $\forall v_1 \exists v_2\, \mathsf{p}(v_1, v_2)$. For (8.ii): $\exists v_1 \forall v_2 \exists v_3\, \mathsf{p}(v_2, v_3, v_1)$. Also other orderings of the quantifiers are possible according to Theorem 9. The only required condition is (expressed with the variable names of the shown interpolant) that $\forall v_2$ must precede $\exists v_3$. For (8.iii): $\exists v_1 \forall v_2 \forall v_3\, (\mathsf{p}(v_2, v_1) \wedge \mathsf{p}(v_3, v_1))$, which is equivalent to $\exists v_1 \forall v_2 \mathsf{p}(v_2, v_1)$. Also arbitrary other quantifier orderings are possible. ⌟

5 Ground Interpolation with Clausal Tableaux

We discuss clausal tableaux as a technique for first-order theorem proving and ground interpolation in the two-stage approach.

5.1 Clausal Tableaux – Proof Objects for Automated Reasoning

The framework of *clausal tableaux* [55, 56, 59, 57, 38] was developed in the 1990s by Letz as a bridge between analytic tableaux and a family of methods for fully automated first-order proving with highly optimized systems. These methods, model elimination [63] and the connection method [11], share with resolution the operation on sets of clauses, but instead of generating consequences they enumerate proof structures. First-order provers that can be described as constructing clausal tableaux include the *Prolog Technology Theorem Prover (PTTP)* [93], *SETHEO* [58, 59] and *METEOR* [4]. Until around 2001, *SETHEO* was a competitive first-order prover. *CMProver* [101, 103] is a Prolog-based system that is still maintained. In 2003 *leanCoP* [79] was designed as a minimalistic Prolog-based system that has since been used for numerous studies and adaptations to non-classical logics. For Craig interpolation, the crucial relevance of the clausal tableau framework is motivated by its *proof objects* – the *clausal tableaux*. Based on clauses instead of complex formulas, they are compatible with highly optimized fully automated systems. Their tree structure allows inductive calculation of ground interpolants as known from analytic tableaux and sequent systems. The simplicity of clausal tableaux facilitates abstract investigations, proof transformations, and developing strengthened variations of Craig-Lyndon interpolation.

▶ **Definition 11** (Clausal Tableau and Related Notions). *A* clausal tableau *(briefly* tableau*)* for *a set F of clauses is a finite ordered tree whose nodes N with exception of the root are labeled with a literal* $\mathsf{lit}(N)$*, such that for each inner node M the disjunction of the literals of all its children in their left-to-right order,* $\mathsf{clause}(M)$*, is an instance of a clause in F. The clauses* $\mathsf{clause}(M)$ *are called the* clauses of *the tableau. A tableau whose clauses are ground is called* ground*. A branch of a tableau is* closed *iff it contains nodes with complementary literals. A node is* closed *iff all branches through it are closed. A tableau is* closed *iff its root is closed.*

All occurrences of variables in the clauses of a clausal tableau are free and their scope spans the whole tableau. That is, we consider *free-variable tableaux* [56, p. 158ff] [57, Sect. 2.2], or *rigid* variables [38, p. 114]. That a clausal tableau indeed represents a proof is stated in the following proposition, which follows from Herbrand's theorem (Theorem 5).

▶ **Proposition 12.** *A set F of clauses is unsatisfiable iff there exists a closed clausal tableau for F.*

Clauses in a closed tableau according to Prop. 12 may have variables. A closed *ground* tableau, in direct correspondence to Herbrand's theorem, can then be obtained by instantiating each variable with an arbitrary ground term built from functions in the given set of clauses and, if there is no constant among these, an extra constant.

As proof systems, clausal tableaux and cut free analytic tableaux, as well as clausal tableaux with atomic cut (Sect. 6.3) and analytic tableaux with atomic cut, polynomially simulate each other if structure-preserving normal forms are permitted in both types of tableaux [55, p. 119].

5.2 The Connection Tableau Calculus

For Craig interpolation, our main interest in clausal tableaux is as proof objects that were delivered by an automated system, without caring about *how* they were found in proof search with some calculus. Nevertheless, we briefly present a clausal tableau calculus, the *connection tableau calculus*, referring to [57] for a comprehensive discussion. Differently from analytic tableaux, the initially given formula is not placed on the tableau, but kept separately, as a set of *input clauses*. The calculus builds the tableau by attaching copies, i.e., variants with fresh variables, of input clauses to the tableau and by instantiating variables in these tableau clauses. Instantiating is done with most general unifiers that equate a leaf literal with the complement of an ancestor literal, such that a branch gets closed. The rules of the calculus involve several nondeterministic selections. For completeness, a backtracking regime has to ensure that each possible selection is eventually made.

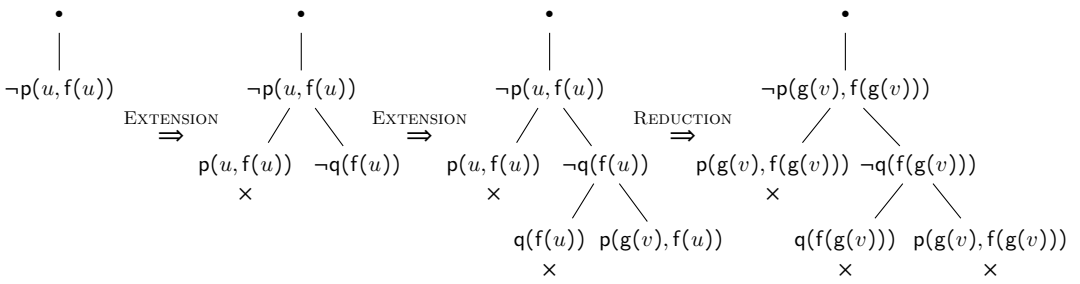
▶ **Definition 13** (Connection Tableau Calculus). *The* connection tableau calculus *consists of the following three rules.*

START*: If the tableau consists only of the root node, select a clause from the input clauses, make a fresh copy and attach children with its literals to the root.*

EXTENSION: *Select an open branch with leaf N and select an input clause $L' \vee C$ such that $\mathsf{lit}(N)$ and $\overline{\mathsf{lit}(L')}$ have a most general unifier. Make a fresh copy $L'' \vee C'$ of the clause and attach children with its literals to N. Let σ be the most general unifier of L and $\overline{L''}$ and apply σ to all literals of the tableau. The branch ending in the child corresponding to L'' is then closed.*

REDUCTION: *Select an open branch with leaf N and select an ancestor N' of N such that $\mathsf{lit}(L)$ and $\overline{\mathsf{lit}(L')}$ have a most general unifier σ. Apply σ to the tableau. The branch ending in N is then closed.*

▶ **Example 14.** Let $F = \{\neg\mathsf{p}(x, \mathsf{f}(x)),\ (\mathsf{p}(x,y) \vee \neg\mathsf{q}(y)),\ (\mathsf{q}(x) \vee \mathsf{p}(\mathsf{g}(y), x))\}$. The connection tableau calculus can build a closed clausal tableau for F as follows.



The first step, START, adds a copy $\neg\mathsf{p}(u, \mathsf{f}(u))$ of the first input clause. Then an EXTENSION step adds a copy $\mathsf{p}(x', y') \vee \neg\mathsf{q}(y')$ of the second input clause and applies the unifier $\{x' \mapsto u, y' \mapsto \mathsf{f}(u)\}$. A second EXTENSION step adds a copy $\mathsf{q}(x'') \vee \mathsf{p}(\mathsf{g}(v), x'')$ of the third input clause and applies the unifier $\{x'' \mapsto \mathsf{f}(u)\}$. Finally, a REDUCTION step with the node labeled by $\mathsf{p}(a, \mathsf{f}(u))$ as N and its ancestor labeled by $\neg\mathsf{p}(u, \mathsf{f}(u))$ as N' applies the unifier $\{u \mapsto \mathsf{g}(v)\}$. ⌟

If we add a further input clause $C = \mathsf{p}(x, y) \vee \neg\mathsf{r}(y)$ to F from Example 14 and select it as start clause, then no further rule is applicable. Also, if, as in the example, $\neg\mathsf{p}(x, \mathsf{f}(x))$ is selected as start clause but the first extension step is with C, then no further rule is applicable. In such cases alternate selections have to be explored. In implementations this is typically done with chronological backtracking embedded in an iterative deepening upon the depth of the tableau tree or some other measure (e.g., [58, Sect. 6.2]). With iterative deepening, if the minimal depth of a closed tableau for the given formula is n, proof search exhaustively explores for all $i \in \{0, \ldots, n-1\}$ the trees of depth up to i that are generated by the calculus rules. Finally, in iteration n it terminates with the first closed tableau it finds. In variations of this setup, the prover enumerates alternate closed tableaux, ordered by increasing depth, which is of interest for interpolation since different tableaux yield different interpolants.

A core idea of the connection method is to guide proof search by the *connections* in the given formula, pairs of literal occurrences with the same predicate but complementary polarity. This is reflected in the connection tableau calculus in that at each step except of START a

pair of literal occurrences is made complementary through unification, closing an open branch. The generated tableaux are thus *strongly connected*, which is defined as follows.

▶ **Definition 15** (Strong Connection Condition). *A clausal tableau is* strongly connected *iff every inner node with exception of the root has a node with complementary literal as a child.*

The strong connection condition does not affect completeness, i.e., whenever there is a closed clausal tableau for a set F of clauses, then there is a strongly connected closed clausal tableau for F [57, Sect. 5.2], [38, Sect. 4].

5.3 The Regularity Restriction

Regularity is an important restriction of clausal tableaux, defined as follows.

▶ **Definition 16** (Regular). *A clausal tableau is* regular *iff no node has an ancestor with the same literal.*

The number of nodes of a non-regular closed tableau can be strictly reduced with the following operation that again yields a closed tableau for the same set of clauses [57, Sect. 2]: *Select a node N with an ancestor N' such that both nodes are labeled with the same literal. Remove the edges originating*[1] *in the parent N'' of N and replace them with the edges originating in N.* Repeating this until the result is regular provides a polynomial proof transformation procedure, or tableau simplification, to achieve regularity. Any closed tableau for a given set F of clauses with a *minimal* number of nodes must be regular. Hence, enforcing regularity can be useful at proof search. For interpolation, regularity simplification reduces the size of the tableau if it is obtained from a prover that does not ensure regularity.

Regularity combined with the strong connection condition is complete [55, 38], i.e., if there is a closed tableau for a set of clauses, then there is one that is both regular and strongly connected. However, with respect to size the interplay of both restrictions is not smooth: clausal tableaux with the strong connection condition cannot polynomially simulate clausal tableaux without that condition, and regular clausal tableaux cannot simulate clausal tableaux with the strong connection condition [55, Sect. 3.4.1] [57, Chapter 7].

5.4 The Hyper Property

The *hyper* property [105] is a restriction of clausal tableaux with applications in strengthened variations of Craig-Lyndon interpolation and in the conversion of resolution proofs to clausal tableaux.

▶ **Definition 17** (Hyper). *A clausal tableau is* hyper *iff the nodes labeled with a negative literal are exactly the leaf nodes.*

[1] Tableau edges are considered as directed downward.

The name *hyper* alludes to hyperresolution and hypertableaux, which aim at narrowing proof search through a related restriction. Any closed clausal tableau for a set F of clauses can be converted to one with the hyper property, as shown with the algorithm presented below [105]. Its specification involves a further tableau property, *leaf-closed*, which, like regularity, can be achieved with a straightforward simplification.

▶ **Definition 18** (Target Node, Leaf-Closed). *A clausal tableau node is* closing *iff it has an ancestor with complementary literal. With a closing node N, a particular such ancestor is associated as* target of N*, written* $\mathsf{tgt}(N)$*. A tableau is* leaf-closing *iff all closing nodes are leaves. A closed tableau that is leaf-closing is called* leaf-closed.

▶ **Algorithm 19** (Hyper Conversion).

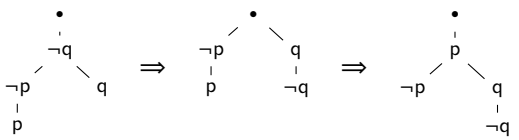
Input: A leaf-closed and regular clausal tableau.

Method: Repeat the following operations until the resulting tableau is hyper.

1. Let N' be the first node visited in pre-order[2] with a child that is an inner node with a negative literal label. Let N be the leftmost such child.
2. Create a fresh copy U of the subtree rooted at N'. In U remove the edges that originate in the node corresponding to N.
3. Replace the edges originating in N' with the edges originating in N.
4. For each leaf descendant M of N' with $\mathsf{lit}(M) = \overline{\mathsf{lit}(N)}$: Create a fresh copy U' of U. Change the origin of the edges originating in the root of U' to M.
5. Simplify the tableau to leaf-closing and regular form.

Output: A leaf-closed, regular and hyper clausal tableau whose clauses are clauses of the input tableau.

▶ **Example 20.** The following tableaux show a conversion with Algorithm 19 in two steps. ⌟



The algorithm is specified by destructive tableau manipulations. A *fresh copy* of an ordered tree T is an ordered tree T' with fresh nodes and edges, related to T through a bijection c such that any node N of T has the same literal label as node $c(N)$ of T' and such that the i-th edge originating in node N of T ends in node M iff the i-th edge originating in node $c(N)$ of T' ends in node $c(M)$. In each iteration the procedure chooses an inner node with negative literal label and modifies the tableau. At termination the tableau is then hyper. Since the procedure copies parts of subtrees it is not a polynomial operation but practical usefulness has been demonstrated [105, 41].

[2] *Pre-order tree traversal* is the method of depth-first traversal where the current node is visited *before* its subtrees are recursively traversed left-to-right.

5.5 Interpolant Calculation from a Clausal Tableau

The calculation of a ground Craig interpolant from a clausal tableau adapts the propositional core of interpolation methods for sequent systems and analytic tableaux [96, 92, 30] to the setting of clausal tableaux. To calculate a Craig-Lyndon separator for sets F, G of clauses we generalize clausal tableaux by an additional node label, *side*, which is shared by siblings and indicates whether a tableau clause is an instance of an input clause in F or in G.

▶ **Definition 21** (Two-Sided Clausal Tableau and Related Notions)**.**

(i) *Let F, G be sets of clauses. A* two-sided clausal tableau for F, G *(briefly* tableau for F, G*) is a clausal tableau for $F \cup G$ whose nodes N with exception of the root are labeled additionally with a* side $\mathsf{side}(N) \in \{\mathsf{F}, \mathsf{G}\}$*, such that the following conditions are met by all nodes N, N': (1) If N and N' are siblings, then $\mathsf{side}(N) = \mathsf{side}(N')$; (2) If N has a child N' with $\mathsf{side}(N') = \mathsf{F}$ ($\mathsf{side}(N') = \mathsf{G}$), then $\mathsf{clause}(N)$ is an instance of a clause in F (G). We also refer to the side of the children of a node N as* side of $\mathsf{clause}(N)$*.*

(ii) *For $\mathcal{A} \in \{\mathsf{F}, \mathsf{G}\}$ and all nodes N of a two-sided clausal tableau define*

$$\mathsf{path}_{\mathcal{A}}(N) \stackrel{\text{def}}{=} \{\mathsf{lit}(N') \mid N' \in \mathit{Path} \text{ and } \mathsf{side}(N') = \mathcal{A}\},$$

where Path is the union of $\{N\}$ and the set of the ancestors of N.

Example 26 in Sect. 5.6 shows examples of the defined notions. To integrate truth value simplifications EQ 9–EQ 12 into interpolant calculation from the very beginning, we define the following variations of conjunction and disjunction: For formulas $F_1, \ldots, F_n$ define $\bigwedge_{i=1}^{n} F_i$ ($\bigvee_{i=1}^{n} F_i$) as $\bot$ ($\top$) if at least one of the formulas F_i is identical to $\bot$ ($\top$), else as the conjunction (disjunction) of those formulas F_i that are not identical to $\top$ ($\bot$). We can now specify ground interpolant calculation inductively as a function of tableau nodes.

▶ **Definition 22** (Ground Interpolant Calculation from a Clausal Tableau)**.** *Let N be a node of a leaf-closed two-sided clausal ground tableau. The value of* $\mathsf{ipol}(N)$ *is a ground NNF formula, defined inductively as specified with the tables below, the left for the base case where N is a leaf, the right for the case where N is an inner node with children $N_1, \ldots, N_n$.*

$\mathsf{side}(N)$	$\mathsf{side}(\mathsf{tgt}(N))$	$\mathsf{ipol}(N)$
F	F	$\bot$
F	G	$\mathsf{lit}(N)$
G	F	$\overline{\mathsf{lit}(N)}$
G	G	$\top$

$\mathsf{side}(N_1)$	$\mathsf{ipol}(N)$
F	$\bigvee_{i=1}^{n} \mathsf{ipol}(N_i)$
G	$\bigwedge_{i=1}^{n} \mathsf{ipol}(N_i)$

The key property of $\mathsf{ipol}(N)$ is stated in the following lemma.

▶ **Lemma 23** (Invariant of Ground Interpolant Calculation from a Clausal Tableau). *Let F, G be sets of ground clauses and let N be a node of a leaf-closed two-sided clausal ground tableau for F, G. It then holds that*

(i) $F \cup \mathsf{path}_\mathsf{F}(N) \models \mathsf{ipol}(N)$ *and* $G \cup \mathsf{path}_\mathsf{G}(N) \models \neg\mathsf{ipol}(N)$

(ii) $\mathcal{L}\mathit{iterals}(\mathsf{ipol}(N)) \subseteq \mathcal{L}\mathit{iterals}(F \cup \mathsf{path}_\mathsf{F}(N)) \cap \overline{\mathcal{L}\mathit{iterals}(G \cup \mathsf{path}_\mathsf{G}(N))}$.

If N_0 is the tableau root, then $\mathsf{path}_\mathsf{F}(N_0) = \mathsf{path}_\mathsf{G}(N_0) = \{\}$. Hence:

▶ **Corollary 24** (Ground Interpolation with Clausal Tableaux). *Let F, G be sets of ground clauses and let N_0 be the root of a leaf-closed two-sided clausal ground tableau for F, G. Then* $\mathsf{ipol}(N_0)$ *is a Craig-Lyndon interpolant for $F, \neg G$.*

A ground interpolant according to Corollary 24, i.e., the value of ipol for the tableau root, is a ground NNF formula. Due to the integrated truth value simplification it does not have truth value constants as strict subformulas. The number of its literal occurrences is at most the number of tableau leaves.

5.6 The Two-Stage Approach: Overall Workflow

We illustrate the overall workflow of the two-stage approach to Craig interpolation with Algorithm 25 below, assuming first-order proving and ground interpolation is performed with a clausal tableau prover as described above. In Sect. 6 we will see that also a resolution-based system can be used, either directly or supplemented with a proof translation to clausal tableaux. Some subtasks can be performed in alternate ways, possibly with substantial effect on success of the prover as well as on size and shape of the resulting interpolant. The keyword Options introduces discussions of such alternatives.

▶ **Algorithm 25** (Craig-Lyndon Interpolation for First-Order Logic with Clausal Tableaux).

Input: First-order formulas F, G such that $F \models G$.

Output: A Craig-Lyndon interpolant H for F, G.

Method: The algorithm proceeds in the following phases.

I. **Eliminating Free Variables.** Replace free variables in F, G with fresh constants to obtain sentences F^S, G^S.

II. **Preprocessing: Skolemization, Clausification, Simplification.** Preprocess each of $F^S, \neg G^S$ separately, as outlined in Sect. 3 to obtain clause sets F', G' with fresh Skolem functions $\boldsymbol{f}', \boldsymbol{g}'$. In case F' (G') contains the empty clause, exit with $H \stackrel{\text{def}}{=} \bot$ ($H \stackrel{\text{def}}{=} \top$). Options: Various forms of Skolemization; various ways of CNF conversion, including structure-preserving forms, and various simplifications, constrained by Prop. 2.

III. **First-Order Proving.** Use a first-order prover to obtain a closed clausal tableau for $F' \cup G'$. Options: Choice and configuration of the first-order prover.

IV. Proof Grounding. Instantiate all variables in the tableau with ground terms. OPTIONS: The choice of the terms for instantiating has effect on the interpolant calculated in phase VI and on the lifting in phase VII. It is possible to use the same term for all variables, or different terms for different variables. Terms with F-only, G-only or FG-shared functions can be preferred.

V. Side Assignment. Attach side labels to the tableau clauses, according to whether they are an instance of a clause in F' or in G'. OPTIONS: It is possible that a clause in F' and a clause in G' are identical or have common instances. Thus, there are cases where a tableau clause can be assigned either one of the two side labels. The choice can have effect on the ground interpolant calculated in phase VI.

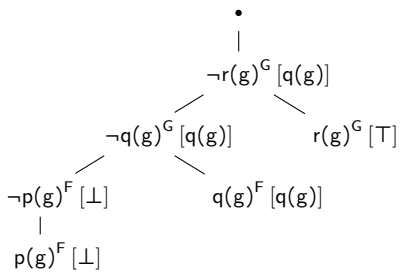
VI. Ground Interpolant Calculation. Calculate the ground interpolant H_{GRD} with the ipol function applied to the tableau. $\langle F^S, G^S, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}} \rangle$ is then a lifting base, where $\boldsymbol{f}, \boldsymbol{g}$ are determined from the Skolem functions $\boldsymbol{f}', \boldsymbol{g}'$ and $\mathcal{F}un(F^S), \mathcal{F}un(G^S)$, as described after Def. 7. OPTIONS: Incorporation of equivalence-preserving simplifications, including dedicated simplifications for equality, e.g., with $t \approx t \equiv \top$.

VII. Interpolant Lifting. Let H^S be the result of replacing terms in H_{GRD} and adding a quantifier prefix according to Theorem 9 with respect to the obtained lifting base. OPTIONS: Theorem 9 constrains the quantifier prefix by a partial order that may have different linear extensions.

VIII. Reintroducing Free Variables. Obtain the final result H by replacing in H^S any constants introduced in step I with the corresponding free variables.

The following simple example illustrates the steps of the two-stage approach to Craig interpolation with Algorithm 25.

▶ **Example 26.** Let $F \stackrel{\text{def}}{=} \forall x\, \mathsf{p}(x) \wedge \forall x\, (\neg \mathsf{p}(x) \vee \mathsf{q}(x))$ and let $G \stackrel{\text{def}}{=} \forall x\, (\neg \mathsf{q}(x) \vee \mathsf{r}(x)) \rightarrow \forall y\, \mathsf{r}(y)$. Since these formulas do not have free variables, we skip steps I and VIII. Step II, Skolemization and clausification applied separately to F and to $\neg G$, yields the first-order clause sets $F' = \{\mathsf{p}(x),\ (\neg \mathsf{p}(x) \vee \mathsf{q}(x))\}$ and $G' = \{(\neg \mathsf{q}(x) \vee \mathsf{r}(x)),\ \neg \mathsf{r}(\mathsf{g})\}$, where g is a Skolem constant. In step III, first-order proving, the connection tableau calculus (Def. 13) for start clause $\neg \mathsf{r}(\mathsf{g})$ yields the following leaf-closed tableau for $F' \cup G'$ (additional node annotations will be explained in a moment).



The calculus propagates the constant g through unification into variables so that in case of the example the tableau is already ground, leaving nothing to do for step IV, proof grounding.

Step V, side assignment, leaves no options as each tableau clause is either an instance of a clause in F' or of a clause in G', but never of a clause in both. Thus, the clauses of the tableau with side F are $\mathsf{p(g)}$ and $\neg\mathsf{p(g)} \vee \mathsf{q(g)}$, and the clauses with side G are $\neg\mathsf{q(g)} \vee \mathsf{r(g)}$ and $\neg\mathsf{r(g)}$. The respective side labels of the nodes are indicated as superscripts. To give examples for $\mathsf{path}_{\mathcal{A}}(N)$, let N be the bottom left node, which has literal $\mathsf{p(g)}$. It then holds that $\mathsf{path}_{\mathsf{F}}(N) = \{\neg\mathsf{p(g)}, \mathsf{p(g)}\}$ and $\mathsf{path}_{\mathsf{G}}(N) = \{\neg\mathsf{r(g)}, \neg\mathsf{q(g)}\}$.

We now have a two-sided leaf-closed ground tableau for F', G' and can, in step VI, calculate the ground interpolant H_{GRD} with the ipol function. The values of ipol for the individual nodes are annotated in brackets. Its value for the root is $H_{\text{GRD}} = \mathsf{q(g)}$. The tuple $\langle F, G, \{\}, \{\mathsf{g}\}, \mathsf{q(g)}\rangle$ forms an interpolant lifting base $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}}\rangle$. This holds in general for $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}}\rangle$ obtained with the described steps of Algorithm 25. In our example, we can verify this with $F_{\text{EXP}} = \mathsf{p}(u) \wedge (\neg\mathsf{p}(u) \vee \mathsf{q}(u))$, $G_{\text{EXP}} = \neg((\neg\mathsf{q}(v) \vee \mathsf{r}(v)) \wedge \neg\mathsf{r(g)})$ and $\eta = \{u \mapsto \mathsf{g},\ v \mapsto \mathsf{g}\}$. Finally, in step VII, interpolant lifting, we apply Theorem 9 and obtain $H = \forall v_1\, \mathsf{q}(v_1)$ as a Craig-Lyndon interpolant for F, G. ⌟

6 Ground Interpolation with Resolution

We discuss resolution as a technique for first-order theorem proving and ground interpolation in the two-stage approach, and relate it to clausal tableaux in these roles.

6.1 From a Deduction via a Deduction Tree to a Ground Deduction

We consider a simple sound and complete first-order resolution calculus, which we call $\mathcal{R}$. It has the two following two inference rules.

Binary Resolution $\dfrac{C \vee L \quad D \vee K}{(C \vee D)\sigma}$ where σ is the most general unifier of L and $\overline{K}$

Factoring $\dfrac{C \vee L \vee K}{(C \vee L)\sigma}$ where σ is the most general unifier of L and K

Clauses are considered here as *multisets* of literals. It is assumed that premises of binary resolution do not share variables, which is achieved by renaming variables if necessary. The conclusion of binary resolution is called *resolvent upon* L, K, the conclusion of factoring *factor with respect to* L, K. The notion of *proof* is captured in the following definition.

▶ **Definition 27** (Deduction). *Let $\mathcal{I}$ be a calculus characterized by a set of inference rules for clauses. An* $\mathcal{I}$-deduction *of a clause C from a set F of clauses is a sequence of clauses $C_1, \ldots, C_k = C$ such that each C_i is either in F (C_i is then called an* input clause*), or the conclusion of an inference rule of $\mathcal{I}$ for premises preceding C_i. An $\mathcal{I}$-deduction of the empty clause $\bot$ from F is called* $\mathcal{I}$-refutation*, or* $\mathcal{I}$-proof *of F.*

The $\mathcal{R}$-calculus is *sound*: if there is an $\mathcal{R}$-refutation of F, then F is unsatisfiable, which follows since for both inference rules the conclusion is entailed by the premises. The $\mathcal{R}$-calculus is also complete: if F is unsatisfiable, then there is an $\mathcal{R}$-refutation of F [17, 7].

▶ **Example 28.** Let F be the clause set $\{\neg\mathsf{p}(x) \vee \mathsf{p}(\mathsf{f}(x)),\ \mathsf{p}(\mathsf{g}(x)),\ \neg\mathsf{p}(\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{g}(x))))))\}$, which is unsatisfiable. The following table shows an $\mathcal{R}$-deduction of $\bot$ from F. We use $\mathsf{f}^2(_)$ as shorthand for $\mathsf{f}(\mathsf{f}(_))$, and analogously $\mathsf{f}^4(_)$ for $\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(_))))$.

Clause-Id	Clause	Justification
C_1	$\neg\mathsf{p}(x) \vee \mathsf{p}(\mathsf{f}(x))$	Input clause
C_2	$\mathsf{p}(\mathsf{g}(x))$	Input clause
C_3	$\neg\mathsf{p}(\mathsf{f}^4(\mathsf{g}(x)))$	Input clause
C_4	$\neg\mathsf{p}(x) \vee \mathsf{p}(\mathsf{f}^2(x))$	Resolvent of C_1 and C_1
C_5	$\neg\mathsf{p}(x) \vee \mathsf{p}(\mathsf{f}^4(x))$	Resolvent of C_4 and C_4
C_6	$\mathsf{p}(\mathsf{f}^4(\mathsf{g}(x)))$	Resolvent of C_2 and C_5
C_7	$\bot$	Resolvent of C_6 and C_3

Binary resolution and factoring can be combined into a single inference rule, as in the original presentation of resolution [85]. Numerous refinements of resolution aim at improving proof search by reducing the vast number of deductions that can be generated. Interpolant calculation, however, starts from a *given proof*. Many resolution refinements can be translated to our two basic rules. *Prover9* actually comes with a tool to convert its proofs to basic rules. Thus, interpolant calculation for just a basic form of resolution does not exclude employing advanced resolution refinements at *proof search*.

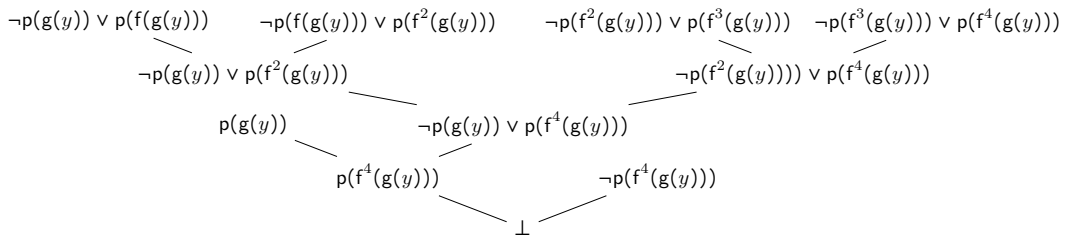
Our interpolant calculation operates on a resolution deduction that has only ground clauses. It is obtained via expanding the given first-order $\mathcal{R}$-deduction into a deduction *tree*.

▶ **Definition 29** (Deduction Tree). *An $\mathcal{I}$-*deduction tree *of a clause C is an (upward) tree with nodes labeled by clauses, such that the clause of the root is C and the clause of any inner node is the conclusion of an inference rule of $\mathcal{I}$ from the clauses of the parents as premises.*

Variables in a deduction tree have global scope, as in a clausal tableau, and differently from a deduction. Since the deduction tree expands the DAG structure of the premise/conclusion relationship represented by the deduction into a tree, the number of nodes of the deduction tree may be exponentially larger than the number of clauses of the deduction. The construction of a deduction tree from a given deduction can be sketched as follows. We start with creating the leaf nodes of the deduction tree, one leaf node for each *instance* of an input clause in the deduction, each leaf node with a *fresh copy* of the respective input clause such that no variables are shared between leaves. Then we proceed downwards to the root by attaching children according to the inferences in the deduction, however, now without renaming variables before applying inference rules. Instead, we compute the most general unifier and apply it to all variables in the tree under construction, such that also occurrences in ancestor nodes are substituted.[3]

[3] We assume here w.l.o.g. that for the most general unifier its domain and the set of variables occurring in its range are disjoint subsets of the set of variables in the unified terms [27, Rem. 4.2].

▶ **Example 30.** The $\mathcal{R}$-deduction of Example 28 expands into the following $\mathcal{R}$-deduction tree. We may assume that variable y stems from the sole involved copy of C_2.



We now move to *ground* resolution, with the ground resolution calculus $\mathcal{R}_{\text{GRD}}$ that operates on sets of ground clauses. It has the following two inference rules.

$$\textbf{Ground Resolution}\quad \frac{C \vee L \quad D \vee \overline{L}}{C \vee D} \qquad \textbf{Merging}\quad \frac{C \vee L \vee L}{C \vee L}$$

The conclusion of merging is called *merge with respect to* L. $\mathcal{R}_{\text{GRD}}$- and $\mathcal{R}$-deductions can be related via their deduction tree as follows.

▶ **Definition 31** ($\mathcal{R}_{\text{GRD}}$-Deduction for an $\mathcal{R}$-Deduction)**.** *Let F be a set of clauses. An $\mathcal{R}_{GRD}$-deduction $\mathcal{D}_{GRD}$ from a set F_{GRD} of ground clauses is said to be* for *an $\mathcal{R}$-deduction $\mathcal{D}$ from a set F of clauses if $\mathcal{D}_{GRD}$ can be obtained from $\mathcal{D}$ by expansion to an $\mathcal{R}$-Deduction tree, instantiating all variables in this $\mathcal{R}$-deduction tree by ground terms, which yields a $\mathcal{R}_{GRD}$-deduction tree, followed by converting this $\mathcal{R}_{GRD}$-deduction tree into a $\mathcal{R}_{GRD}$-deduction.*

Definition 31 is expressed in terms of transformations, which for practical interpolation have to be implemented. For instantiating with ground terms there are options as discussed for phase IV of Algorithm 25. The final conversion of the $\mathcal{R}_{\text{GRD}}$-deduction tree to a $\mathcal{R}_{\text{GRD}}$-deduction may be just a linearizing of the tree structure. In case different nodes are labeled with identical ground clauses, it might be shortened to a representation as DAG.

6.2 Interpolant Calculation from a Ground Deduction

To compute a Craig-Lyndon interpolant for first-order formulas F, G such that $F \models G$ on the basis of a resolution proof we have to build an interpolant lifting base $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{\text{GRD}} \rangle$.

▶ **Proposition 32** (Deduction and Lifting Base)**.** *Let F, G be formulas such that $F \models G$. Let F', G' are obtained from $F, \neg G$ by prenexing, Skolemization, and CNF transformation. Let $\mathcal{D}_{GRD}$ be an $\mathcal{R}_{GRD}$-proof for an $\mathcal{R}$-proof of $F' \cup G'$. Let $\boldsymbol{f}$ ($\boldsymbol{g}$) be the union of the Skolem functions $\boldsymbol{f}'$ ($\boldsymbol{g}'$) introduced for F (G) and the F-only (G-only) functions. If, so far, neither in $\boldsymbol{f}$, nor in $\boldsymbol{g}$, nor in the $FG-$shared functions is a constant, then add a fresh constant c_0 to either $\boldsymbol{f}$ or $\boldsymbol{g}$. Let F_{GRD} (G_{GRD}) be the input clauses of $\mathcal{D}_{GRD}$ that are instances of F' (G'). If H_{GRD} is a Craig-Lyndon interpolant of $F_{GRD}, \neg G_{GRD}$, then $\langle F, G, \boldsymbol{f}, \boldsymbol{g}, H_{GRD} \rangle$ is an interpolant lifting base.*

The construction of $\boldsymbol{f}$ and $\boldsymbol{g}$ is immediate from Prop. 32. To complete the lifting base we calculate a ground interpolant H_{GRD} from the ground deduction $\mathcal{D}_{\text{GRD}}$. To this end we enhance literal occurrences in proofs with a *provenance* label, analogous to the *side* in clausal tableaux.

▶ **Definition 33** (Provenance-Labeling and Related Notions)**.**

(i) *A clause (considered as multiset of literals) is* provenance-labeled *if each of its literals is associated with a* provenance label*, a nonempty subset of* $\{\mathsf{F},\mathsf{G}\}$*. A literal* L *with provenance label* $\mathcal{A}$ *is written* $L^{\mathcal{A}}$*.*

(ii) *A* provenance-labeled ground resolution proof *for sets* F, G *of ground clauses is a ground resolution proof for* $F \cup G$*, where each literal in an input clause from* F *has provenance label* $\{\mathsf{F}\}$*, each literal in an input clause from* G *has provenance label* $\{\mathsf{G}\}$*, and provenance labels are propagated as follows: For a resolvent* $C \vee D$ *the provenance labels are taken from* C, D *in the premises; for a merge* $C \vee L$*, the provenance label of* L *is the union of the provenance labels of the two merged occurrences of* L *in the premise, and the provenance labels of* C *are taken from* C *in the premise.*

(iii) *For* $\mathcal{A} \in \{\mathsf{F},\mathsf{G}\}$ *and provenance-labeled clause* $C = \bigvee_i^n L_i^{\mathcal{A}_i}$ *define*

$$\mathsf{subclause}_{\mathcal{A}}(C) \stackrel{\text{def}}{=} \bigvee_{\mathcal{A} \in \mathcal{A}_i} L_i.$$

▶ **Example 34.** Let $C = p^{\{\mathsf{F}\}} \vee q^{\{\mathsf{G}\}} \vee r^{\{\mathsf{F},\mathsf{G}\}} \vee s^{\{\mathsf{F}\}} \vee s^{\{\mathsf{G}\}}$, then $\mathsf{subclause}_{\mathsf{F}}(C) = p \vee r \vee s$ and $\mathsf{subclause}_{\mathsf{G}}(C) = q \vee r \vee s$. ⌟

▶ **Definition 35** (Ground Interpolant Calculation from a Resolution Proof)**.** *Let* C *be a clause in a provenance-labeled ground resolution proof for clausal ground formulas* F, G*. The value of* $\mathsf{ripol}(C)$ *is a ground NNF formula, defined inductively as follows. For the base cases where* C *is an input clause and the case where* C *is obtained by merging, the value of* $\mathsf{ripol}(C)$ *is specified in the following table.*

Case	$\mathsf{ripol}(C)$
C *is an input clause* $\bigvee_{i=1}^n L_i^{\{\mathsf{F}\}}$ *from* F	$\bot$
C *is an input clause* $\bigvee_{i=1}^n L_i^{\{\mathsf{G}\}}$ *from* G	$\top$
C *is obtained as merge from premise* D	$\mathsf{ripol}(D)$

For the case where $C \vee D$ *is obtained as resolvent from premises* $C \vee L^{\mathcal{A}}$ *and* $D \vee \overline{L}^{\mathcal{B}}$*, the value of* $\mathsf{ripol}(C \vee D)$ *is specified in the following table. It depends on the provenance labels* $\mathcal{A}, \mathcal{B}$ *of the literals resolved upon. We use the shorthands* $H_1 = \mathsf{ripol}(C \vee L^{\mathcal{A}})$ *and* $H_2 = \mathsf{ripol}(D \vee \overline{L}^{\mathcal{B}})$*. For two of the subcases, alternate possibilities are given.*

$\mathcal{A}$	$\mathcal{B}$	$\mathsf{ripol}(C \vee D)$	*Remark*
$\{\mathsf{F}\}$	$\{\mathsf{F}\}$	$H_1 \vee H_2$	
$\{\mathsf{F}\}$	$\{\mathsf{G}\}$	$H_1 \vee (L \wedge H_2)$	
$\{\mathsf{F}\}$	$\{\mathsf{G}\}$	$(L \vee H_1) \wedge H_2$	*Alternate possibility*
$\{\mathsf{F}\}$	$\{\mathsf{F},\mathsf{G}\}$	$H_1 \vee (L \wedge H_2)$	
$\{\mathsf{G}\}$	$\{\mathsf{G}\}$	$H_1 \wedge H_2$	
$\{\mathsf{G}\}$	$\{\mathsf{F},\mathsf{G}\}$	$H_1 \wedge (\overline{L} \vee H_2)$	
$\{\mathsf{F},\mathsf{G}\}$	$\{\mathsf{F},\mathsf{G}\}$	$(\overline{L} \wedge H_1) \vee (L \wedge H_2)$	
$\{\mathsf{F},\mathsf{G}\}$	$\{\mathsf{F},\mathsf{G}\}$	$(L \vee H_1) \wedge (\overline{L} \vee H_2)$	*Alternate equivalent possibility*

The key property of $\mathsf{ripol}(C)$ that holds for all clauses C of the provenance-labeled resolution proof is stated in the following lemma.

▶ **Lemma 36** (Invariant of Ground Interpolant Calculation from a Resolution Proof). *Let F, G be sets of ground clauses and let C be a provenance-labeled clause of a resolution proof for F, G. It then holds that*

(i) $F \models \mathsf{ripol}(C) \vee \mathsf{subclause}_{\mathsf{F}}(C)$ *and* $G \models \neg\mathsf{ripol}(C) \vee \mathsf{subclause}_{\mathsf{G}}(C)$.

(ii) $\mathcal{L}\mathit{iterals}(\mathsf{ripol}(C)) \subseteq \mathcal{L}\mathit{iterals}(F) \cap \overline{\mathcal{L}\mathit{iterals}(G)}$.

If C_0 is the empty clause $\bot$ at the root of the resolution proof, then $\mathsf{subclause}_{\mathsf{F}}(C_0) = \mathsf{subclause}_{\mathsf{G}}(C_0) = \bot$. Hence:

▶ **Corollary 37** (Ground Interpolation with Resolution). *Let F, G be clausal ground formulas and let C_0 be the empty clause obtained with a provenance-labeled ground resolution proof from F, G. Then $\mathsf{ripol}(C_0)$ is a Craig-Lyndon interpolant for $F, \neg G$.*

Definition 35 refines a well known interpolation system for propositional resolution called *HKPYM* by Bonacina and Johansson [14], after the initials of several authors who discovered or investigated it independently. Huang [43] uses it for first-order ground resolution proofs, like we do, but extended to paramodulation. He assumes merging implicitly with ground resolution. The essential difference is that Huang uses for *all* cases of ground resolution with exception of $\{\mathsf{F}\}\{\mathsf{F}\}$ and $\{\mathsf{G}\}\{\mathsf{G}\}$ the schema of our case $\{\mathsf{F},\mathsf{G}\}\{\mathsf{F},\mathsf{G}\}$, that is, $(\overline{L} \wedge H_1) \vee (L \wedge H_2)$. Similarly, HKPYM as defined in [14] uses for *all* these cases our *alternate possibility* for case $\{\mathsf{F},\mathsf{G}\}\{\mathsf{F},\mathsf{G}\}$. Hence, these methods do not construct Craig-*Lyndon* interpolants. Although Huang uses provenance to label literal *occurrences in the proof*, like we do, versions of HKPYM often use a labeling that just distinguishes on the basis of the two given interpolated formulas F, G whether an atom is FG-shared (is "transparent" or "grey"), F-only (has one of two "colors"), or G-only (has the other "color"). With this coarse labeling more literals may enter the interpolant, instead of truth value constants that could be eliminated by simplifying with EQ 9–EQ 14.

6.3 Deduction Trees and Clausal Tableaux

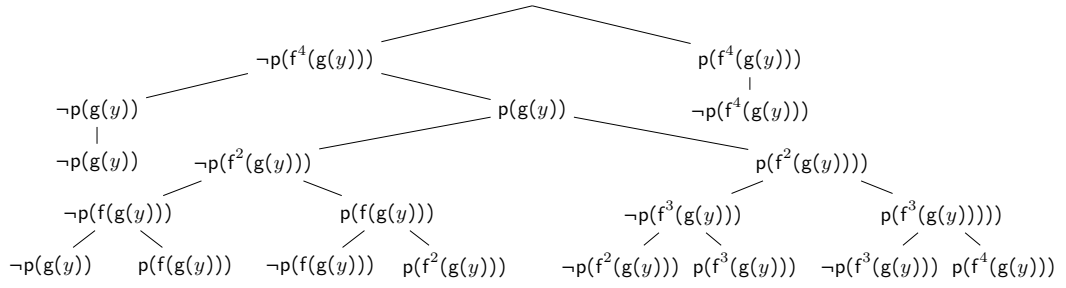
It is well known [57] that a resolution deduction tree of the empty clause $\bot$ represents a closed clausal tableaux in a specific form and vice versa. This correspondence is of interest for interpolation since it can be utilized in practice to convert resolution proofs to clausal tableaux and, moreover, indicates a systematization of resolution-based interpolation methods.

▶ **Definition 38** (Atomic Cut, Clausal Tableau in Cut Normal Form).

(i) *An* atomic cut *is a clause of the form* $\neg p(\boldsymbol{t}) \vee p(\boldsymbol{t})$.

(ii) *A closed clausal tableau is in* cut normal form *for a set of clauses* F *if for all inner nodes* N *whose children are not leaves* $\mathsf{clause}(N)$ *is an atomic cut, and for all inner nodes* N *whose children are leaves* $\mathsf{clause}(N)$ *is an instance of a clause in* F.

▶ **Example 39.** The following closed clausal tableau is in cut normal form. It represents the $\mathcal{R}$-deduction tree from Example 30.

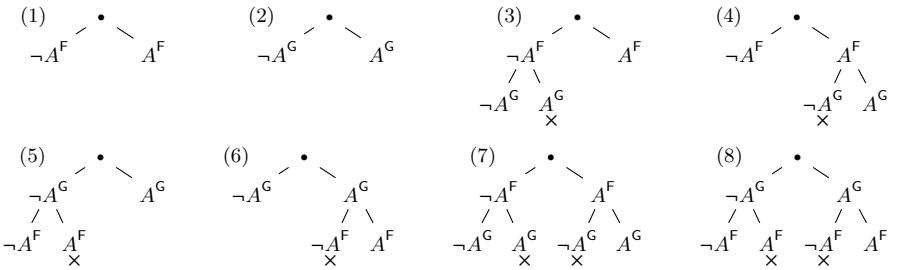


A clausal tableau in cut normal form for F is a special case of a clausal tableau *for* the formula $F \cup \bigcup_{p \in \mathcal{P}red(F)} \{\neg p(\boldsymbol{x}) \vee p(\boldsymbol{x})\}$, which is equivalent to F. Cut normal form can be seen as a notational variant of semantic trees [17, 57]. A given deduction tree of the empty clause $\bot$ from a set of clauses F can be converted in linear time to a closed clausal tableau in cut normal form for F as follows: (1) Delete the factoring or merging steps (considering factoring/merging as integrated into the resolution rule). (2) Remove the root label $\bot$. (3) Replace the labels representing the premises $C \vee L$ and $C \vee \overline{L}$ of a resolution step with the complements of the literals resolved upon, $\overline{L}$ and L, respectively. (4) Turn the tree upside down, such that the root is now at the top. (5) At each leaf that was labeled by an instance $L_1 \vee \ldots L_n \vee L$ of an input clause, and is now labeled by L, attach that clause, i.e., attach children labeled by $L_1, \ldots, L_n, L$.

In the resulting tableau, each involved instance of an input clause is falsified by the branch leading to it. That is, each literal in the clause has a complement in the branch. The converse translation of a closed clausal tableaux in cut normal form to a deduction tree of the empty clause $\bot$ is straightforward, with a potential quadratic increase in size because intermediate resolvents have to be attached [57]. Expressed in terms of semantic trees, this converse translation underlies a classic completeness proof of resolution [17].

The *hyper* property (Sect. 5.4) is, under assumption of regularity and leaf-closedness, incompatible with presence of an atomic cut: Consider an atomic cut $\neg A \vee A$ as tableau clause. If the tableau is hyper, then the node labeled with $\neg A$ is a leaf that is closed by an ancestor with label A. But this ancestor is also an ancestor of the other node of the atomic cut, labeled with A, which violates regularity. Thus, Algorithm 19, which converts a clausal tableau to hyper form, when applied to a closed clausal tableau in cut normal form yields a closed clausal tableau without atomic cuts. All tableau clauses of the converted tableau are instances of the input clauses at the leaves of the tableau in cut normal form. The hyper conversion thus "eliminates" the atomic cuts, which can be practically applied to convert resolution proofs to clausal tableaux without atomic cuts [105].

Since a closed clausal tableau in cut normal form is a special case of a closed clausal tableau we can also use it directly for interpolant calculation with the ipol operator. How should the side labeling be chosen? The clause instances at the leaves evidently receive the side label of the original clause of which they are an instance. The atomic cuts where the predicate is F-only get side F, the atomic cuts where the predicate is G-only get side G. For atomic cuts where the predicate is FG-shared we can take either side or we can stack instances with both side labels upon each other, leading the following eight combinations.



Some branches occurring at stacking are immediately closed, corresponding to a literal in the interpolant calculated by ipol. Each combination has exactly two open branches, such that the stacking effects no substantial increase of the tree size. Depending on the employed stacking schemas, interpolant calculation with ipol on the clausal tableau in cut normal form simulates different resolution-based calculi for ground interpolation, modulo commutativity of $\wedge$ and $\vee$ and truth value simplification. With stacking according to schema (7) we obtain Huang's method, with schema (8) HKPYM. We assume here that as target for closing a branch with a leaf from an instance of an input clause the node with the same side label as the leaf is selected. Schema (2) gives McMillan's method [69, 70, 14]. For a more detailed exposition and examples see [104]. Our calculation by ripol is obtained if the stacking schema is chosen according to the provenance label of the literal occurrences upon which the resolution step is performed. In case both are labeled by $\{\mathsf{F}, \mathsf{G}\}$, schema (7) is chosen. Schema (8) corresponds to the alternate possibility for this case in Def. 35. For the other cases, $H_1 \vee (L \wedge H_2)$ is simulated for positive (negative) L by schema (4) (schema (3)), and $(L \vee H_1) \wedge H_2$ for positive (negative) L by schema (5) (schema (6)).

Table 5 A first-order axiomatization of equality, shown as clauses. Substitutivity axioms SUBSTPRED$_{p,i}$ and SUBSTFUN$_{f,i}$ are for each predicate p (function f) with arity $n > 0$ in the vocabulary, and for each argument position $i \in \{1, \ldots, n\}$.

REFLEXIVITY	$x = x$
SYMMETRY	$x \neq y \vee y = x$
TRANSITIVITY	$x \neq y \vee y \neq z \vee x = z$
SUBSTPRED$_{p,i}$	$\neg p(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) \vee x \neq y \vee p(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)$
SUBSTFUN$_{f,i}$	$x \neq y \vee f(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)$

7 Craig-Lyndon Interpolation and Equality

Adding equality axioms, e.g., those from Table 5, is a simple way to incorporate equality into first-order logic. For provers with no dedicated equality support, adding such axioms is common practice. Equality-specific inferences often can be translated into inferences of an equality-free calculus, if the input is enriched by equality axioms. Paramodulation [84] provides an example. The same holds for superposition rules [8] since they are restrictions of paramodulation. Thus, a practical workflow for incorporating equality into interpolation with first-order provers is performing the proof search with dedicated equality support, followed by translating the proof to an equality-free calculus with axiomatized equality. Interpolant calculation is then applied to the translated proof, with equality handled as a predicate.

A Craig-Lyndon interpolant for formulas F, G of first-order logic with equality is then a Craig-Lyndon interpolant for formulas $E_F \wedge F$, $E_G \rightarrow G$ of first-order logic without equality, where E_F and E_G are conjunctions of equality axioms, say from Table 5. Axioms SUBSTPRED$_{p,i}$ are placed in E_F (E_G) if p is an F-only (G-only) predicate. In these axioms $=$ occurs only in negative polarity. The other equality axioms, which involve $=$ positively, can be placed arbitrarily in E_F or E_G. By controlling their placement, Craig-Lyndon interpolation yields interpolants according to the following theorem, due to Oberschelp [76] (see also [73]).

▶ **Theorem 40** (Oberschelp Interpolation). *Let F, G be formulas of first-order logic with equality such that $F \models G$. Then there exists a formula H of first-order logic with equality such that*

(1) *$F \models H$ and $H \models G$.*

(2) *$\mathcal{P}red^{\pm}(H) \subseteq \mathcal{P}red^{\pm}(F) \cap \mathcal{P}red^{\pm}(G)$, $\mathcal{V}ar(H) \subseteq \mathcal{V}ar(F) \cap \mathcal{V}ar(G)$, and $\mathcal{C}onst(H) \subseteq \mathcal{C}onst(F) \cap \mathcal{C}onst(G)$.*

(3) *$\mathcal{F}un(H) \subseteq \mathcal{F}un(F) \cup \mathcal{F}un(G)$.*

(4) *If $=$ occurs positively (negatively) in H, then $=$ occurs positively (negatively) in F (G).*

Proof. We obtain H as Craig-Lyndon interpolant for $E_F \wedge F$, $E_G \rightarrow G$, where E_F and E_G are defined as follows, considering conditions (4) contrapositively. If $=$ occurs only negatively or not at all in F (only positively or not at all in G), then E_F (E_G) is the conjunction of the axioms SUBSTPRED$_{p,i}$ for the F-only (G-only) predicates p, and E_G (E_F) is the conjunction

of the remaining equality axioms for the vocabularies of F, G. Else $=$ occurs positively in F or negatively in G. In this case let E_F (E_G) include the substitutivity axioms for F-only (G-only) predicates and functions, and place the remaining equality axioms for the vocabularies of F, G arbitrarily in E_F or E_G. ◀

Theorem 40 strengthens Craig-Lyndon interpolation only for formulas without functions except of constants. As shown by Brand in the context of his *modification method* [16] (see [23] for a summary and further references), the substitutivity axioms are dispensable for a set of clauses that are *flat*, i.e., all occurrences of non-variable terms are arguments to the equality predicate $=$. Any clause can be converted to an equivalent flat clause through "pulling out" terms with EQ 24. For example, $\mathsf{g}(\mathsf{a}, \mathsf{b}) = \mathsf{b}$ is equivalent to the flat clause $\mathsf{b} \neq x \vee \mathsf{a} \neq z \vee \mathsf{g}(z, x) = x$. Conversion to flat form introduces $=$ only with *negative* polarity. This can be utilized to show the following theorem, due to Fujiwara [31] and proven also by Motohashi [73], which strengthens both Craig-Lyndon interpolation and Theorem 40.

▶ **Theorem 41** (Oberschelp-Fujiwara Interpolation). *Let F, G be formulas of first-order logic with equality such that $F \models G$. Then there exists a formula H of first-order logic with equality such that*
(1) *$F \models H$ and $H \models G$.*
(2) *$\mathcal{V}oc^{\pm}(H) \subseteq \mathcal{V}oc^{\pm}(F) \cap \mathcal{V}oc^{\pm}(G)$.*
(3) *If $=$ occurs positively (negatively) in H, then $=$ occurs positively (negatively) in F (G).*

Proof. We obtain H as Craig-Lyndon interpolant for $E_F \wedge F'$, $E_G \rightarrow G'$, where F' and $\neg G'$ are flattened formulas that are equivalent to F and $\neg G$, respectively, and E_F and E_G are defined as follows. If $=$ occurs only negatively (positively) or not at all in F (G), then E_F (E_G) is $\top$ and E_G (E_F) is the conjunction of the equality axioms Reflexivity, Symmetry, and Transitivity. Else $=$ occurs positively in F or negatively in G. In this case, place Reflexivity, Symmetry, and Transitivity arbitrarily in E_F or E_G. ◀

8 Contributions of Automated Reasoning to Craig Interpolation

Methods from automated reasoning led to new strengthened variations of Craig interpolation, a new technique for Craig interpolation in a non-classical logic, techniques for putting specific strengthened variations of Craig interpolation into practice, and some general observations on Craig interpolation. We outline some of these results and provide references.

Craig-Lyndon Interpolation via Consequence Finding. First-order resolution is *complete for consequence finding* [54], i.e., whenever a clause C is entailed by a set of clauses F, then there is a deduction from F of a clause D that subsumes C. On this basis, Slagle [91] presented in 1970 variations of Craig-Lyndon interpolation. For propositional logic he strengthens Craig-Lyndon interpolation in that the interpolant H for sets F, G of clauses is a set of clauses *deduced by resolution* from F and that each clause of G is *subsumed* by some clause of H.

Craig Interpolation with Local Proofs. Aside of the two-stage approach, another approach to Craig interpolation with automated reasoning systems is pursued: *Local proofs*, where the basic idea is that all inference steps are *local*, that is, they do not involve both F-only and G-only symbols, which allows a particularly easy interpolant calculation. Jhala and McMillan [45] introduced this approach in 2006 for propositional logic. Subsequently it was generalized to first-order logic [71]. Kovács, Voronkov and their collaborators investigated it for first-order logic and implemented it in the *Vampire* prover [49, 42, 51]. Further aspects were shown by Bonacina and Johansson [15]. Interpolation with local proofs is incomplete for first-order logic [51]: Let $F = \forall x\, \mathsf{p}(\mathsf{a}, x)$ and let $G = \forall y\, \neg \mathsf{p}(y, \mathsf{b})$. Then $F \wedge G$ is unsatisfiable and $\exists y \forall x\, \mathsf{p}(y, x)$ is a separator for F, G. But, since a is F-only and b is G-only, any inference with F and G as premises is non-local. Results obtained with this approach include proof transformations to minimize calculated interpolants [42], transformations from non-local to local proofs, e.g., [42], and a proof that there is no lower bound on the number of quantifier alternations in separators for two universal first-order sentences [51].

Range-Restricted and Horn Interpolation. *Range-restriction* [98] is a syntactic property of first-order formulas that ensures domain independence [1] and constrains both the CNF and the DNF of the formula. A set of clauses is *Horn* if each clause has at most one positive literal. Range-restriction as well as the Horn property transfer from interpolated formulas to interpolants, if interpolation is performed with clausal tableaux that are *hyper* [105].

Practical Access Interpolation. Alternatively to range-restriction, constraining quantification to respect *binding patterns* [10] (see also Chapter 2 and Chapter 14) ensures domain independence. Interpolation with clausal tableaux that are hyper also transfers this constrained quantification from interpolated formulas to interpolants, in a workflow with structure-preserving normal forms and a variation of interpolant lifting that is interleaved with ground interpolation, because the constrained quantification has no prenex form [102].

Craig-Lyndon Interpolation for the Intermediate Logic of Here and There with Application to Synthesis of Answer Set Programs. *Strong equivalence* [60] is a useful notion of equivalence for answer set programs under stable model semantics [35]. It can be expressed as equivalence in the three-valued logic of here and there (HT), also known as Gödel's G_3. Although interpolation for this logic was known [64], practical construction of Craig interpolants from proofs has been shown only recently [41], based on an encoding of HT in classical logic that is conventionally used to prove strong equivalence with classical provers [62]. To construct a Craig-Lyndon interpolant for HT formulas F, G, first a classical Craig-Lyndon interpolant H'' for their encodings F', G' is constructed. Then, from H'' a stronger formula H' is constructed that can be decoded into a HT formula H, the desired Craig-Lyndon interpolant for F, G. The underlying argument uses that any proof of $F' \vDash H''$ can be modified to a proof of $F' \vDash H'$. A corresponding Beth theorem can be applied to synthesize answer set programs modulo strong equivalence with respect to a background program [41]. The interpolation technique has been adapted [106] to Mints' sequent system for HT [72].

9 Second-Order Quantifier Elimination

Second-order quantifier elimination is an approach to *uniform* interpolation based on *equivalence*, computing for a given second-order formula an *equivalent* formula of first-order logic with equality. We assume that the given second-order formula has the form

$$\exists p\, F, \tag{1}$$

where F is a first-order formula and p is a predicate. This is without loss of generality: first-order logic allows to represent n-ary functions by $n+1$-ary predicates, $\forall p\, F \equiv \neg\exists p\, \neg F$, and multiple occurrences of second-order quantifiers may be eliminated innermost-first. Also for F without equality, first-order formulas equivalent to $\exists p\, F$ may be with equality. Since not every second-order formula is equivalent to a first-order formula with equality, second-order quantifier elimination cannot succeed for all inputs. But there are formula classes for which it succeeds and elimination algorithms often succeed on problems from applications, sometimes subsuming special algorithms.

In this section we will summarize the two core approaches to second-order quantifier elimination. As a comprehensive source we recommend the monograph by Gabbay, Schmidt and Szałas [33]. As entry points for recent developments, we refer to the SOQE workshop series [48, 88] and to Chapter 15. The following examples illustrate various aspects of second-order quantifier elimination and indicate some potential applications.

▶ **Example 42.** Each of the following examples shows an equivalence of a second-order and a first-order formula. In all cases, the first-order formula can be obtained with known algorithms for second-order quantifier elimination.

(i) The following equivalence illustrates *forgetting*.

$$\exists q\, (\forall x\, (\mathsf{p}(x) \to q(x)) \wedge \forall x\, (q(x) \to \mathsf{r}(x))) \;\equiv\; \forall x\, (\mathsf{p}(x) \to \mathsf{r}(x)).$$

Forgetting about predicate q in a first-order formula F appears as elimination problem $\exists q\, F$. Predicate q does not occur in the result, while relationships between the other predicates are retained.

(ii) By the principle of *Leibniz' equality* two objects are *equal* if they are not distinguishable by properties. This can be directly phrased as follows.

$$\forall p\, (p(\mathsf{a}) \to p(\mathsf{b})) \;\equiv\; \mathsf{a} = \mathsf{b}.$$

(iii) *Predicate circumscription* [65, 24] is a technique from knowledge representation to enforce that predicate extensions are as small as possible. The *circumscription of predicate p in formula F* states that F holds and that there is no predicate p' such that $F\{p \mapsto p'\}$ holds and the extension of p' is strictly contained in that of p. This can be expressed as a second-order formula, shown here for $F = \mathsf{p}(\mathsf{a}) \wedge \mathsf{p}(\mathsf{b})$.

$$\begin{aligned} & \mathsf{p}(\mathsf{a}) \wedge \mathsf{p}(\mathsf{b}) \wedge \neg\exists p' [(p'(\mathsf{a}) \wedge p'(\mathsf{b})) \wedge \forall x\, (p'(x) \to \mathsf{p}(x)) \wedge \neg\forall x\, (\mathsf{p}(x) \to p'(x))] \\ \equiv\; & \mathsf{p}(\mathsf{a}) \wedge \mathsf{p}(\mathsf{b}) \wedge \forall x\, (\mathsf{p}(x) \to (x = \mathsf{a} \vee x = \mathsf{b})). \end{aligned}$$

(iv) Second-order quantifier elimination can be applied to automate correspondence theory in modal logic. Modal formulas can be expressed in *standard translation* as formulas of classical first-order logic. For a modal *axiom*, the unary predicates that correspond to parameters are then universally quantified. For example, axiom **T**, that is, $\Box p \rightarrow p$, is represented by the left side of the equivalence below. Its right side expresses the corresponding reflexivity of the accessibility relation as a first-order formula.

$$\forall p \forall w \, [\, \forall v \, (\mathsf{r}(w, v) \rightarrow p(v)) \rightarrow p(w)] \;\equiv\; \forall w \, \mathsf{r}(w, w)$$

(v) Let F, G be first-order formulas and let $P = \{p_1 \ldots p_n\}$ be a set of predicates. Then the *weakest sufficient condition of G with respect to F in terms of P* [61, 26, 33] is the weakest (w.r.t. entailment) formula H involving only predicates from P such that $F \wedge H \models G$. Formula H can be characterized semantically as $\forall q_1 \ldots \forall q_m \, (F \rightarrow G)$, where $\{q_1, \ldots, q_m\} = \mathcal{P}red(F \rightarrow G) \setminus P$. Second-order quantifier elimination then "computes" the weakest sufficient condition as a first-order formula. It can be viewed as an *abductive explanation*, the weakest explanation of observation G with background knowledge base F, where P is the set of *abducible predicates*. The following equivalence provides an example. We use as shorthands s, r for the 0-ary predicates $\mathsf{sprinklerWasOn}, \mathsf{rainedLastNight}$; g, b for the constants $\mathsf{grass}, \mathsf{boots}$; and w for the predicate *wet*.

$$\forall w \, ([(\mathsf{s} \rightarrow w(\mathsf{g})) \wedge (\mathsf{r} \rightarrow w(\mathsf{g})) \wedge (w(\mathsf{g}) \rightarrow w(\mathsf{b}))] \rightarrow w(\mathsf{b})) \;\equiv\; \mathsf{s} \vee \mathsf{r}.$$

The background knowledge base F is a conjunction of three implications. The observation G is $w(\mathsf{b})$. The abducibles P are the 0-ary predicates s and r. We obtain $H = \mathsf{s} \vee \mathsf{r}$ as abductive explanation, or weakest sufficient condition.

(vi) Second-order quantifier elimination succeeds for the class of relational monadic formulas, i.e., formulas with no functions and only unary predicates. An elimination algorithm, such as Behmann's method [9, 100], provides a *decision procedure*: eliminating all predicates yields a first-order formula with no predicates, but possibly with equality, whose status is easy to check: unsatisfiable, valid, or expressing a constraint on the cardinality of the domain. In the following example the domain must have at least two elements.

$$\exists p \, (\exists x \, p(x) \wedge \exists x \, \neg p(x)) \;\equiv\; \exists x \exists y \, x \neq y. \qquad \lrcorner$$

To see why second-order quantifier elimination, based on equivalence, produces uniform interpolants, based on notions of consequence and on vocabulary restrictions, we make the following observations.

▶ **Proposition 43.** *Let F, G be first order formulas and let $p_1, \ldots, p_n$ be predicates such that $G \equiv \exists p_1 \ldots \exists p_n \, G$. Then*

$$F \models G \;\; \textit{iff} \;\; \exists p_1 \ldots \exists p_n \, F \models G.$$

For uniform interpolation *syntactic* properties concerning the vocabulary of the involved formulas are considered. They relate to the *semantic* characterization of Prop. 43 as follows. The precondition $G \equiv \exists p_1 \ldots \exists p_n\, G$, which can be equivalently expressed as $\exists p_1 \ldots \exists p_n\, G \models G$, is implied by the syntactic property $\mathcal{P}red(G) \cap \{p_1, \ldots, p_n\} = \varnothing$. The second-order formula $\exists p_1 \ldots \exists p_n\, F$ satisfies the syntactic properties $\mathcal{P}red(\exists p_1 \ldots \exists p_n\, F) \cap \{p_1, \ldots, p_n\} = \varnothing$ and $\mathcal{P}red(\exists p_1 \ldots \exists p_n\, F) \subseteq \mathcal{P}red(F)$. If H is a first-order formula equivalent to $\exists p_1 \ldots \exists p_n\, F$ obtained with some second-order quantifier elimination method that does not introduce additional predicate symbols and eliminates all occurrences of $p_1, \ldots, p_n$, then the syntactic properties $\mathcal{P}red(H) \cap \{p_1, \ldots, p_n\} = \varnothing$ and $\mathcal{P}red(H) \subseteq \mathcal{P}red(F)$ hold. If H is, more generally, characterized just as a first-order formula equivalent to $\exists p_1 \ldots \exists p_n\, F$, then any Craig interpolant H' for F', H, where F' is F with $p_1, \ldots, p_n$ replaced by dedicated fresh predicate symbols, provides a first-order equivalent to $\exists p_1 \ldots \exists p_n\, F$ with the syntactic properties $\mathcal{P}red(H') \cap \{p_1, \ldots, p_n\} = \varnothing$ and $\mathcal{P}red(H') \subseteq \mathcal{P}red(F)$.

9.1 Direct Methods – Ackermann's Lemma and the DLS Algorithm

The family of *direct* methods for second-order quantifier elimination is characterized by equivalence-preserving rewriting to a form where all occurrences of second-order quantifiers are applied to formulas of certain shapes for which known equivalences permit schematic elimination. EQ 27 and EQ 28 are such schemas, known as the two versions of *Ackermann's lemma*, since they go back to Ackermann's 1935 paper [2] on elimination. Ackermann's lemma is the basis of the *DLS* algorithm [95, 24], named after its creators Doherty, Łukaszewicz, and Szałas. It initiated in the mid-1990s the *direct methods*, or *Ackermann approach* [87] to elimination. Before presenting DLS, we show some applications of Ackermann's lemma.

▶ **Example 44.**

(i) Example 42.i directly matches Ackermann's lemma in form EQ 28. If we flip the conjuncts on the left side of the example, it matches Ackermann's lemma in form EQ 27.

(ii) The left side of Example 42.ii is equivalent to $\neg\exists p\,[\forall x\,(x = \mathsf{a} \rightarrow p(x)) \wedge \neg p(\mathsf{b})]$, where the second-order subformula matches the left side of EQ 28.

(iii) The left side of Example 42.iv is equivalent to $\neg\exists w \exists p\,[\forall v\,(\mathsf{r}(w, v) \rightarrow p(v)) \wedge \neg p(w)]$, where the second-order subformula matches the left side of EQ 28. ⌟

▶ **Algorithm 45** (DLS)**.**

Input: A second-order formula $\exists p\, F$, where p is a predicate and F is a formula of first-order logic with equality.

Output: A formula of first-order logic with equality that is equivalent to the input formula or FAIL, indicating failure of the algorithm.

Method: The algorithm proceeds in four phases.

I. Preprocessing. Convert the input to the form

$$\exists \boldsymbol{x} \exists p \bigvee_{i=1}^{n} (A_i \wedge B_i), \tag{2}$$

where A_i and B_i are first-order formulas such that $-p \notin \mathcal{P}red^{\pm}(A_i)$ and $+p \notin \mathcal{P}red^{\pm}(B_i)$, for $i \in \{1, \ldots, n\}$. Specifically: (1.) Eliminate $\rightarrow$ and $\leftrightarrow$ (EQ 1, EQ 2, EQ 3). (2.) Remove void quantifiers (EQ 22). (3.) Move $\neg$ inwards until all occurrences precede atoms (EQ 4–EQ 5, EQ 15, EQ 16). (4.) Move $\forall$ to the right and $\exists$ to the left, as long as possible (EQ 17–EQ 20, renaming bound variables if necessary). (5.) Distribute all top-level conjunctions over the disjunctions occurring in conjuncts (EQ 8). If the result, after rearranging with respect to associativity and commutativity of $\wedge, \vee$, is not of the form (2), then exit with FAIL. Otherwise replace (2) with the equivalent formula

$$\exists \boldsymbol{x} \bigvee_{i=1}^{n} \exists p\, (A_i \wedge B_i) \tag{3}$$

and apply the next phases of the algorithm separately to each disjunct of (3). If this succeeds with formula R_i as first-order equivalent of the disjunct $\exists p\, (A_i \wedge B_i)$ for all $i \in \{1, \ldots, n\}$, then return $\exists \boldsymbol{x} \bigvee_{i=1}^{n} R_i$ as overall output of the algorithm.

II. Preparation for Ackermann's Lemma. The goal of this phase is to transform a formula $\exists p\, (A \wedge B)$, where $-p \notin \mathcal{P}red^{\pm}(A)$ and $+p \notin \mathcal{P}red^{\pm}(B)$ to a form that matches the left side of Ackermann's lemma in form EQ 27 or EQ 28. Both forms can always be obtained with distributing conjunctions over disjunctions (EQ 8), pulling out terms (EQ 24–EQ 23), Skolemization (EQ 29), and restoring implication (EQ 1). The algorithm computes both forms since the unskolemization in the next phase may succeed only for one, and also one form may be substantially smaller than the other.

III. Application of Ackermann's Lemma. Eliminate the second-order quantifier by rewriting with Ackermann's lemma and then try to unskolemize the Skolem functions introduced in the previous step on the basis of EQ 29, applied from right to left, with an unskolemization procedure. Unskolemization either succeeds or terminates with failure. If it fails for both forms of Ackermann's lemma, exit with FAIL. It if fails for one form, proceed with the other one. If it succeeds on both, pick one to proceed.

IV. Simplification. Simplify the result of the previous phase by equivalence-preserving transformations. Since EQ 24 and EQ 25 have been applied during the preparation for Ackermann's lemma for pulling out terms, their converse application to push in terms now often shortens the formula substantially. ⌟

If DLS exits with FAIL, this is either due to failure in phase I (*Preprocessing*) or due to failure of unskolemization in phase III (*Application of Ackermann's Lemma*) for both forms of Ackermann's lemma. Skolemization [33, 75] and unskolemization [66, 29, 18] are advanced topics on their own. Failure of DLS due to failure of unskolemization can be avoided by

introducing *branching quantifiers*, also known as *Henkin quantifiers*. The result of elimination is then, however, not necessarily a formula of classical first-order logic.

The output formula of DLS may be with equality, also in cases where the input formula is without equality. The following example illustrates introduction of equality and Skolemization in the preparation phase of DLS and unskolemization in the lemma application phase.

▶ **Example 46.** We apply DLS to the second-order formula $\exists p\,(A \wedge B)$, where $A = \forall x\,(\mathsf{q}(x) \rightarrow (p(x,\mathsf{a}) \vee p(x,\mathsf{b})))$ and $B = \forall x\,\neg p(x,\mathsf{c})$. This second-order formula already has the shape (3) such that the preprocessing phase has no effect. Preparation for Ackermann's lemma has then to be considered for the two forms of Ackermann's lemma, EQ 27 and EQ 28. Form EQ 27 requires to bring B into the shape $\forall xy\,(p(x,y) \rightarrow G)$. With EQ 25 we can rewrite B accordingly to $\forall xy\,(p(x,y) \rightarrow y \neq c)$. We can then apply EQ 27 with A in the role of F and obtain $A\{p \mapsto \lambda xy.y \neq c\}$, that is, $\forall x\,(\mathsf{q}(x) \rightarrow (\mathsf{a} \neq \mathsf{c} \vee \mathsf{b} \neq \mathsf{c}))$ as result.

Preparation for Ackermann's lemma in the form EQ 28 is more intricate, involving Skolemization. We convert A in the following steps to an equivalent formula, a conjunction where one conjunct has the shape $\forall xy\,(G \rightarrow p(x,y))$ as required by EQ 28 and the other conjunct has no occurrence of p.

1.		$\forall x\,[\mathsf{q}(x) \rightarrow (p(x,\mathsf{a}) \vee p(x,\mathsf{b}))]$	A
2.	$\equiv$	$\forall x\,[\mathsf{q}(x) \rightarrow \exists u\,((u = \mathsf{a} \vee u = \mathsf{b}) \wedge p(x,u))]$	by EQ 23
3.	$\equiv$	$\forall x\,[\mathsf{q}(x) \rightarrow \exists u\,((u = \mathsf{a} \vee u = \mathsf{b}) \wedge \forall y\,(y = u \rightarrow p(x,y)))]$	by EQ 24
4.	$\equiv$	$\forall x \exists u \forall y\,[\mathsf{q}(x) \rightarrow ((u = \mathsf{a} \vee u = \mathsf{b}) \wedge (y = u \rightarrow p(x,y)))]$	by prenexing
5.	$\equiv$	$\forall x \forall y\,[\mathsf{q}(x) \rightarrow ((s(x) = \mathsf{a} \vee s(x) = \mathsf{b}) \wedge (y = s(x) \rightarrow p(x,y)))]$	by EQ 29 (Skolemization)
6.	$\equiv$	$\exists s \forall x \forall y\,([\mathsf{q}(x) \rightarrow ((s(x) = \mathsf{a} \vee s(x) = \mathsf{b}))] \wedge$ $[(\mathsf{q}(x) \wedge y = s(x)) \rightarrow p(x,y)])$	by EQ 7
7.	$\equiv$	$\exists s\,(\forall x\,[\mathsf{q}(x) \rightarrow ((s(x) = \mathsf{a} \vee s(x) = \mathsf{b}))] \wedge$ $\forall xy\,[(\mathsf{q}(x) \wedge y = s(x)) \rightarrow p(x,y)]).$	by EQ 17, EQ 22

Let A' and B' be the two top conjuncts of the last of these formulas, i.e., $A' = \forall xy\,[(\mathsf{q}(x) \wedge y = s(x)) \rightarrow p(x,y)])$ and $B' = \forall x\,[\mathsf{q}(x) \rightarrow ((s(x) = \mathsf{a} \vee s(x) = \mathsf{b}))]$. Then $A \equiv \exists s\,(B' \wedge A')$ and $\exists p\,(A \wedge B) \equiv \exists s \exists p\,(A' \wedge B' \wedge B)$. Applying EQ 28 with A' in the role of $\forall xy\,(G \rightarrow p(x,y))$ and $B' \wedge B$ in the role of F yields $\exists s\,(B' \wedge B)\{p \mapsto \lambda xy.(\mathsf{q}(x) \wedge y = s(x))\}$, that is,

$$\exists s\,(\forall x\,[\mathsf{q}(x) \rightarrow ((s(x) = \mathsf{a} \vee s(x) = \mathsf{b}))] \wedge \forall x\,\neg(\mathsf{q}(x) \wedge \mathsf{c} = s(x))).$$

With EQ 17 and EQ 29, now applied right-to-left as unskolemization, we obtain

$$\forall x \exists y\,([\mathsf{q}(x) \rightarrow ((y = \mathsf{a} \vee y = \mathsf{b}))] \wedge \neg(\mathsf{q}(x) \wedge \mathsf{c} = y)),$$

which is equivalent to $\forall x\,(\mathsf{q}(x) \rightarrow (\mathsf{a} \neq \mathsf{c} \vee \mathsf{b} \neq \mathsf{c}))$, the result obtained before with Ackermann's lemma in the form EQ 27. DLS can then pick and return this shorter representation. ⌟

Comprehensive examples of DLS are given in [95, 24]. Details of DLS can be varied and refined, as discussed by Gustafsson [37] and Conradie [18], who also identifies a class of

formulas on which DLS succeeds. To let DLS succeed on all inputs $\exists p\, F$ for propositional F, distribution of disjunction over conjunctions (EQ 7) has to be incorporated [100].

Behmann's method [9, 100] is similar to DLS but restricted to relational monadic formulas, for which it guarantees success. It is based on a simple special case of Ackermann's lemma: $\exists p\,[\forall x\,(p(x) \vee F) \wedge \forall x\,(G \vee \neg p(x))] \equiv \forall x\,(F \vee G)$. Ackermann's works on elimination include a variation of unskolemization for predicates that is shown here as EQ 30 [3]. Applied from right to left it reduces the arity of predicates, ideally leading to relational monadic form, which then makes Behmann's method applicable. Some modern applications of this approach are discussed in [100].

If the elimination result is generalized to a formula of classical fixpoint logic, then Ackermann's lemma can be generalized by replacing condition $p \notin \mathcal{P}red(G)$ with $-p \notin \mathcal{P}red^{\pm}(G)$, i.e., allowing in G occurrences of p provided these are positive. This generalization of Ackermann's lemma is due to Nonnengart and Szałas [74]. In classical fixpoint logic atomic formulas can be fixpoint formulas $\textsc{Gfp}_{p,\boldsymbol{x}}G(\boldsymbol{t})$ and $\textsc{Lfp}_{p,\boldsymbol{x}}G(\boldsymbol{t})$, for *greatest* and *least fixpoint*. Our notation $F\{p \mapsto \lambda\boldsymbol{x}.G\}$ for substitution of a predicate p by a formula $\lambda\boldsymbol{x}.G$ is extended to substitution of a predicate p by a fixpoint formula $\textsc{Gfp}_{p,\boldsymbol{x}}G$ or $\textsc{Lfp}_{p,\boldsymbol{x}}G$, where operators $\textsc{Gfp}, \textsc{Lfp}$ bind the variables $\boldsymbol{x}$ in the same way as λ. The expressions $\textsc{Gfp}_{p,\boldsymbol{x}}G(\boldsymbol{t})$ and $\textsc{Lfp}_{p,\boldsymbol{x}}G(\boldsymbol{t})$ denote $\textsc{Gfp}_{p,\boldsymbol{x}}G$ and $\textsc{Lfp}_{p,\boldsymbol{x}}G$, respectively, applied to the tuple $\boldsymbol{t}$ of terms. For example, $\mathsf{p(a)}\{\mathsf{p} \mapsto \textsc{Gfp}_{\mathsf{p},x}G\} = \textsc{Gfp}_{\mathsf{p},x}G(\mathsf{a})$. The semantics of fixpoint formulas can be specified as follows: $\textsc{Gfp}_{p,\boldsymbol{x}}G(\boldsymbol{t})$ is true iff $\textsc{Gfp}_{p,\boldsymbol{x}}G\{p \mapsto r\}(\boldsymbol{t})$ is true, where r is the *greatest* (w.r.t. $\subseteq$) relation satisfying $\forall\boldsymbol{x}\,(r(\boldsymbol{x}) \equiv G\{p \mapsto r\})$. Analogously, $\textsc{Lfp}_{p,\boldsymbol{x}}G(\boldsymbol{t})$ is true iff $\textsc{Lfp}_{p,\boldsymbol{x}}G\{p \mapsto r\}(\boldsymbol{t})$ is true, where r is the *smallest* (w.r.t. $\subseteq$) relation satisfying $\forall\boldsymbol{x}\,(r(\boldsymbol{x}) \equiv G\{p \mapsto r\})$.

▶ **Theorem 47** (Fixpoint Generalization of Ackermann's Lemma [74]). *Let F be a formula of first-order logic with equality, let p be a predicate and let G be a formula such that $-p \notin \mathcal{P}red^{\pm}(G)$ and such that no free variables of G are bound by a quantifier in F. Then*

(i) *If $-p \notin \mathcal{P}red^{\pm}(F)$, then* $\exists p\,[\forall\boldsymbol{x}\,(p(\boldsymbol{x}) \to G) \wedge F] \equiv F\{p \mapsto \textsc{Gfp}_{p,\boldsymbol{x}}G\}$.

(ii) *If $+p \notin \mathcal{P}red^{\pm}(F)$, then* $\exists p\,[\forall\boldsymbol{x}\,(G \to p(\boldsymbol{x})) \wedge F] \equiv F\{p \mapsto \textsc{Lfp}_{p,\boldsymbol{x}}G\}$.

DLS* [25] is an extension of DLS that takes this fixpoint generalization of Ackermann's lemma into account. Ackermann's lemma (EQ 27, EQ 28) as well as its fixpoint generalization can be generalized by considering instead of the polarity of p in F that p is *monotone* (*down-monotone*) in F and, for the fixpoint version, that p is monotone in G [34, 33].

9.2 Predicate Elimination with Resolution – The SCAN Algorithm

The *SCAN* algorithm for second-order quantifier elimination was introduced in 1992 by Gabbay and Ohlbach [32]. Its name is an acronym of *"Synthesizing Correspondence Axioms for Normal logics"*. Actually, the method was a re-discovery of a result by Ackermann from 1935 [2]. Our exposition is oriented at the monograph by Gabbay, Schmidt and Szałas [33]. The general idea of SCAN is to generate sufficiently many logical consequences of the

Table 6 The constraint resolution calculus C.

Deduction	$\dfrac{N}{N \cup \{C\}}$	where C is a C-resolvent or a C-factor of premises in N
Purification	$\dfrac{N \cup \{C \vee (\neg)p(\boldsymbol{s})\}}{N}$	if p is a non-base predicate and no non-redundant inferences with respect to the particular literal $(\neg)p(\boldsymbol{s})$ in the premise $\{C \vee (\neg)p(\boldsymbol{s})\}$ and the rest of the clauses in N can be performed

Table 7 The inference rules of C.

C-Resolution	$\dfrac{C \vee p(\boldsymbol{s}) \quad D \vee \neg p(\boldsymbol{t})}{C \vee D \vee \boldsymbol{s} \not\approx \boldsymbol{t}}$	provided p is a non-base predicate, the two premises have no variables in common and are distinct clauses
(Positive) C-Factoring	$\dfrac{C \vee p(\boldsymbol{s}) \vee p(\boldsymbol{t})}{C \vee p(\boldsymbol{s}) \vee \boldsymbol{s} \not\approx \boldsymbol{t}}$	provided p is a non-base predicate

given second-order formula such that all further consequences that can be generated from consequences with predicates to be eliminated are *redundant*. The set of consequences without predicates to be eliminated is then equivalent to the given second-order formula.

▶ **Algorithm 48** (SCAN)**.**

INPUT: A second-order formula $\exists p_1 \ldots \exists p_n\, F$, where $p_1, \ldots, p_n$ are predicates and F is a formula of first-order logic with equality.

OUTPUT: The algorithm does not terminate for all inputs. If it terminates, the output is a formula of first-order logic with equality that is equivalent to the input formula or FAIL, indicating failure of the algorithm.

METHOD: The algorithm proceeds in three stages.

I. **Clausification.** The usual CNF conversion for first-order formulas, including Skolemization, is applied to the first-order component F of the input. Its result is a set N of clauses such that $\exists \boldsymbol{f} \forall \boldsymbol{x}\, N \equiv F$, where $\boldsymbol{f}$ are the Skolem functions introduced at the conversion and $\boldsymbol{x}$ are the free variables of N.

II. **Constraint Resolution.** This stage operates on the clause set N obtained in the previous stage. Predicates $p_1, \ldots, p_n$ are distinguished as *non-base* predicates. The calculus C (Table 6) is applied to N to generate a set N_∞ of clauses such that none of the non-base predicates occurs in N_∞ and N_∞ is equivalent to $\exists p_1 \ldots \exists p_n\, \exists \boldsymbol{f} \forall \boldsymbol{x}\, N$. If this stage terminates, the obtained N_∞ is finite.

III. **Unskolemization.** Apply unskolemization, for example with McCune's algorithm [66], to N_∞ to eliminate the Skolem functions $\boldsymbol{f}$ that were introduced at clausification. If this succeeds, return the result of unskolemization, else, exit with FAIL. ⌟

SCAN may fail either in stage II due to non-termination of C-resolution or in stage III due to failure of unskolemization. The *deduction* rule of calculus C computes new clauses

using the inference rules *C-resolution* and *C-factoring* (Table 7). As usual for resolution, premises are assumed to be normalized by variable renaming such that they have no shared variables. The role of unification in resolution is in constraint resolution taken by adding negated equality literals, "constraints", to the conclusion.

Theorem 49 below characterizes correctness for constraint resolution with the C calculus, which includes the properties required in the *Constraint Resolution* stage of SCAN. The theorem statement uses the terminology from the framework for saturation-based proving by Bachmair and Ganzinger [7]. A central notion is the property *redundant*, which can hold for inferences and for clauses. We may assume the so-called *trivial redundancy criterion*, where an inference is *redundant* in a clause set N if its conclusion is in N, and a clause is never considered as *redundant*. Optionally, to take account of equivalence-preserving deletion and reduction rules, e.g., deletion of tautological or subsumed clauses, which can be freely added to C without compromising correctness, other redundancy criteria can be employed, where also *clauses* may be classified as redundant [33, 7]. A clause set N is *C-saturated up to redundancy* if all inferences with non-redundant premises from N are redundant in N. A clause set N is *C-closed* if N is C-saturated up to redundancy and the *purification* rule is not applicable. A *C-derivation* is a (possibly infinite) sequence $N_0, N_1, \ldots$ of clause sets such that for every $i \geq 0$, N_{i+1} is obtained from N by the application of a rule in C. The *limit* of a C-derivation is the set $N_\infty \stackrel{\text{def}}{=} \bigcup_{j\geq 0} \bigcap_{k\geq j} N_k$ of persisting clauses. A C-derivation $N(= N_0), N_1, \ldots$ from N is *fair* iff the conclusion of every non-redundant inference from non-redundant premises in N_∞ is in some N_j. Intuitively, fairness means that no non-redundant inferences are delayed indefinitely. We are now ready to state the correctness theorem C, which underlies SCAN.

▶ **Theorem 49** (Correctness of Constraint Resolution with C [32, 33])**.** *Let N be a set of clauses and suppose $p_1, \ldots, p_k$ are distinguished as non-base predicates in N. Let $N(= N_0), N_1, \ldots$ be a fair C-derivation from N with limit N_∞. Then (1) N_∞ is C-closed; (2) None of the non-base predicates occurs in N_∞; (3) N_∞ is equivalent to $\exists p_1 \ldots \exists p_n\, N$.*

The set N_∞ may be infinite. If C-resolution terminates, then N_∞ is finite and can be passed to the unskolemization stage of SCAN. Ackermann [2] actually considered infinite sets N_∞ as elimination results. Like the output of DLS, the output of SCAN may be with equality, also in cases where the input formula is without equality. The following simple example illustrates the three stages of SCAN.

▶ **Example 50.** Consider the second-order formula $\exists q\,[(\mathsf{p}(\mathsf{a}) \to q(\mathsf{a})) \wedge (q(\mathsf{b}) \to \exists x\, \mathsf{r}(x))]$. Clausification of its first order component yields the following clauses, where s is a Skolem constant.

C_1	$\neg\mathsf{p}(\mathsf{a}) \vee q(\mathsf{a})$	Input clause
C_2	$\neg q(\mathsf{b}) \vee \mathsf{r}(s)$	Input clause

We now perform constraint resolution with the non-base predicate q. A C-resolution deduction step adds the following clause.

C_3	$\neg\mathsf{p}(\mathsf{a}) \vee \mathsf{r}(s) \vee \mathsf{a} \neq \mathsf{b}$	C-resolvent of C_1 and C_2

A purification step then deletes C_1, and a second purification step deletes C_2. We thus leave the constraint resolution stage with the singleton set containing C_3 as result N_∞. Unskolemization then gives us the first-order formula $\exists x\,(\neg \mathsf{p}(\mathsf{a}) \vee \mathsf{r}(x) \vee \mathsf{a} \neq \mathsf{b})$ as final result of the second-order quantifier elimination by SCAN. This formula may be rearranged as $(\mathsf{p}(\mathsf{a}) \wedge \mathsf{a} = \mathsf{b}) \rightarrow \exists x\, \mathsf{r}(x)$. ⌟

Refinements and variations of SCAN are discussed in [29, 77, 99, 36, 33]. If the C calculus is equipped with deletion of subsumed clauses, then SCAN is complete for the case where the quantified predicates are nullary [33]. An adaptation of SCAN for modal logics is complete for Sahlqvist formulas [36].

Acknowledgments

The author thanks Wolfgang Bibel, Patrick Koopmann and Philipp Rümmer for their valuable comments and suggestions. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley, 1995.

2 Wilhelm Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Math. Ann.*, 110:390–413, 1935. doi:10.1007/BF01448035.

3 Wilhelm Ackermann. Zum Eliminationsproblem der mathematischen Logik. *Math. Ann.*, 111:61–63, 1935. doi:10.1007/BF01472201.

4 Owen L. Astrachan. METEOR: Exploring model elimination theorem proving. *J. Autom. Reasoning*, 13(3):283–296, 1994. doi:10.1007/BF00881946.

5 Matthias Baaz, Uwe Egly, and Alexander Leitsch. Normal form transformations. In John Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 1, chapter 5, pages 273–333. Elsevier, 2001. doi:10.1016/B978-044450813-3/50007-2.

6 Matthias Baaz and Alexander Leitsch. *Methods of Cut-Elimination*. Springer, 2011. doi:10.1007/978-94-007-0320-9.

7 Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 1, chapter 2, pages 19–99. Elsevier, 2001. doi:10.1016/B978-044450813-3/50004-7.

8 Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic paramodulation and superposition. In Deepak Kapur, editor, *CADE-11*, volume 607 of *LNCS (LNAI)*, pages 462–476, 1992. doi:10.1007/3-540-55602-8_185.

9 Heinrich Behmann. Beiträge zur Algebra der Logik, insbesondere zum Entscheidungsproblem. *Math. Ann.*, 86(3–4):163–229, 1922. doi:10.1007/BF01457985.

10 Michael Benedikt, Julien Leblay, Balder ten Cate, and Efthymia Tsamoura. *Generating Plans from Proofs: The Interpolation-based Approach to Query Reformulation*. Morgan & Claypool, 2016. doi:10.1007/978-3-031-01856-5.

11 Wolfgang Bibel. *Automated Theorem Proving*. Vieweg, Braunschweig, 1987. First edition 1982. doi:10.1007/978-3-322-90102-6.

12 Wolfgang Bibel and Jens Otten. From Schütte's formal systems to modern automated deduction. In Reinhard Kahle and Michael Rathjen, editors, *The Legacy of Kurt Schütte*, chapter 13, pages 215–249. Springer, 2020. doi:10.1007/978-3-030-49424-7_13.

13 Jasmin Christian Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *J. Formaliz. Reason.*, 9(1):101–148, 2016. doi:10.6092/ISSN.1972-5787/4593.

14 Maria Paola Bonacina and Moa Johansson. Interpolation systems for ground proofs in automated deduction: a survey. *J. Autom. Reasoning*, 54(4):353–390, 2015. doi:10.1007/s10817-015-9325-5.

15 Maria Paola Bonacina and Moa Johansson. On interpolation in automated theorem proving. *J. Autom. Reasoning*, 54(1):69–97, 2015. doi:10.1007/s10817-014-9314-0.

16 D. Brand. Proving theorems with the modification method. *SIAM J. of Computing*, 4(4):412–430, 1975. doi:10.1137/0204036.

17 Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Automated Theorem Proving*. Academic Press, 1973.

18 Willem Conradie. On the strength and scope of DLS. *J. Applied Non-Classical Logic*, 16(3–4):279–296, 2006. doi:10.3166/jancl.16.279-296.

19 William Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *J. Symb. Log.*, 22(3):250–268, 1957. doi:10.2307/2963593.

20 William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. Symb. Log.*, 22(3):269–285, 1957. doi:10.2307/2963594.

21 William Craig. Elimination problems in logic: A brief history. *Synthese*, 164(3):321–332, 2008. doi:10.1007/s11229-008-9352-4.

22 William Craig. The road to two theorems of logic. *Synthese*, 164(3):333–339, 2008. doi:10.1007/s11229-008-9353-3.

23 Anatoli Degtyarev and Andrei Voronkov. Equality reasoning in sequent-based calculi. In Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 1, chapter 10, pages 611–706. Elsevier, 2001. doi:10.1016/B978-044450813-3/50012-6.

24 Patrick Doherty, Witold Łukaszewicz, and Andrzej Szałas. Computing circumscription revisited: A reduction algorithm. *J. Autom. Reasoning*, 18(3):297–338, 1997. doi:10.1023/A:1005722130532.

25 Patrick Doherty, Witold Łukaszewicz, and Andrzej Szałas. General domain circumscription and its effective reductions. *Fundam. Informaticae*, 36(1):23–55, 1998. doi:10.3233/FI-1998-3612.

26 Patrick Doherty, Witold Łukaszewicz, and Andrzej Szałas. Computing strongest necessary and weakest sufficient conditions of first-order formulas. In Bernhard Nebel, editor, *IJCAI-01*, pages 145–151. Morgan Kaufmann, 2001.

27 Elmar Eder. Properties of substitutions and unification. *J. Symb. Comput.*, 1(1):31–46, 1985. doi:10.1016/S0747-7171(85)80027-4.

28 Niklas Eén and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. In Fahiem Bacchus and Toby Walsh, editors, *SAT '05*, volume 3569 of *LNCS*, pages 61–75, 2005. doi:10.1007/11499107_5.

29 Thorsten Engel. Quantifier elimination in second-order predicate logic. Master's thesis, Max-Planck-Institut für Informatik, Saarbrücken, 1996.

30 Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 2nd edition, 1996. doi:10.1007/978-1-4612-2360-3.

31 Tsuyoshi Fujiwara. A variation of Lyndon-Keisler's homomorphism theorem and its applications to interpolation theorems. *J. Math. Soc. Japan*, 30(2):287–302, 1978. doi:10.2969/jmsj/03020287.

32 Dov Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second-order predicate logic. In Bernhard Nebel, Charles Rich, and William R. Swartout, editors, *KR'92*, pages 425–435. Morgan Kaufmann, 1992.

33 Dov M. Gabbay, Renate A. Schmidt, and Andrzej Szałas. *Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. College Publications, 2008.

34 Dov M. Gabbay and Andrzej Szałas. Second-order quantifier elimination in higher-order contexts with applications to the semantical analysis of conditionals. *Studia Logica*, 87(1):37–50, 2007. doi:10.1007/S11225-007-9075-4.

35 Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *ICLP/SLP 1988*, pages 1070–1080, Cambridge, MA, 1988. MIT Press.

36 Valentin Goranko, Ullrich Hustadt, Renate A. Schmidt, and Dimiter Vakarelov. SCAN is complete for all Sahlqvist formulae. In Rudolf Berghammer, Bernhard Möller, and Georg Struth, editors, *RelMiCS 7*, volume 3051 of *LNCS*, pages 149–162, 2004. doi:10.1007/978-3-540-24771-5_13.

37 Joakim Gustafsson. An implementation and optimization of an algorithm for reducing formulae in second-order logic. Technical Report LiTH-MAT-R-96-04, Univ. Linköping, 1996.

38 Reiner Hähnle. Tableaux and related methods. In Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 1, chapter 3, pages 101–178. Elsevier, 2001. doi:10.1016/b978-044450813-3/50005-9.

39 John Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009. doi:10.1017/CBO9780511576430.

40 Jacques Herbrand. *Recherches sur la théorie de la démonstration*. PhD thesis, University of Paris, 1930.

41 Jan Heuer and Christoph Wernhard. Synthesizing strongly equivalent logic programs: Beth definability for answer set programs via Craig interpolation in first-order logic. In Christoph Benzmüller, Marjin J. H. Heule, and Renate A. Schmidt, editors, *IJCAR 2024*, volume 14739 of *LNCS (LNAI)*, pages 172–193. Springer, 2024. doi:10.1007/978-3-031-63498-7_11.

42 Krystof Hoder, Laura Kovács, and Andrei Voronkov. Playing in the grey area of proofs. In John Field and Michael Hicks, editors, *POPL 2012*, pages 259–272. ACM, 2012. doi:10.1145/2103656.2103689.

43 Guoxiang Huang. Constructing Craig interpolation formulas. In Ding-Zhu Du and Ming Li, editors, *COCOON '95*, volume 959 of *LNCS*, pages 181–190. Springer, 1995. doi:10.1007/BFb0030832.

44 Matti Järvisalo, Armin Biere, and Marijn Heule. Blocked clause elimination. In Javier Esparza and Rupak Majumdar, editors, *TACAS 2010*, volume 6015 of *LNCS*, pages 129–144, 2010. doi:10.1007/978-3-642-12002-2_10.

45 Ranjit Jhala and Kenneth L. McMillan. A practical and complete approach to predicate refinement. In Holger Hermanns and Jens Palsberg, editors, *TACAS 2006*, volume 3920 of *LNCS*, pages 459–473. Springer, 2006. doi:10.1007/11691372_33.

46 Cezary Kaliszyk and Josef Urban. FEMaLeCoP: Fairly efficient machine learning connection prover. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *LPAR-20*, volume 9450 of *LNCS (LNAI)*, pages 88–96. Springer, 2015. doi:10.1007/978-3-662-48899-7_7.

47 Benjamin Kiesl, Martin Suda, Martina Seidl, Hans Tompits, and Armin Biere. Blocked clauses in first-order logic. In Thomas Eiter and David Sands, editors, *LPAR-21*, volume 46 of *EPiC*, pages 31–48, 2017. doi:10.29007/C3WQ.

48 Patrick Koopmann, Sebastian Rudolph, Renate A. Schmidt, and Christoph Wernhard, editors. *SOQE 2017*, volume 2013 of *CEUR Workshop Proc.* CEUR-WS.org, 2017. URL: https://ceur-ws.org/Vol-2013.

49 Laura Kovács and Andrei Voronkov. Interpolation and symbol elimination. In Renate A. Schmidt, editor, *CADE-22*, volume 5663 of *LNCS*, pages 199–213. Springer, 2009. doi:10.1007/978-3-642-02959-2_17.

50 Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *CAV 2013*, volume 8044 of *LNCS*, pages 1–35. Springer, 2013. doi:10.1007/978-3-642-39799-8_1.

51 Laura Kovács and Andrei Voronkov. First-order interpolation and interpolating proof systems. In Thomas Eiter and David Sands, editors, *LPAR-21*, volume 46 of *EPiC*, pages 49–64. EasyChair, 2017. doi:10.29007/1qb8.

52 Georg Kreisel and Jean-Louis Krivine. *Elements of mathematical logic. (Model theory)*. North-Holland, Amsterdam, 2. edition, 1971. First edition 1967. Translation of the French *Eléments de logique mathématique, théorie des modeles* published by Dunod, Paris in 1964.

53 Oliver Kullmann. On a generalization of extended resolution. *Discret. Appl. Math.*, 96-97:149–176, 1999. doi:10.1016/S0166-218X(99)00037-2.

54 Richard Char-Tung Lee. *A completeness theorem and computer program for finding theorems derivable from given axioms*. PhD thesis, University of California, Berkeley, CA, 1967.

55 Reinhold Letz. *First-Order Calculi and Proof Procedures for Automated Deduction*. Dissertation, TU München, 1993. https://web.archive.org/web/20230604101128/https://www2.tcs.ifi.lmu.de/~letz/diss.ps, accessed Jun 12, 2026.

56 Reinhold Letz. First-order tableau methods. In Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors, *Handb. of Tableau Methods*, pages 125–196. Kluwer Academic Publishers, 1999. doi:10.1007/978-94-017-1754-0_3.

57 Reinhold Letz. *Tableau and Connection Calculi. Structure, Complexity, Implementation*. Habilitationsschrift, TU München, 1999. https://web.archive.org/web/20230604101128/https://www2.tcs.ifi.lmu.de/~letz/habil.ps, accessed Jun 12, 2026.

58 Reinhold Letz, Johann Schumann, Stefan Bayerl, and Wolfgang Bibel. SETHEO: A high-performance theorem prover. *J. Autom. Reasoning*, 8(2):183–212, 1992. doi:10.1007/BF00244282.

59 Reinhold Letz and Gernot Stenz. Model elimination and connection tableau procedures. In Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 2, chapter 28, pages 2015–2114. Elsevier, 2001. doi:10.1016/B978-044450813-3/50030-8.

60 Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. *ACM Trans. Comp. Log.*, 2(4):526–541, 2001. doi:10.1145/383779.383783.

61 Fangzhen Lin. On strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, 128:143–159, 2001.

62 Fangzhen Lin. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *KR-02*, pages 170–176. Morgan Kaufmann, 2002.

63 Donald W. Loveland. Mechanical theorem proving by model elimination. *JACM*, 15(2):236–251, 1968. doi:10.1145/321450.321456.

64 Larisa L. Maksimova. Craig's theorem in superintuitionistic logics and amalgamable varieties of pseudo-Boolean algebras. *Algebra and Logic*, 16(6):427–455, 1977. doi:10.1007/BF01670006.

65 John McCarthy. Circumscription – a form of non-monotonic reasoning. *AI*, 13:27–39, 1980. doi:10.1016/0004-3702(80)90011-9.

66 William McCune. Un-Skolemizing clause sets. *Information Processing Letters*, 29(5):257–263, 1988.

67 William McCune. OTTER 3.3 Reference Manual. Technical Report ANL/MCS-TM-263, Argonne National Laboratory, 2003. https://www.cs.unm.edu/~mccune/otter/Otter33.pdf, accessed Jun 12, 2026.

68 William McCune. Prover9 and Mace4. http://www.cs.unm.edu/~mccune/prover9, accessed Jun 12, 2026, 2005–2010.

69 Kenneth L. McMillan. Interpolation and SAT-based model checking. In Warren A. Hunt Jr. and Fabio Somenzi, editors, *CAV 2023*, volume 2725 of *LNCS*, pages 1–13. Springer, 2003. doi:10.1007/978-3-540-45069-6_1.

70 Kenneth L. McMillan. An interpolating theorem prover. *Theor. Comput. Sci.*, 345(1):101–121, 2005.

71 Kenneth L. McMillan. Quantified invariant generation using an interpolating saturation prover. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS 2008*, volume 4963 of *LNCS*, pages 413–427. Springer, 2008. doi:10.1007/978-3-540-78800-3_31.

72 Grigori Mints. Cut-free formulations for a quantified logic of here and there. *Ann. Pure Appl. Log.*, 162(3):237–242, 2010. doi:10.1016/J.APAL.2010.09.009.

73 Nobuyoshi Motohashi. Equality and Lyndon's interpolation theorem. *J. Symb. Log.*, 49(1):123–128, 1984. doi:10.2307/2274095.

74 Andreas Nonnengart and Andrzej Szałas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In E. Orlowska, editor, *Logic at Work. Essays Ded. to the Mem. of Helena Rasiowa*, pages 89–108. Springer, 1998.

75 Andreas Nonnengart and Christoph Weidenbach. Computing small clause normal forms. In Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 1, chapter 6, pages 335–367. Elsevier, 2001. doi:10.1016/B978-044450813-3/50008-4.

76 Arnold Oberschelp. On the Craig-Lyndon interpolation theorem. *J. Symb. Log.*, 33(2):271–274, 1968. doi:10.2307/2269873.

77 Hans Jürgen Ohlbach. SCAN – Elimination of predicate quantifiers: System description. In Michael A. McRobbie and John K. Slaney, editors, *CADE-13*, volume 1104 of *LNCS (LNAI)*, pages 161–165. Springer, 1996. doi:10.1007/3-540-61511-3_77.

78 Jens Otten. The nanoCoP 2.0 connection provers for classical, intuitionistic and modal logics. In Anupam Das and Sara Negri, editors, *TABLEAUX 2021*, volume 12842 of *LNCS (LNAI)*, pages 236–249. Springer, 2021. doi:10.1007/978-3-030-86059-2_14.

79 Jens Otten and Wolfgang Bibel. leanCoP: lean connection-based theorem proving. *J. Symb. Comput.*, 36(1-2):139–161, 2003. doi:10.1016/S0747-7171(03)00037-3.

80 Dag Prawitz. An improved proof procedure. *Theoria*, 26:102–139, 1960.

81 Dag Prawitz. Advances and problems in mechanical proof procedures. *Machine Intelligence*, 4:59–71, 1969. Reprinted with author preface in J. Siekmann, G. Wright (eds.): *Automation of Reasoning, vol 2: Classical Papers on Computational Logic 1967–1970, Springer, 1983, pp. 283–297.*

82 Michael Rawson and Giles Reger. Eliminating models during model elimination. In Anupam Das and Sara Negri, editors, *TABLEAUX 2021*, volume 12842 of *LNCS (LNAI)*, pages 250–265. Springer, 2021. doi:10.1007/978-3-030-86059-2_15.

83 Michael Rawson, Christoph Wernhard, Zsolt Zombori, and Wolfgang Bibel. Lemmas: Generation, selection, application. In Revantha Ramanayake and Josef Urban, editors, *TABLEAUX 2023*, LNAI, pages 153–174, 2023. doi:10.1007/978-3-031-43513-3_9.

84 George A. Robinson and Larry Wos. Paramodulation and theorem-proving in first-order theories with equality. In Bernard Meltzer and Donald Michie, editors, *Machine Intelligence IV*, pages 135–150. Edinburgh University Press, 1969.

85 J. Alan Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23–41, 1965. doi:10.1145/321250.321253.

86 J. Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier, 2001.

87 Renate A. Schmidt. The Ackermann approach for modal logic, correspondence theory and second-order reduction. *J. Applied Logic*, 10(1):52–74, 2012. doi:http://dx.doi.org/10.1016/j.jal.2012.01.001.

88 Renate A. Schmidt, Christoph Wernhard, and Yizheng Zhao, editors. *SOQE 2021*, volume 3009 of *CEUR Workshop Proc.* CEUR-WS.org, 2021. URL: https://ceur-ws.org/Vol-3009.

89 Stephan Schulz, Simon Cruanes, and Petar Vukmirović. Faster, higher, stronger: E 2.3. In Pascal Fontaine, editor, *CADE 27*, number 11716 in LNAI, pages 495–507. Springer, 2019. doi:10.1007/978-3-030-29436-6_29.

90 Kurt Schütte. Ein System des verknüpfenden Schliessens. *Arch. math. Logik*, 2:55–67, 1956. doi:10.1007/BF01969991.

91 James R. Slagle. Interpolation theorems for resolution in lower predicate calculus. *JACM*, 14(3):535–542, 1970. doi:10.1145/321592.321604.

92 Raymond M. Smullyan. *First-Order Logic*. Springer, 1968. Also republished with corrections by Dover publications, 1995.

93 Mark E. Stickel. A Prolog Technology Theorem Prover. *New Gener. Comput.*, 2(4):371–383, 1984. doi:10.1007/BF03037328.

94 Geoff Sutcliffe. Stepping Stones in the TPTP World. In Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt, editors, *IJCAR 2024*, number 14739 in LNCS (LNAI), pages 30–50. Springer, 2024. doi:10.1007/978-3-031-63498-7_3.

95 Andrzej Szałas. On the correspondence between modal and classical logic: An automated approach. *J. Logic and Computation*, 3:605–620, 1993. doi:10.1093/LOGCOM/3.6.605.

96 Gaisi Takeuti. *Proof Theory*. North-Holland, second edition, 1987.

97 Arne S. Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, second edition, 2000.

98 Allen Van Gelder and Rodney W. Topor. Safety and translation of relational calculus queries. *ACM Trans. Database Syst.*, 16(2):235–278, 1991. doi:10.1145/114325.103712.

99 Christoph Wernhard. Semantic knowledge partitioning. In José Júlio Alferes and João Leite Leite, editors, *JELIA 04*, volume 3229 of *LNCS (LNAI)*, pages 552–564. Springer, 2004. doi:10.1007/978-3-540-30227-8_46.

100 Christoph Wernhard. Second-order quantifier elimination on relational monadic formulas – A basic method and some less expected applications. In Hans de Nivelle, editor, *TABLEAUX 2015*, volume 9323 of *LNCS (LNAI)*. Springer, 2015. doi:10.1007/978-3-319-24312-2_18.

101 Christoph Wernhard. The PIE system for proving, interpolating and eliminating. In Pascal Fontaine, Stephan Schulz, and Josef Urban, editors, *PAAR 2016*, volume 1635 of *CEUR Workshop Proc.*, pages 125–138. CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1635/paper-11.pdf.

102 Christoph Wernhard. Craig interpolation and access interpolation with clausal first-order tableaux. Technical Report Knowledge Representation and Reasoning 18-01, Technische Universität Dresden, 2018. doi:10.48550/arXiv.1802.04982.

103 Christoph Wernhard. Facets of the PIE environment for proving, interpolating and eliminating on the basis of first-order logic. In Petra Hofstedt et al., editors, *DECLARE 2019, Revised Selected Papers*, volume 12057 of *LNCS (LNAI)*, pages 160–177. Springer, 2020. doi:10.1007/978-3-030-46714-2_11.

104 Christoph Wernhard. Craig interpolation with clausal first-order tableaux. *J. Autom. Reasoning*, 65(5):647–690, 2021. doi:10.1007/s10817-021-09590-3.

105 Christoph Wernhard. Range-restricted and Horn interpolation through clausal tableaux. In Revantha Ramanayake and Josef Urban, editors, *TABLEAUX 2023*, volume 14278 of *LNCS (LNAI)*, pages 3–23. Springer, 2023. doi:10.1007/978-3-031-43513-3_1.

106 Christoph Wernhard. Craig-Lyndon interpolation for the logic of here and there with a variation of Mints' sequent system. In Stefan Hetzl, Jean Christoph Jung, Renate A. Schmidt, and Christoph Wernhard, editors, *CI-BD-SOQE 2026*, CEUR Workshop Proceedings, 2026. Preprint: https://arxiv.org/abs/2601.04080.

Chapter 13

Craig Interpolation in Program Verification

Philipp Rümmer
University of Regensburg, Germany
Uppsala University, Sweden

Abstract

Craig interpolation is used in program verification for automating key tasks such as the inference of loop invariants and the computation of program abstractions. This chapter covers some of the most important techniques that have been developed in this context over the last years, focusing on two aspects: the derivation of Craig interpolants modulo the theories and data types used in verification and the basic design of verification algorithms applying interpolation.

Contents

1 Introduction 462
1.1 Informal Overview: Invariant Inference using Craig Interpolation 463
1.2 Organization of the Chapter 467
2 Satisfiability and Interpolation Modulo Theories 467
2.1 Basic Definitions 467
2.1.1 First-Order Logic and Theories 468
2.1.2 Craig Interpolation Modulo Theories 469
2.1.3 Extended Forms of Craig Interpolation 471
2.2 Interpolation Modulo Linear Arithmetic 473
2.2.1 Linear Real Arithmetic (LRA) 474
2.2.2 Beyond Conjunctions of Non-strict Inequalities 477
2.2.3 Linear Integer Arithmetic (LIA) 479
2.3 Further Interpolation Approaches 483
2.3.1 Proof-Based Interpolation 483
2.3.2 Graph-Based Interpolation 484
2.3.3 Interpolation by Reduction 484
3 Verification Algorithms Based on Interpolation 485
3.1 BMC: Bounded Model Checking 485
3.2 IMC: Interpolation and SAT-Based Model Checking 486



© Philipp Rümmer;
licensed under Creative Commons License CC BY 4.0

In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 13, pages 461–501. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.m

3.2.1 The IMC Algorithm . . . 486
3.2.2 Related Interpolation-Based Algorithms . . . 488
3.3 CEGAR: Counterexample-Guided Abstraction Refinement . . . 489
3.3.1 Existential Abstractions . . . 489
3.3.2 Predicate Abstraction . . . 490
3.3.3 Counterexample-Guided Abstraction Refinement . . . 490
3.3.4 Variants of Predicate Abstraction and CEGAR . . . 491
3.4 CHC: Craig Interpolation and Constrained Horn Clauses . . . 492
3.4.1 Definitions and Basic Properties . . . 492
3.4.2 Encoding of Recursive Programs using CHCs . . . 494
4 Conclusions . . . 495
Acknowledgments . . . 496
References . . . 496

1 Introduction

In program verification, the question is considered how it can be shown (effectively or even automatically) whether a given software program is consistent with its specification. The study of program verification is as old as computer science and has over the years led to a plethora of different techniques, studying the question from various angles and developing approaches that have found widespread application in both academia and industry.

Craig interpolation found its way to the field of verification when it was discovered that interpolation can be used to infer *invariants* [49], resulting in a new paradigm for automating one of the key tasks in verification. The discovery gave rise to a number of new verification algorithms, initially focusing on the verification of *hardware circuits,* for which mainly interpolation in propositional logic was used, and later extending the methods to the analysis of *software programs* and other classes of systems. The newly discovered use of Craig interpolation initiated the development of more effective interpolation algorithms and tools for many different logics, including propositional logic, first-order logic, and the theories relevant in verification, and was an inspiring factor for the application of Craig interpolation also in fields other than verification. Many automated verification tools today apply Craig interpolation for tasks such as reasoning about control flow or data, inferring invariants and other program annotations, or automatically computing program abstractions.

This chapter focuses on two main aspects related to Craig interpolation in verification: techniques for computing Craig interpolants modulo some of the theories used in program verification and the basic design of verification algorithms applying interpolation. Given the huge amount of research in the field, the chapter does not attempt to provide a complete survey of all results, but rather introduces some of the main approaches in detail and with examples and provides a starting point for exploring the world of interpolation-based verification.

1.1 Informal Overview: Invariant Inference using Craig Interpolation

We first illustrate the use of Craig interpolation for inferring loop invariants using an example. The Fibonacci sequence [25] is defined by the equations

$$F_0 = 0, \quad F_1 = 1, \quad F_{i+2} = F_i + F_{i+1}$$

and is computed by the function shown in Fig. 1a. After the ith iteration of the loop, the program variable `a` stores the Fibonacci number F_i and the variable `b` the number F_{i+1}. The function has the precondition `n >= 0`, stated as a `requires` clause, and the postcondition `\result >= 0` expressing that the result F_n returned by the function is not negative ("`ensures`").

The aim of formal verification is to construct a rigorous mathematical proof that a program is consistent with its specification. In order to verify the *partial correctness* [10] of `fib`, it has to be proven that the function, whenever it terminates for a non-negative argument n with some result a, ensures the postcondition $a \geq 0$. Partial correctness proofs are commonly carried out by an induction over the number of loop iterations executed by the program. To streamline this reasoning process, the notion of *inductive loop invariants* (or just *loop invariants*) has been introduced [10], which capture properties that cannot be violated during program execution. More precisely, an inductive loop invariant for a loop is a formula I over the program variables that are in scope in the loop body, satisfying:

- *Initiation:* whenever the loop is entered, the invariant I holds.
- *Consecution:* whenever the loop body is executed starting in a state in which the invariant I and the loop condition hold, the invariant I also holds after executing the loop body.
- *Sufficiency:* whenever the loop terminates in a state satisfying the loop invariant I, the rest of the program executes without errors and the postcondition is established.

For our example program, the three conditions are captured formally on the right-hand side of Fig. 1 using the program instructions `assert` and `assume` [21]. The semantics of `assert(`ϕ`)` is to evaluate a condition ϕ in the current program state and raise an error if ϕ is violated; if ϕ holds, `assert(`ϕ`)` has no effect and program execution continues. The semantics of `assume(`ϕ`)` is similar, but instead of raising an error in case ϕ is violated, the statement will instead suspend the program execution, preventing the rest of the program from being executed. A statement `assume(false)`, in particular, stops further program execution and has similar effect as a non-terminating loop. To deal with `assert` and `assume`, we generalize our notion of partial correctness and say that a program is *partially correct* if no `assert` statement in the program can fail, regardless of the program inputs and the initial state. In the example, the sufficient inductive invariants are exactly the formulas I for which the three programs in Figs. 1b to 1d are partially correct.

We can observe that all three conditions are satisfied by the formula $I = \mathtt{a} \geq 0 \wedge \mathtt{b} \geq 0$. This formula I is therefore a loop invariant and represents a witness for the partial correctness of `fib`. The simpler formula $\mathtt{a} \geq 0$, although it is a property that holds throughout any

```
1  /**
2   * requires n >= 0;
3   * ensures  \result >= 0;
4   */
5  int fib(int n) {
6    int a = 0;
7    int b = 1;
8    int t;
9
10   for (int i = 0; i < n; ++i) {
11     t = b;
12     b = a + b;
13     a = t;
14   }
15
16   return a;
17 }
```

(a) Program computing the nth element of the Fibonacci sequence.

```
1  assume(n >= 0); // requires
2  int a = 0; int b = 1; int t;
3  int i = 0;
4  assert(I);
```

(b) Initiation

```
1  assume(I);
2  assume(i < n);
3  t = b;
4  b = a + b;
5  a = t;
6  ++i;
7  assert(I);
```

(c) Consecution

```
1  assume(I);
2  assume(!(i < n));
3  assert(a >= 0); // ensures
```

(d) Sufficiency

Figure 1 Fibonacci program and the conditions for inductive loop invariants.

execution of the program as well, does not satisfy the *consecution* condition and is therefore not a correct inductive invariant. More precisely, when substituting $\mathtt{a} \geq 0$ for I in Fig. 1c, we obtain a program that is not partially correct, an assertion failure occurs when the program is run starting in the state $\mathtt{a} = 0, \mathtt{b} = -1, \mathtt{i} = 0, \mathtt{n} = 1$.

Coming up with inductive loop invariants, like the formula $\mathtt{a} \geq 0 \wedge \mathtt{b} \geq 0$, is a known challenge in verification, for the same reasons as it is not easy to find induction formulas in proofs. Craig interpolation is a useful and widely applied technique for finding invariant candidates. To identify the connection between Craig interpolation and loop invariants, consider the unwinding of the Fibonacci program in Fig. 2a, which is obtained by listing the statements on the execution path that executes the loop body twice. The assumptions added to the program capture the status of the loop condition, which is *true* for two loop iterations and *false* after that, and ensure that the executions of the program in Fig. 2a correspond exactly to the executions of the original program in Fig. 1a with two loop iterations. Unwinding loops to obtain loop-free programs is a common technique in verification.

Consider then possible *intermediate assertions* K_0, K_1, K_2 that enable us to decompose the program in Fig. 2a into multiple parts at the marked locations. We say that a formula A is an intermediate assertion for a partially correct program $P;Q$ if the programs $P;\mathtt{assert}(A)$

```
1  assume(n >= 0);
2  int a = 0;
3  int b = 1;
4  int t;
5  int i = 0;
6
7  // Intermediate assertion K0
8
9  assume(i < n); // 1st iteration
10 t = b; b = a + b; a = t; ++i;
11
12 // Intermediate assertion K1
13
14 assume(i < n); // 2nd iteration
15 t = b; b = a + b; a = t; ++i;
16
17 // Intermediate assertion K2
18
19 assume(!(i < n)); // Loop exit
20 assert(a >= 0);
```

(a) Unwinding of the loop.

```
1  assume(n >= 0); // requires
2  int a = 0; int b = 1; int t;
3  int i = 0;
4  assert(K0);
```

(b) Sub-program 1

```
1  assume(K0);
2  assume(i < n);
3  t = b; b = a + b; a = t; ++i;
4  assert(K1);
```

(c) Sub-program 2

```
1  assume(K1);
2  assume(i < n);
3  t = b; b = a + b; a = t; ++i;
4  assert(K2);
```

(d) Sub-program 3

```
1  assume(K2);
2  assume(!(i < n));
3  assert(a >= 0); // ensures
```

(e) Sub-program 4

Figure 2 Unwinding two iterations of the loop in the Fibonacci program.

and `assume(`A`);`Q are both partially correct as well; in other words, if A is a postcondition that is ensured by the first sub-program P, but that is also precondition ensuring error-free execution of the second sub-program Q. The three intermediate assertions in Fig. 2a similarly split the program into four sub-programs, shown on the right-hand side of Fig. 2, each of which needs to be partially correct for the intermediate assertions to be suitable.

The use of Craig interpolants for finding inductive loop invariants is based on two key observations:

(i) intermediate assertions are a relaxation of inductive loop invariants, and

(ii) intermediate assertions are a case of Craig interpolants.

To see point (i), we can compare the conditions in Fig. 1 with the conditions given in Fig. 2. Conditions 1b and 2b correspond to each other, as do conditions 1d and 2e, whereas the consecution condition 1c is only approximated by conditions 2c and 2d. Therefore, any formula I satisfying the conditions in Fig. 1, when substituted for K_0, K_1, K_2, will also satisfy the conditions in Fig. 2 and give rise to a correct decomposition of the program.

```
1  assume(n >= 0);
2  int a = 0, b = 1;
3  int t, i = 0;
4  assume(i < n);
5  t = b; b = a + b; a = t; ++i;
6
7  // Intermediate assertion K
8
9  assume(i < n);
10 t = b; b = a + b; a = t; ++i;
11 assume(!(i < n));
12 assert(a >= 0);
```

(a) Computation of a single intermediate assertion.

$$
\left.\begin{array}{l} n_0 \geq 0 \wedge \\ a_0 = 0 \wedge b_0 = 1 \wedge i_0 = 0 \wedge \\ i_0 < n_0 \wedge \\ t_1 = b_0 \wedge b_1 = a_0 + b_0 \wedge a_1 = t_1 \wedge \\ i_1 = i_0 + 1 \end{array}\right\} A
$$

$$\wedge$$

$$
\left.\begin{array}{l} i_1 < n_0 \wedge \\ t_2 = b_1 \wedge b_2 = a_1 + b_1 \wedge a_2 = t_2 \wedge \\ i_2 = i_1 + 1 \wedge \\ i_2 \not< n_0 \\ \rightarrow a_2 \geq 0 \end{array}\right\} C
$$

(b) Verification condition for partial correctness.

Figure 3 Intermediate assertions.

Vice versa, not all possible intermediate assertions K_0, K_1, K_2 are correct loop invariants, since intermediate assertions will in general not satisfy consecution 1c. Given a method to explore possible intermediate assertions K_0, K_1, K_2, however, we can generate *candidates* for inductive loop invariants. We survey some of the algorithms that have been proposed for generating and refining invariant candidates in this way in Section 3.

To see point (ii), we focus on the case of just a single intermediate assertion to be computed for the unwound Fibonacci program, shown in Fig. 3a. The partial correctness of the program can be checked by translating the program, including the `assert` and `assume` statements, to a *verification condition* [10], which is in this case a formula over linear integer arithmetic. The verification condition is shown in Fig. 3b and is constructed in such a way that the condition is *valid* if and only if the program is *partially correct.* For this, the condition lists all assumptions made in the program, includes the assignments to variables in the form of equations, and states the assertion as the last inequality $a_2 \geq 0$. Since the program variables can change their values during program execution, a program variable can correspond to multiple symbols in the verification condition: for instance, program variable `a` turns into the logical constants a_0, a_1, a_2.

By comparing the verification condition in Fig. 3b with our definition of intermediate assertions, we can see that the correct intermediate assertions K correspond to the formulas ϕ for which the implications $A \rightarrow \phi$ and $\phi \rightarrow C$ are valid, with A, C as marked in Fig. 3b. Indeed, K has to be a formula that is entailed by the program statements prior to the respective point in the program, which is expressed by the implication $A \rightarrow \phi$, and it has to be strong enough to ensure that the rest of the program executes correctly, hence $\phi \rightarrow C$. This already indicates that the intermediate assertions correspond to the *Craig interpolants*

for the implication $A \to C$, as interpolants have to satisfy the same implication conditions.

To bridge the gap between intermediate assertions and Craig interpolants further, we have to consider which symbols are allowed to occur in the intermediate assertion K. The formula K is supposed to be an expression over the program variables `n, a, b, t, i`. In the verification condition, those variables have been replaced with different indexed versions; at the point where K is supposed to be evaluated (line 7 of Fig. 3a), the most recent versions are named n_0, a_1, b_1, t_1, i_1. Naturally, those are exactly the symbols that occur in both A and C, as all the older previous versions of the variables will be present only in A, while future versions can only occur in C. Craig interpolants for the implication $A \to C$ therefore range over exactly the right set n_0, a_1, b_1, t_1, i_1 of symbols. To derive intermediate assertions for our program, we can first compute a Craig interpolant ϕ for $A \to C$, and then rename the symbols to obtain the intermediate assertion $K = \phi[n_0 \mapsto \mathtt{n}, a_1 \mapsto \mathtt{a}, b_1 \mapsto \mathtt{b}, t_1 \mapsto \mathtt{t}, i_1 \mapsto \mathtt{i}]$.

We will return to the Fibonacci example later in this chapter, when it is discussed how we can compute Craig interpolants modulo the theories relevant for the program. One possible interpolant we will then derive is the formula $\phi = (b_1 \geq 1)$, which can be translated to the intermediate assertion $K = (\mathtt{b} \geq 1)$.

1.2 Organization of the Chapter

The rest of the chapter consists of two main parts. In Section 2, we introduce the required background in logic and formally define several notions of Craig interpolation. We then introduce some of the techniques that have been developed in the verification field for computing Craig interpolants, focusing on interpolants for arithmetic formulas that are essential for verification. In Section 3, the second part, we put the interpolation procedures to use by exploring several interpolation-based verification algorithms.

2 Satisfiability and Interpolation Modulo Theories

2.1 Basic Definitions

Automatic approaches in verification often work in a setting of *quantifier-free theories* in *first-order logic,* where the theories are chosen to reflect the data types found in programming languages, while the restriction to quantifier-free formulas is motivated by the desire to stay in a decidable language. For reasoning about quantifier-free formulas modulo theories, the *Satisfiability Modulo Theories* (SMT) paradigm [5] has been developed, in which SAT solvers are combined with theory solvers to obtain efficient decision procedures. Similarly, much of the research on Craig interpolation in verification has focused on methods to compute quantifier-free Craig interpolants modulo theories.

2.1.1 First-Order Logic and Theories

We assume a setting of multi-sorted (classical) first-order logic, see for instance [32]. Slightly extending the notation introduced in Chapter 2, a signature Σ is now a collection of relation symbols, constant and function symbols, as well as sorts, where each predicate R is endowed with a uniquely defined rank $\alpha(R) = (\sigma_1, \ldots, \sigma_n)$ of argument sorts, and each constant and function symbol f with a rank $\alpha(f) = (\sigma_1, \ldots, \sigma_n, \sigma_0)$ consisting of the argument sorts $\sigma_1, \ldots, \sigma_n$ and the result sort σ_0. Formulas ϕ and terms t are defined by the following grammar:

$$\phi \ ::= \ R(t, \ldots, t) \mid t = t \mid \top \mid \phi \wedge \phi \mid \neg\phi \mid \exists x : \sigma.\, \phi \ , \qquad\qquad t \ ::= \ x \mid f(t, \ldots, t) \ .$$

In the grammar, R ranges over relation symbols, f over function symbols, x over variables, and σ over sorts. Constants are identified with nullary functions. We generally assume that terms and formulas are well-typed, which means that relation and function symbols are only applied to terms of the right sort, and that $=$ is only used to compare terms of the same sort. We also use the short-hand notations $\bot$, $\phi \vee \phi$, $\phi \rightarrow \phi$, and $\forall x : \sigma.\, \phi$, all with the usual definition. Given a formula ϕ or term t, we denote the sets of occurring relation and function symbols by $\mathrm{sig}(\phi) \subseteq \Sigma$ and $\mathrm{sig}(t) \subseteq \Sigma$, respectively.

A *closed formula* or *sentence* is a formula in which all variables are bound. A *ground formula* is a formula that does not contain any variables (and, therefore, no quantifiers). Note that ground formulas can still contain constant symbols. A *quantifier-free* formula is a formula that does not contain quantifiers, but that might contain variables.

The semantics of formulas is defined, as is usual, in terms of Σ-*structures* S that map each sort σ in Σ to some non-empty domain D_σ, each relation symbol in Σ to a set-theoretical relation over the argument domains, and each function symbol in Σ to a set-theoretical function from the argument domains to the result domain. To define the satisfiability of a formula modulo theories, we need the notion of *extensions* of signatures and structures. We say that a signature Ω is an *extension* of Σ, denoted $\Sigma \subseteq \Omega$, if Ω contains more sorts, relation symbols, or function functions than Σ, without changing the rank $\alpha(R)$ or $\alpha(f)$ of symbols that already exist in Σ. If $\Sigma \subseteq \Omega$, then a Σ-structure S_Σ is *extended* by an Ω-structure S_Ω if S_Ω interprets the sorts and symbols already present in Σ in the same way as S_Σ, while adding interpretations for the new sorts and symbols in Ω.

▶ **Definition 1** (Theory). *A* theory T *is defined by a signature* Σ_T *and a set* $\mathcal{S}_T$ *of structures over* Σ_T*. A sentence* ϕ *over the signature* Σ_T *is called* T-satisfiable (or T-sat) *if it evaluates to true for some structure in* $\mathcal{S}_T$*. A sentence* ϕ *over an extension* $\Omega \supseteq \Sigma_T$ *is called* T-satisfiable *if it evaluates to true in some extension of a structure in* $\mathcal{S}_T$ *to* Ω*.*

For sentences in some signature Ω that is larger than Σ_T, this means that additional sorts and relation or function symbols can be interpreted arbitrarily. As a special case, we say that a sentence ϕ is simply *satisfiable* if it is $T_\emptyset$-satisfiable in the empty theory $T_\emptyset$. Sentences that are not T-satisfiable are called T*-unsatisfiable (or* T*-unsat),* and sentences whose negation is

T-unsatisfiable are called T-*valid*. We also speak of a set M of sentences being T-satisfiable if there is a common structure satisfying all sentences in M.

We write $\Gamma \models_T \psi$ if the set Γ of sentences entails the sentence ψ modulo T: whenever an extension of a structure in $\mathcal{S}_T$ satisfies all sentences in Γ, it also satisfies ψ. As a special case, we write $\phi \models_T \psi$ if sentence ϕ entails sentence ψ modulo T, which is equivalent to $\{\phi\} \models_T \psi$ and to the conjunction $\phi \wedge \neg\psi$ being T-unsatisfiable. We write $\phi \equiv_T \psi$ for two sentences being T-equivalent, defined as $\phi \models_T \psi$ and $\psi \models_T \phi$, and leave out the T when it is clear from the context.

2.1.2 Craig Interpolation Modulo Theories

Since algorithms in verification (just like in automated reasoning) are usually formulated in terms of *satisfiable* or *unsatisfiable* formulas, instead of *valid* or *counter-satisfiable* formulas, it is common in the area to also define Craig interpolation in the "reversed" way. The relationship between standard Craig interpolants and *separators* has already been discussed in Chapter 1, clarifying that the difference between the two notions is mostly superficial, at least in classical logic. In order to avoid confusion with the definition in the literature, we mostly stick to this reversed version of interpolation throughout the chapter, but also introduce the standard version for completeness reasons.

▶ **Definition 2** (Craig interpolant modulo theory). *Suppose that T is a theory and $A \to C$ a sentence. A sentence I is called a* Craig interpolant modulo T *for $A \to C$ if the following conditions hold:*

- $A \models_T I$ *and* $I \models_T C$, *and*
- $\mathrm{sig}(I) \subseteq (\mathrm{sig}(A) \cap \mathrm{sig}(C)) \cup \Sigma_T$.

Compared to the definition of Craig interpolants in first-order logic from Chapter 2, we now consider entailment $\models_T$ modulo the theory T, and we allow the interpolant I to include arbitrary *interpreted* symbols from the theory signature Σ_T. As a further, more technical difference, we only consider closed formulas (sentences) in our definition; this restriction could be lifted easily but is usually unproblematic, since free variables can be replaced with constant symbols. The definition in Chapter 2 otherwise corresponds to our definition for the special case of the empty theory $T_\emptyset$.

▶ **Example 3.** Consider the implication $A \to C$, with A, C defined as in Fig. 3b over the theory of linear integer arithmetic (LIA). Correct Craig interpolants modulo LIA include $I_1 = b_1 \geq 1$ and $I_2 = a_1 \geq 0 \wedge b_1 \geq 0$. ⌟

As discussed in Chapter 1, an interpolation problem $A \to C$ can equivalently be expressed as a conjunction, leading to the alternative reverse notion of interpolation:

▶ **Definition 4** (Separator, reverse Craig interpolant). *Suppose that T is a theory and $A \wedge B$ a sentence. A sentence I is called a* T-separator *or* reverse Craig interpolant modulo T *for $A \wedge B$ if the following conditions hold:*

- $A \models_T I$ *and* $I \wedge B \models_T \bot$, *and*
- $\mathrm{sig}(I) \subseteq (\mathrm{sig}(A) \cap \mathrm{sig}(B)) \cup \Sigma_T$.

▶ **Theorem 5** (Existence of T-separators [44]). *Suppose that T is a theory and $A \wedge B$ a sentence that is T-unsatisfiable. Then there is a T-separator for $A \wedge B$; more precisely, there are T-separators I_1, I_2 such that:*

- $A \models_T I_1$ *and* $I_1 \wedge B \models \bot$,
- $A \models I_2$ *and* $I_2 \wedge B \models_T \bot$,
- $\mathrm{sig}(I_1) \subseteq (\mathrm{sig}(A) \cup \Sigma_T) \cap \mathrm{sig}(B)$, *and*
- $\mathrm{sig}(I_2) \subseteq \mathrm{sig}(A) \cap (\mathrm{sig}(B) \cup \Sigma_T)$.

The theorem provides a stronger result about the existence of separators than required by Definition 4. The sentences I_1, I_2 in the theorem are both T-separators according to Definition 4, but we are even at liberty to assume that one of the two entailment conditions is *general* entailment $\models$, and only the other entailment is modulo the theory T. Similarly, the theorem provides stronger guarantees about the signature used in I_1, I_2, as the formulas only have to refer to interpreted symbols from Σ_T that also occur in one of the two formulas A, B.

Although it provides a general guarantee for the existence of reverse interpolants, Theorem 5 is usually not strong enough for applications in verification, as the theorem does not make any statement about the occurrence of *quantifiers* in I_1 and I_2. In general, even if the formulas A, B are quantifier-free, there might only be T-separators containing quantifiers, which is usually less desirable for applications.

▶ **Example 6.** McCarthy proposed a theory of functional arrays that is today widely used in program verification [48]. The theory provides functions $\mathit{select}(M, a)$ for reading from an array M at index a, and $\mathit{store}(M, a, x)$ for updating an array M at index a to the new value x. In the theory of arrays, the conjunction

$$\underbrace{M' = \mathit{store}(M, a, x)}_{A} \wedge \underbrace{b \neq c \wedge \mathit{select}(M, b) \neq \mathit{select}(M', b) \wedge \mathit{select}(M, c) \neq \mathit{select}(M', c)}_{B}$$

is unsatisfiable, but all reverse interpolants contain quantifiers [51]. Indeed, interpolants can only refer to the symbols M', M, which occur in both A and B, as well as the array functions *select* and *store*. A quantified interpolant is $\exists a, x.\, M' = \mathit{store}(M, a, x)$. ⌟

▶ **Definition 7** (General quantifier-free interpolation [14]). *A theory T admits* general quantifier-free interpolation *if for every T-unsatisfiable conjunction $A \wedge B$ there is a quantifier-free T-separator.*

As the example illustrates, the standard theory of arrays does not admit general quantifier-free interpolation, while many other theories (for instance, Linear Real Arithmetic, LRA, Section 2.2.1) have this property. It is possible to extend the theory of arrays, however, to obtain quantifier-free interpolation [13, 62, 36] (also see Section 2.3.1).

It is often useful to consider a somewhat weaker criterion for quantifier-free interpolation:

▶ **Definition 8** (Plain quantifier-free interpolation [14]). *A theory T admits* plain quantifier-free interpolation *if for every T-unsatisfiable conjunction $A \wedge B$ that only contains symbols from Σ_T and constant symbols there is a quantifier-free T-separator.*

The difference between "general" and "plain" is that the former demands quantifier-free interpolants even for conjunctions $A \wedge B$ that contain relation or function symbols that are not part of T, whereas the latter is happy with only additional constant symbols.

▶ **Example 9.** The theory of linear integer arithmetic (LIA) admits plain quantifier-free interpolants, provided that the theory includes divisibility statements $k \mid \cdot$ for every $k \in \mathbb{N}$; a formula $k \mid t$ is read "k divides t" and asserts that t is a multiple of k. Plain quantifier-free interpolation follows thanks to the observation that weakest and strongest interpolants in LIA can be computed using quantifier elimination [32], in exactly the same way as was shown for propositional logic in Chapter 1. Whether LIA admits *general quantifier-free interpolation* depends on the exact signature of the theory that is assumed. To see this, consider the unsatisfiable conjunction

$$\underbrace{x = 2 \cdot y \wedge p(y)}_{A} \wedge \underbrace{x = 2 \cdot z \wedge \neg p(z)}_{B}$$

where p is a relation symbol that is not in the signature of LIA. A separator for the conjunction $A \wedge B$ is the formula $I = p(x \div 2)$, where $\div$ denotes division with remainder. If the function $\div$ is not included in the signature of LIA, no quantifier-free separators exist [11], and the theory loses the general quantifier-free interpolation property. ⌟

2.1.3 Extended Forms of Craig Interpolation

The notion of separators (or reverse Craig interpolants) defined in the previous section is sometimes called *binary interpolation,* as only two formulas A, B are involved. It is possible to define corresponding n-ary versions, in which an arbitrary number of formulas can be separated, which is often advantageous in verification. The two main extended notions of interpolation that are used today, and that are commonly supported by SMT solvers, are *sequence* and *tree* interpolation.

▶ **Definition 10** (Sequence interpolant [34, 52]). *Suppose T is a theory and $A_1 \wedge \cdots \wedge A_n$ a sentence. A* sequence interpolant modulo T *or* inductive sequence of interpolants modulo T *for $A_1 \wedge \cdots \wedge A_n$ is a sequence $I_0, I_1, \ldots, I_n$ of sentences such that:*

- $I_0 = \top, I_n = \bot$,
- *for all $i \in \{1, \ldots, n\}$: $I_{i-1} \wedge A_i \models_T I_i$, and*
- *for all $i \in \{0, \ldots, n\}$:* $\mathrm{sig}(I_i) \subseteq (\mathrm{sig}(A_1 \wedge \cdots \wedge A_i) \cap \mathrm{sig}(A_{i+1} \wedge \cdots \wedge A_n)) \cup \Sigma_T$.

It is easy to see that a separator I for $A_1 \wedge A_2$ corresponds to the sequence interpolant $\top, I, \bot$, so that sequence interpolation strictly extends binary interpolation. It should

be noted, as well, that the individual formulas of a sequence interpolant depend on each other, as each formula I_{i-1} has to be strong enough to imply, together with A_i, the next formula I_i. Sequence interpolants can be derived by repeatedly computing binary interpolants: we set $I_0 = \top$ and then compute each formula I_i, for $i = \{1, \ldots, n\}$, by computing a separator for the conjunction $(I_{i-1} \wedge A_i) \wedge (A_{i+1} \wedge \cdots \wedge A_n)$. For reasons of efficiency, this iterative computation of sequence interpolants is rarely used, however, SMT solvers instead implement interpolation procedures that can compute the whole sequence interpolant in one go.

▶ **Example 11.** As we observed in Section 1.1 and Fig. 3b, a single intermediate assertion for a program can be derived using a binary interpolant by translating the statements on a program path to an implication $A \to C$. We can use sequence interpolants to derive multiple consecutive assertions in a program. For this, consider again the version of the Fibonacci example shown in Fig. 2, in which the loop was unwound two times with the goal of finding intermediate assertions K_0, K_1, K_2. The locations of the intermediate assertions in the program indicate that we should now group together the program statements to obtain four conjuncts; the assertion `a >= 0` is negated to state that we are interested in runs violating the assertion:

$$\underbrace{\begin{array}{l} n_0 \geq 0 \wedge a_0 = 0 \wedge \\ b_0 = 1 \wedge i_0 = 0 \end{array}}_{A_1} \wedge \underbrace{\begin{array}{l} i_0 < n_0 \wedge t_1 = b_0 \wedge \\ b_1 = a_0 + b_0 \wedge \\ a_1 = t_1 \wedge i_1 = i_0 + 1 \end{array}}_{A_2} \wedge \underbrace{\begin{array}{l} i_1 < n_0 \wedge t_2 = b_1 \wedge \\ b_2 = a_1 + b_1 \wedge \\ a_2 = t_2 \wedge i_2 = i_1 + 1 \end{array}}_{A_3} \wedge \underbrace{i_2 \not< n_0 \wedge a_2 \not\geq 0}_{A_4}$$

The conjunction $A_1 \wedge A_2 \wedge A_3 \wedge A_4$ is unsatisfiable, and a sequence interpolant is given by the formulas

$$\top,\ a_0 + b_0 \geq 1,\ b_1 \geq 1,\ a_2 \geq 1,\ \bot\ .$$

The formulas in the sequence interpolant can be translated to intermediate program assertions: $K_0 =$ `a + b >= 1`, $K_1 =$ `b >= 1`, $K_2 =$ `a >= 1`. ⌟

Tree interpolants generalize inductive sequences of interpolants, and are designed with inter-procedural program verification in mind. In this context, the tree structure of the interpolation problem corresponds to the call graph of a program. Tree interpolants make it possible to derive not only intermediate assertions in programs, but also annotations that relate multiple program states, for instance postconditions [33, 30] (also see Section 3.4.2).

▶ **Definition 12** (Tree interpolant [33, 59, 31]). *Suppose T is a theory and $G = (V, E)$ a finite rooted tree, writing $E(v, w)$ to express that the node w is a direct child of v. Further, suppose that each node $n \in V$ of the tree is labeled with a sentence $A(n)$. A function I that labels every node $n \in V$ with a sentence $I(n)$ is called a* tree interpolant modulo T *(for G and A) if the following properties hold:*

(i) *for the root node $v_0 \in V$, it is the case that $I(v_0) = \bot$,*

(ii) *for every node $v \in V$, the following entailment holds:*

$$\bigwedge_{\substack{w \in V \\ (v,w) \in E}} I(w) \wedge A(v) \;\models_T\; I(v) \;,$$

(iii) *for every node $v \in V$, it is the case that:*

$$\mathrm{sig}(I(v)) \;\subseteq\; \Big(\bigcup_{\substack{w \in V \\ E^*(v,w)}} \mathrm{sig}(A(w)) \;\cap\; \bigcup_{\substack{w \in V \\ \neg E^*(v,w)}} \mathrm{sig}(A(w)) \Big) \cup \Sigma_T$$

where E^* is the reflexive transitive closure of E.

Like in the case of sequence interpolants, also tree interpolants can be derived by repeatedly computing binary interpolants [59], although SMT solvers usually provide native support for tree interpolants as well.

Tree interpolation generalizes all of the more specialized forms of interpolation we have encountered so far. To see that tree interpolation subsumes sequence interpolation, note that an n-ary conjunction $A_1 \wedge \cdots \wedge A_n$ can be translated to a tree $G_1 = (\{1, \ldots, n\}, \{(i, i-1) \mid 2 \leq i \leq n\})$ in which each node i is the child of node $i+1$ and is labeled with the formula A_i. The root of the tree is the node n. Similarly, we can reduce *parallel interpolants* [43, 54], as introduced in Chapter 1, to tree interpolants by arranging an implication $\varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow \psi$ as a tree $G_2 = (\{0, 1, \ldots, n\}, \{(0, i) \mid 1 \leq i \leq n\})$ in which the root 0 is labeled with $\neg\psi$ and each child i with the formula φ_i. Parallel interpolants are related to the notion of *symmetric interpolants* [39], which can be similarly reduced to tree interpolation.

Tree interpolants can be generalized further to *disjunctive interpolants* [58], which have, however, not found wider adoption in verification. It has also been observed that the different variants of interpolation correspond to solving particular recursion-free fragments of constrained Horn clauses [59], which is a connection that we return to in Section 3.4.

2.2 Interpolation Modulo Linear Arithmetic

We now survey some of the techniques that have been developed for constructing quantifier-free interpolants modulo a theory T, focusing on the case of *linear arithmetic* that is ubiquitous in verification. For arithmetic formulas, interpolants are commonly *extracted from proofs* in some suitable proof system, resembling the approaches that were discussed in earlier chapters. For verification, extracting interpolants from proofs is attractive, since proofs tend to focus on the relevant information in a verification condition. When computing intermediate assertions (as discussed in Section 1.1) using proof-based interpolation, only the program variables will be mentioned that were important for the SMT solver to prove the correctness of the program. Interpolants from proofs are therefore a useful tool when computing program abstractions.

2.2.1 Linear Real Arithmetic (LRA)

One of the most common theories supported by interpolating SMT solvers is *linear arithmetic* over real numbers or the integers [61, 20]. In the theory of *Linear Real Arithmetic,* LRA, we consider formulas that contain linear inequalities $0 \leq \alpha_1 x_1 + \cdots + \alpha_k x_k + \alpha_0$, where $\alpha_0, \alpha_1, \ldots, \alpha_k \in \mathbb{Q}$ are rational numbers and $x_1, \ldots, x_k$ are constant symbols of sort *Real*. The semantics of scaling with some rational number αx, addition $+$, and the relation symbol $\leq$ is as expected. We initially only consider formulas that are conjunctions of positive inequalities $0 \leq \alpha_1 x_1 + \cdots + \alpha_k x_k + \alpha_0$. Equations $s = t$ between real-valued terms can be translated to a conjunction $0 \leq s - t \wedge 0 \leq t - s$ of two inequalities, while the handling of strict inequalities $0 < s$, disjunctions, and negated equations is discussed in Section 2.2.2.

To interpolate linear inequalities, we rely on a classical result in linear programming:

▶ **Lemma 13** (Farkas' lemma [61]). *Let $A \in \mathbb{R}^{m \times n}$ be a real-valued matrix and $b \in \mathbb{R}^m$ be a column vector. There is a column vector $x \in \mathbb{R}^n$ such that $Ax \leq b$ if and only if for every row vector $y \in \mathbb{R}^m$ with $y \geq 0$ and $yA = 0$ it is the case that $yb \geq 0$.*

The lemma implies a simple uniform representation of witnesses for the unsatisfiability of linear inequalities over the reals: whenever a conjunction $\bigwedge_i 0 \leq t_i$ of linear inequalities is unsatisfiable, we can form a non-negative linear combination $0 \leq \sum_i \alpha_i t_i$ of the inequalities in which the right-hand side $\sum_i \alpha_i t_i$ simplifies to a constant negative value $\beta < 0$. If the matrix A and the vector b in the lemma only contain rational numbers, then also the vector y can be chosen to be rational.

▶ **Lemma 14** (Interpolants for linear inequalities [55, 51]). *Suppose that*

$$\underbrace{\bigwedge_i 0 \leq t_i}_{A} \;\wedge\; \underbrace{\bigwedge_j 0 \leq s_j}_{B}$$

is a conjunction of linear inequalities, and that the linear combination $\sum_i \alpha_i t_i + \sum_j \beta_j s_j$, with $\alpha_i, \beta_j \geq 0$, is constant and negative. Then $I = \big(0 \leq \sum_i \alpha_i t_i\big)$, after simplification and removing terms with coefficient 0, is an LRA-separator for $A \wedge B$.

▶ **Example 15.** We first illustrate this result using an example. Consider the conjunction

$$\underbrace{0 \leq y - 1 \wedge 0 \leq z - x - 2y - 2}_{A} \;\wedge\; \underbrace{0 \leq x \wedge 0 \leq -z + 2}_{B} \;.$$

This sentence is unsatisfiable, since we can form a non-negative contradictory linear combination of the inequalities:

$$0 \leq 2 \cdot (y - 1) + (z - x - 2y - 2) + x + (-z + 2) \;\equiv\; 0 \leq -2 \;.$$

We can therefore construct an LRA-separator I by considering the corresponding linear combination of the inequalities in A. Note that the linear combination, before simplification,

still contained the symbol y that is not common to A and B. After simplification, this symbol has disappeared:

$$0 \leq 2 \cdot (y-1) + (z - x - 2y - 2) \;\equiv\; \underbrace{0 \leq z - x - 4}_{I} \;.$$

⌟

Proof of Lemma 14. The properties $A \models_{\text{LRA}} I$ and $I \wedge B \models_{\text{LRA}} \bot$ hold by construction. To see that $\text{sig}(I) \subseteq (\text{sig}(A) \cap \text{sig}(B)) \cup \Sigma_{\text{LRA}}$, note that I can only contain constant symbols that occur in A. Suppose now that some constant symbol x occurs in $\sum_i \alpha_i t_i$ with non-zero coefficient. Because $\sum_i \alpha_i t_i + \sum_j \beta_j s_j$ is constant, the symbol x then also has to occur in $\sum_j \beta_j s_j$, and x is therefore in $\text{sig}(A) \cap \text{sig}(B)$. ◀

It is common to define this method of computing LRA-separators in terms of proof rules. For this, we can first consider the proof rules shown in Table 1a, which form a calculus for deriving the unsatisfiability of a set of inequalities [51]. The proof rules are defined in terms of *sequents* $\Gamma \vdash \phi$, in which Γ is a finite set of sentences (in our case, set of inequalities), and ϕ is a single sentence. The semantics of a sequent is that the conjunction of the formulas in Γ entails ϕ. The proof rules express that the sequent underneath the bar (the *conclusion*) can be derived from the sequents above the bar (the *premises*).

The rules Hyp and Comb enable the derivation of positive linear combinations of the given inequalities Γ. Rule Con detects unsatisfiable inequalities, i.e., constant inequalities with a negative constant term. The rule Simp simplifies inequalities with the help of a rewriting relation $\rightsquigarrow$, which can be implemented by sorting the terms in an inequality and removing all terms with coefficient 0 [57]. An example proof is given in Fig. 4.

It follows from Lemma 13 that the calculus in Table 1a is sound and complete for showing LRA-unsatisfiability: the sequent $\Gamma \vdash \bot$ can be derived if and only if Γ is a finite LRA-unsatisfiable set of inequalities.

To derive interpolants, we augment the proof rules with *annotations* that reduce, in the root of a proof tree, to an LRA-separator. The resulting interpolation proof rules are similar in spirit to the interpolating proof rules for propositional logic introduced in Chapter 1; like the rules for propositional logic, interpolants are constructed *bottom-up,* starting at the leaves of a proof. We follow the style of interpolating proof rules in [51, 27], but streamline the formulation as we do not consider the combination with other theories at the moment. The interpolating LRA proof rules are given in Table 1b. For each of the proof rules in Table 1a, there are one or multiple interpolating rules, now reasoning about two kinds of sequents. The simpler kind of sequent has the shape $(A, B) \blacktriangleright I$ and contains two finite sets A, B of sentences, as well as a single sentence I. A sequent $(A, B) \blacktriangleright I$ is considered valid if and only if I is an LRA-separator for $\bigwedge A \wedge \bigwedge B$ according to Definition 4. Validity of $(A, B) \blacktriangleright I$ implies that $A \cup B$ is LRA-unsatisfiable.

Table 1 Proof rules for linear real arithmetic (LRA) and linear integer arithmetic (LIA).

$$\textsc{Hyp}\ \frac{}{\Gamma \vdash 0 \le t}\ (0 \le t \in \Gamma) \qquad \textsc{Comb}\ \frac{\Gamma \vdash 0 \le s \qquad \Gamma \vdash 0 \le t}{\Gamma \vdash 0 \le \alpha s + \beta t}\ (\alpha, \beta > 0)$$

$$\textsc{Con}\ \frac{\Gamma \vdash 0 \le \alpha}{\Gamma \vdash \bot}\ (\alpha < 0) \qquad \textsc{Simp}\ \frac{\Gamma \vdash 0 \le s}{\Gamma \vdash 0 \le t}\ (s \rightsquigarrow t)$$

(a) Proof rules for linear real arithmetic. The arrow $\rightsquigarrow$ denotes simplification of an expression.

$$\textsc{Hyp-A}^I\ \frac{}{(A,B) \vdash 0 \le t\,[t]}\ (0 \le t \in A) \qquad \textsc{Hyp-B}^I\ \frac{}{(A,B) \vdash 0 \le t\,[0]}\ (0 \le t \in B)$$

$$\textsc{Comb}^I\ \frac{(A,B) \vdash 0 \le s\,[s'] \qquad (A,B) \vdash 0 \le t\,[t']}{(A,B) \vdash 0 \le \alpha s + \beta t\,[\alpha s' + \beta t']}\ (\alpha, \beta > 0)$$

$$\textsc{Con}^I\ \frac{(A,B) \vdash 0 \le \alpha\,[t']}{(A,B) \blacktriangleright 0 \le t'}\ (\alpha < 0) \qquad \textsc{Simp}^I\ \frac{(A,B) \vdash 0 \le s\,[s']}{(A,B) \vdash 0 \le t\,[t']}\ \begin{array}{l}(s \rightsquigarrow t,\\ s' \rightsquigarrow t')\end{array}$$

(b) Interpolating proof rules for linear real arithmetic.

$$\textsc{Disj-A}^I\ \frac{(A \cup \{K_1\}, B) \blacktriangleright I \qquad (A \cup \{K_2\}, B) \blacktriangleright J}{(A \cup \{K_1 \vee K_2\}, B) \blacktriangleright I \vee J}$$

$$\textsc{Disj-B}^I\ \frac{(A, B \cup \{K_1\}) \blacktriangleright I \qquad (A, B \cup \{K_2\}) \blacktriangleright J}{(A, B \cup \{K_1 \vee K_2\}) \blacktriangleright I \wedge J}$$

(c) Interpolating proof rules for disjunctions.

$$\textsc{BnB}\ \frac{\Gamma \cup \{0 \le n - x\} \vdash \bot \qquad \Gamma \cup \{0 \le x - n - 1\} \vdash \bot}{\Gamma \vdash \bot}\ (x \in \mathrm{sig}(\Gamma), n \in \mathbb{Z})$$

$$\textsc{Strengthen}\ \frac{\Gamma \cup \{t = 0\} \vdash \bot \qquad \Gamma \cup \{0 \le t - 1\} \vdash \bot}{\Gamma \cup \{0 \le t\} \vdash \bot}$$

$$\textsc{Div}\ \frac{\Gamma \vdash 0 \le \alpha_1 x_1 + \cdots + \alpha_k x_k + \alpha_0}{\Gamma \vdash 0 \le (\alpha_1/\beta)x_1 + \cdots + (\alpha_k/\beta)x_k + \lfloor \alpha_0/\beta \rfloor}\ (\beta = \gcd(\alpha_1, \ldots, \alpha_k), \beta > 0)$$

(d) Proof rules for linear integer arithmetic.

$$\textsc{BnB-A}^I\ \frac{(A \cup \{0 \le n - x\}, B) \blacktriangleright I \qquad (A \cup \{0 \le x - n - 1\}, B) \blacktriangleright J}{(A, B) \blacktriangleright I \vee J}\ (x \in \mathrm{sig}(A), n \in \mathbb{Z})$$

$$\textsc{BnB-B}^I\ \frac{(A, B \cup \{0 \le n - x\}) \blacktriangleright I \qquad (A, B \cup \{0 \le x - n - 1\}) \blacktriangleright J}{(A, B) \blacktriangleright I \wedge J}\ (x \in \mathrm{sig}(B), n \in \mathbb{Z})$$

$$\textsc{Div}^I\ \frac{(A,B) \vdash 0 \le \alpha_1 x_1 + \cdots + \alpha_k x_k + \alpha_0\,[t']}{(A,B) \vdash 0 \le (\alpha_1/\beta)x_1 + \cdots + (\alpha_k/\beta)x_k + \lfloor \alpha_0/\beta \rfloor\,[t' \div \beta]}\ (\beta = \gcd(\alpha_1, \ldots, \alpha_k), \beta > 0)$$

(e) Interpolating proof rules for linear integer arithmetic.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash 0 \le z-x-2y-2}\,\textsc{Hyp} \quad \dfrac{}{\Gamma \vdash 0 \le x}\,\textsc{Hyp}}{\Gamma \vdash 0 \le (z-x-2y-2)+x}\,\textsc{Comb}}{\Gamma \vdash 0 \le z-2y-2}\,\textsc{Simp} \quad \dfrac{}{\Gamma \vdash 0 \le y-1}\,\textsc{Hyp}}{\Gamma \vdash 0 \le (z-2y-2)+2(y-1)}\,\textsc{Comb}}{\Gamma \vdash 0 \le z-4}\,\textsc{Simp}}{\mathcal{P}}$$

$$\dfrac{\dfrac{\dfrac{\mathcal{P} \quad \dfrac{}{\Gamma \vdash 0 \le -z+2}\,\textsc{Hyp}}{\Gamma \vdash 0 \le (z-4)+(-z+2)}\,\textsc{Comb}}{\Gamma \vdash 0 \le -2}\,\textsc{Simp}}{\Gamma \vdash \bot}\,\textsc{Con}$$

Figure 4 Proof of unsatisfiability for $\Gamma = \{0 \le y-1, 0 \le z-x-2y-2, 0 \le x, 0 \le -z+2\}$ in Example 15.

The second kind of sequent (also called *inequality interpolation* [51]) has the shape $(A, B) \vdash 0 \le t\,[\,t'\,]$, again containing two sets A, B of sentences, one inequality $0 \le t$, and one term t' forming the annotation. The intuition is that a sequent of this kind corresponds to a non-interpolating sequent $A \cup B \vdash 0 \le t$, but in addition it records the contribution to $0 \le t$ from A. A sequent $(A, B) \vdash 0 \le t\,[\,t'\,]$ is valid if the following conditions hold:

- $A \models_{\mathrm{LRA}} 0 \le t'$,
- $B \models_{\mathrm{LRA}} 0 \le t - t'$, and
- $\mathrm{sig}(t') \subseteq \mathrm{sig}(A) \cup \Sigma_{\mathrm{LRA}}$ and $t - t' \rightsquigarrow s$ with $\mathrm{sig}(s) \subseteq \mathrm{sig}(B) \cup \Sigma_{\mathrm{LRA}}$.

The last condition expresses that t' is a term only containing symbols from A, while the difference $t - t'$ can be simplified to a term only containing symbols from B (ignoring symbols in Σ_{LRA}). In other words, the annotation splits the inequality $0 \le t$ into an A-part and a B-part. The proof rules in Table 1b are formulated in such a way that every sequent that can be derived is valid; for this, $\textsc{Hyp-A}^I$ and $\textsc{Hyp-B}^I$ record whether an inequality comes from A or from B, $\textsc{Comb}^I$ propagates this information, and $\textsc{Simp}^I$ simplifies the annotation t' along with the derived formula $0 \le t$. The rule $\textsc{Con}^I$ concludes the derivation of an LRA-separator: if an unsatisfiable inequality $0 \le t$ has been derived, the annotation t' gives rise to a separator $0 \le t'$. The interpolating version of the proof in Fig. 4 is given in Fig. 5 and produces the same separator $0 \le z - x - 4$ as the reasoning in Example 15.

2.2.2 Beyond Conjunctions of Non-strict Inequalities

Our discussion so far was restricted to conjunctions of non-strict inequalities $0 \le t$ and equations $s = t$. The formulation in terms of interpolating proof rules leads to a flexible framework, however, that can be extended to include further operators:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma^I \vdash\ 0 \le z-x-2y-2\,[\,z-x-2y-2\,]}\ \textsc{Hyp-A}^I \quad \dfrac{}{\Gamma^I \vdash\ 0 \le x\,[\,0\,]}\ \textsc{Hyp-B}^I}{\Gamma^I \vdash\ 0 \le (z-x-2y-2)+x\,[\,(z-x-2y-2)+0\,]}\ \textsc{Comb}^I}{\Gamma^I \vdash\ 0 \le z-2y-2\,[\,z-x-2y-2\,]}\ \textsc{Simp}^I}{\mathcal{P}_1}$$

$$\dfrac{\dfrac{\dfrac{\mathcal{P}_1 \quad \dfrac{}{\Gamma^I \vdash\ 0 \le y-1\,[\,y-1\,]}\ \textsc{Hyp-A}^I}{\Gamma^I \vdash\ 0 \le (z-2y-2)+2(y-1)\,[\,(z-x-2y-2)+2(y-1)\,]}\ \textsc{Comb}^I}{\Gamma^I \vdash\ 0 \le z-4\,[\,z-x-4\,]}\ \textsc{Simp}^I}{\mathcal{P}_2}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{P}_2 \quad \dfrac{}{\Gamma^I \vdash\ 0 \le -z+2\,[\,0\,]}\ \textsc{Hyp-B}^I}{\Gamma^I \vdash\ 0 \le (z-4)+(-z+2)\,[\,(z-x-4)+0\,]}\ \textsc{Comb}^I}{\Gamma^I \vdash\ 0 \le -2\,[\,z-x-4\,]}\ \textsc{Simp}^I}{\Gamma^I \blacktriangleright\ 0 \le z-x-4}\ \textsc{Con}^I$$

Figure 5 Interpolating version of the proof in Fig. 4. We use the abbreviation $\Gamma^I = (A, B) = (\{0 \le y-1, 0 \le z-x-2y-2\}, \{0 \le x, 0 \le -z+2\})$.

- *Boolean structure* in formulas can be handled by introducing further proof rules according to the principles developed in Chapter 5. For completeness, we show the interpolating proof rules for splitting disjunctions in the A- and B-sets in Table 1c. The rules introduce a disjunction or conjunction in the interpolant, depending on whether a formula from A or from B is considered. Similar rules can be defined for conjunctions and negations.
  It is also generally possible to combine an interpolation procedure for a theory T with the propositional methods from Chapter 1, resulting in interpolation procedures that can handle arbitrary quantifier-free formulas [51]. This is the approach implemented in interpolating SMT solvers, which combine a SAT solver for efficient propositional reasoning with theory solvers that only see conjunctions of theory literals [5].
- *Strict inequalities* $0 < t$ in formulas can be handled by replacing Farkas' lemma with Motzkin's transposition theorem [61], which gives a complete criterion for the unsatisfiability of a conjunction of strict and non-strict inequalities. Based on the transposition theorem, it is possible to formulate a lemma similar to Lemma 14 for interpolation in the presence of both strict and non-strict inequalities [60]. An alternative path is to add separate proof rules for equations and uninterpreted functions [51], which can then be used to reason about strict inequalities $0 < t$ by rewriting them to a conjunction $0 \le t \wedge 0 \ne t$.
- *Negated equations* $s \ne t$ can be replaced by disjunctions $0 < s - t \vee 0 < t - s$, which can then be handled by the rules for disjunctions and strict inequalities.

2.2.3 Linear Integer Arithmetic (LIA)

The computation of Craig interpolants becomes a bit more involved when moving from real numbers to the integers. The good news is that the proof rules presented in the previous section remain sound also for integer arithmetic; as a result, also any LRA-separator derived using the rules in Table 1b is a correct LIA-separator [51]. Initial software verification tools based on Craig interpolation therefore resorted to pure LRA reasoning, even for programs operating on integers or bounded machine integers. The LRA proof rules are not complete for integer reasoning, of course, since there are formulas that are satisfiable over the reals but unsatisfiable over integers. To prove or interpolate such formulas, SMT solvers augment the LRA rules with additional LIA-specific rules, which are, however, harder to turn into interpolating rules.

For *linear integer arithmetic,* LIA, we consider formulas that can contain linear inequalities $0 \leq \alpha_1 x_1 + \cdots + \alpha_k x_k + \alpha_0$, where $\alpha_0, \alpha_1, \ldots, \alpha_k \in \mathbb{Z}$ are now integers and $x_1, \ldots, x_k$ are constant symbols of sort *Int*, as well as divisibility statements $k \mid \cdot$ for every $k \in \mathbb{N}$ (see Example 9). The semantics of such formulas is as expected.

▶ **Example 16.** Consider the following families of formulas (where $n \in \mathbb{N}, n > 1$) [45]:

$$A_n \;=\; 0 \leq y + 2nx + n - 1 \wedge 0 \leq -y - 2nx, \qquad B_n \;=\; 0 \leq y + 2nz - 1 \wedge 0 \leq -y - 2nz + n\ .$$

Using more intuitive notation, the formulas can be written as $A_n \equiv -n < y + 2nx \leq 0$ and $B_n \equiv 0 < y + 2nz \leq n$, and it is easy to see that $A_n \wedge B_n$ is satisfiable over the reals, but unsatisfiable over the integers (for any $n > 1$). Up to equivalence, the only LIA-separator for $A_n \wedge B_n$ is the sentence:

$$I_n \;=\; (2n \mid y) \vee (2n \mid y + 1) \vee (2n \mid y + 2) \vee \cdots \vee (2n \mid y + n - 1)\ . \tag{1}$$

Given a LIA signature providing only inequalities and divisibility statements, the size of separators is linear in n and therefore exponential in the size of A_n, B_n. ⌟

There are three different kinds of LIA proof rules commonly implemented in solvers, in addition to the LRA rules that we have already seen in the previous section:

- *Branch-and-bound (BnB)* [61]: a solver first checks the satisfiability of a formula ϕ over the reals instead of the integers, an approach known as *relaxation.* If ϕ is found to be LRA-unsatisfiable, it is necessarily also LIA-unsatisfiable; if, on the other hand, a solution over the real numbers is found, the solver checks whether there are any symbols x that were assigned a non-integer value $\alpha \in \mathbb{R} \setminus \mathbb{Z}$. When such an x exists, the search branches and continues separately for the two cases $x \leq \lfloor \alpha \rfloor$ and $x \geq \lceil \alpha \rceil$. The proof rule BnB representing the branch-and-bound strategy is shown in Table 1d.
  While branch-and-bound is an important proof rule used in solvers, and tends to perform well on many formulas, it does not give rise to a complete calculus for LIA. This can be seen when considering the formulas in Example 16, which have LRA-solutions of unbounded

$$\textsc{Hyp}\ \frac{}{\Gamma \vdash\ 0 \le y + 2nx + n - 1} \qquad \textsc{Hyp}\ \frac{}{\Gamma \vdash\ 0 \le -y - 2nz + n}$$
$$\textsc{Comb}\ \frac{\Gamma \vdash\ 0 \le (y + 2nx + n - 1) + (-y - 2nz + n)}{\textsc{Simp}\ \dfrac{\Gamma \vdash\ 0 \le 2nx - 2nz + 2n - 1}{\textsc{Div}\ \dfrac{\Gamma \vdash\ 0 \le x - z + 0}{\textsc{Simp}\ \dfrac{\Gamma \vdash\ 0 \le x - z}{\mathcal{P}_1}}}}$$

$$\textsc{Hyp}\ \frac{}{\Gamma \vdash\ 0 \le -y - 2nx} \qquad \textsc{Hyp}\ \frac{}{\Gamma \vdash\ 0 \le y + 2nz - 1}$$
$$\textsc{Comb}\ \frac{\Gamma \vdash\ 0 \le (-y - 2nx) + (y + 2nz - 1)}{\textsc{Simp}\ \dfrac{\Gamma \vdash\ 0 \le 2nz - 2nx - 1}{\textsc{Div}\ \dfrac{\Gamma \vdash\ 0 \le z - x - 1}{\mathcal{P}_2}}}$$

$$\textsc{Comb}\ \frac{\mathcal{P}_1 \qquad \mathcal{P}_2}{\textsc{Simp}\ \dfrac{\Gamma \vdash\ 0 \le (x - z) + (z - x - 1)}{\textsc{Con}\ \dfrac{\Gamma \vdash\ 0 \le -1}{\Gamma \vdash\ \bot}}}$$

Figure 6 Schematic cutting planes proof of unsatisfiability for the formulas $\Gamma = \{0 \le y + 2nx + n - 1, 0 \le -y - 2nx, 0 \le y + 2nz - 1, 0 \le -y - 2nz + n\}$ in Example 16.

size, but no LIA-solutions. Regardless of how often the BnB rule is applied, there are still proof branches left in which the formulas in Γ are LRA-satisfiable but LIA-unsatisfiable.

- *Strengthening* [56, 57]: similarly as with branch-and-bound, inequalities $0 \le t$ to be solved can be used to split the search into the cases $t = 0$ and $1 \le t$, expressing the fact that an integer term t cannot have a value strictly between 0 and 1. Strengthening is the core rule used by the Omega test [56], in which the boundary case $t = 0$ is also called a "splinter". By suitably restricting the number of splinters that are introduced, the Omega test uses strengthening for quantifier elimination in LIA.

  The corresponding rule Strengthen is shown in Table 1d. Strengthening can be used to obtain a complete calculus for LIA, but it requires a separate set of rules to solve the equations that stem from the splinters [56, 57]; such rules for equations are not shown here. In Example 16, the Strengthen rule can be applied n times to the first inequality in A_n to obtain an LRA-unsatisfiable set of inequalities, which can be proven using the rules in Table 1a, as well as n splinters that can be handled using equational reasoning.
- *Cuts* [61, 20] allow solvers to round the constant term in inequalities in such a way that no integer solutions are lost. Cuts can be seen as a targeted application of the strengthening rule, restricting application to cases in which the introduced equation $t = 0$ is immediately unsatisfiable. There are different strategies available for cuts, including

Gomory cuts [61, 20] and the cuts-from-proofs method [17], both often used in solvers. A general rule Div implementing cuts is shown as the third rule in Table 1d. The rule Div is applicable if the coefficients of the non-constant terms in an inequality have some factor $\beta > 0$ in common. In this case, all coefficients in the inequality can be divided by β while rounding down the constant term α_0/β, strengthening the inequality. Cuts give rise to a complete calculus for LIA and have the advantage that no branching is needed; proofs including cuts are also called *cutting planes proofs.* To obtain even better performance in practice, many solvers combine cuts with the branch-and-bound method.
We show a cutting planes proof for the formulas in Example 16 in Fig. 6. The proof works for any positive value n and requires two applications of the Div rule, after which a contradiction can be derived using Comb.

All three kinds of proof rules can be made interpolating. We show the interpolating versions of *branch-and-bound* and *cuts* in Table 1e, using the same sequent notation as for the interpolating LRA rules in Table 1b. Interpolating versions of the *strengthening* rule are somewhat more involved, due to the need to introduce also interpolating rules for linear equations [38], but can be derived as well [12, 27].

The simplest rule to interpolate is branch-and-bound, since the branching performed by BnB can be seen as the splitting of a disjunction $0 \leq n - x \vee 0 \leq x - n - 1$ (which is a valid formula in LIA). The two interpolating rules corresponding to BnB are BnB-A^I and BnB-B^I: depending on whether the constant x is a symbol from A or from B, one or the other rule has to be used. The rules BnB-A^I and BnB-B^I correspond to the rules Disj-A^I and Disj-B^I for splitting disjunctions; like the rules for disjunctions, they derive the overall interpolant for (A, B) by combining the sub-interpolants I, J using either a disjunction or a conjunction. If x occurs in both A and B, either rule can be used, but the rules will, in general, lead to different interpolants.

Interpolating cutting planes proofs is more involved, since the inequalities tightened in a proof using the rule Div might contain some symbols that only occur in A, and some symbols that only occur in B, and therefore cannot be associated uniquely with either side. It is possible, however, to extend the annotation approach for LRA interpolation to LIA by allowing the annotation t' in a sequent $(A, B) \vdash 0 \leq t\ [\,t'\,]$ to contain the operator $\div$ for integer division with remainder, provided that the denominator is an integer constant [27, 55]. We use the Euclidean definition of division with remainder [46], in which $a \div b$ for $a, b \in \mathbb{Z}$, $b \neq 0$ is defined to be the unique number $q \in \mathbb{Z}$ such that $a = b \cdot q + r$ with $0 \leq r < |b|$. To handle integer division, it is also necessary to generalize the Simp^I rule by adding a case for pulling terms out of the scope of $\div$, namely $(\alpha s + t) \div \beta \rightsquigarrow (\alpha/\beta)s + (t \div \beta)$ whenever $\beta \mid \alpha$. This rewrite is correct, since the equation $(\beta x + y) \div \beta = x + (y \div \beta)$ holds for Euclidian division, provided that $\beta \neq 0$.

The rule Div^I in Table 1e is deceivingly simple and uses $\div$ to divide the annotation t' in a sequent by the same number β as the main inequality. The effect of Div^I is best illustrated using an example. Fig. 7 shows the interpolating version of the proof in Fig. 6, proving that

$$\dfrac{\dfrac{\dfrac{

the formulas in Example 16 are unsatisfiable for any positive n. The two applications of $\textsc{Div}^I$ introduce a division $\div 2n$ in the annotation term. The annotations are subsequently simplified by applying the rule $\textsc{Simp}^I$, which reduces the terms $2nx$ and $-2nx$ to x and $-x$, respectively, removing the symbol x from the scope of $\div$. Since x and $-x$ cancel each other out, the final LIA-separator is the formula $0 \leq (y + n - 1) \div 2n + (-y) \div 2n$ that only contains the symbol y that is common to A and B. The reader can verify that this separator is equivalent to I_n in Example 16 but exponentially more succinct thanks to the use of $\div$.

The interesting question, of course, is why the rule $\textsc{Div}^I$ always produces correct interpolants. Adapted to the case of LIA, the conditions to be satisfied by an interpolating sequent $(A, B) \vdash 0 \leq t\,[\,t'\,]$ are:

- $A \models_{\text{LIA}} 0 \leq t'$,
- $B \models_{\text{LIA}} 0 \leq t - t'$, and
- $\text{sig}(t') \subseteq \text{sig}(A) \cup \Sigma_{\text{LIA}}$ and $t - t' \rightsquigarrow s$ with $\text{sig}(s) \subseteq \text{sig}(B) \cup \Sigma_{\text{LIA}}$.

We can check, using purely arithmetic reasoning, that the first two conditions hold in any sequent $(A, B) \vdash 0 \leq t\,[\,t'\,]$ derived using the rules in Table 1b and rule $\textsc{Div}^I$. To see that the last conditions are preserved by $\textsc{Div}^I$, note that the ability to rewrite $t - t'$ to a term s with $\text{sig}(s) \subseteq \text{sig}(B) \cup \Sigma_{\text{LIA}}$ implies that whenever a term αx occurs in t, for some x that is not present in B, there has to be a corresponding term αx in t' canceling out αx. When applying $\textsc{Div}^I$, it then has to be the case that α is a multiple of the denominator β. An application of the rule $\textsc{Simp}^I$ can therefore simplify the annotation $(\cdots + \alpha x + \cdots) \div \beta$ to $(\alpha/\beta)x + (\cdots) \div \beta$.

More generally, after applying $\textsc{Simp}^I$, it will never be the case that symbols that do not occur in B are in the scope of the $\div$ operator in the annotation t'. Despite the use of $\div$ in the annotations, the interpolating proof rules are able to eliminate symbols that are present in A but not in B from annotations in more or less the same way as for LRA, and the derived interpolants are correct.

2.3 Further Interpolation Approaches

2.3.1 Proof-Based Interpolation

The extraction of (quantifier-free) T-separators from proofs is possible not only for LRA and LIA, but also for many other theories. The interpolating proof rules for LRA can be extended to handle also the theory of Equality and Uninterpreted Functions (EUF) [51]. The LRA calculus can also be extended to non-linear arithmetic (Non-linear Real Arithmetic, NRA), replacing the witnesses produced by Farkas' lemma by Sum-Of-Squares (SOS) witnesses [16, 65]. NRA interpolants can also be computed with the help of the model-constructing satisfiability calculus MCSAT [40].

Several interpolating calculi have been presented for the theory of arrays [13, 62, 36]. As illustrated in Example 6, one difficulty to be solved with arrays is the fact that the standard version of arrays does not admit (plain or general) quantifier-free interpolation. This problem

can be addressed by extending the theory by a function symbol *diff* satisfying the following axiom:

$$\forall M, M'.\ \big(M \neq M' \rightarrow \mathit{select}(M, \mathit{diff}(M, M')) \neq \mathit{select}(M', \mathit{diff}(M, M'))\big)\ .$$

The axiom implies extensionality of arrays, i.e., two arrays are equal if and only if they have the same contents. In Example 6, we can then construct the quantifier-free reverse interpolant

$$M' = \mathit{store}\big(M, \mathit{diff}(M, M'), \mathit{select}(M', \mathit{diff}(M, M'))\big)\ .$$

2.3.2 Graph-Based Interpolation

In the context of SMT, the special case of the satisfiability of a formula in the empty theory $T_\emptyset$ is commonly called *satisfiability modulo Equality and Uninterpreted Functions (EUF).* EUF is in SMT solvers usually handled with the help of the congruence closure algorithm [2], which decides whether a given equation $s = t$ follows from a finite set E of ground equations, i.e., $E \models_{T_\emptyset} s = t$. For this, congruence closure constructs an equality graph (the *e-graph*) in which the terms and sub-terms of E are the nodes, the equations E and inferred equalities are edges, and a second set of edges is used to represent the sub-term relation. Congruence closure can show the unsatisfiability of a formula ϕ by inferring that some equation $s = t$ follows from the equations present in ϕ, while ϕ also contains the negated equation $s \neq t$. The e-graph represents the proof explaining the unsatisfiability of a formula.

In case of an unsatisfiable conjunction $A \wedge B$ of equations, separators can be extracted from the e-graph [22]. This is done by summarizing paths in the graph; if a path is created only by equations from the formula A, it can be represented by a single equation that is entailed by A, and similarly for equations from B. Reasoning in terms of congruences ($s = t$ implies $f(s) = f(t)$) gives rise to implications in the computed separators.

2.3.3 Interpolation by Reduction

A further paradigm for computing Craig interpolants is *reduction* [41]. Reduction is applicable whenever direct interpolation of a conjunction $A \wedge B$ in some theory T is difficult, but it is possible to translate $A \wedge B$ to a conjunction $A' \wedge B'$ in a theory T' in which interpolants can be computed. The T'-separator I' then has to be translated back to obtain a T-separator I for the original conjunction $A \wedge B$; in the special case that T is an extension of T', back-translation is unnecessary, since I' is already a correct T-separator for $A \wedge B$.

The reduction approach can be applied to a wide range of theories. The theory of arrays and the theory of sets with finite cardinality constraints can be reduced to EUF, while the theory of multisets can be reduced to the combined theory of EUF and LIA [41]. Since reduction can introduce quantifiers, however, it is in general necessary to post-process interpolants using a quantifier elimination procedure for quantifier-free interpolation. In follow-up work, it was shown that the theory of algebraic data types with size constraints can

be reduced to the combined theory of EUF and LIA [37], and that the theory of bit-vectors can be reduced to LIA [3]. The latter reduction happens lazily with the help of tailor-made proof rules that successively translate bit-vector formulas to LIA formulas.

3 Verification Algorithms Based on Interpolation

We now introduce some of the main verification algorithms based on Craig interpolation, starting with the SAT-based model checking algorithm proposed by McMillan in 2003 [49]. We present the algorithms in our setting of quantifier-free interpolation in a theory T, although the earlier papers focus on propositional interpolation.

3.1 BMC: Bounded Model Checking

Interpolation-based verification algorithms were inspired by *Bounded Model Checking (BMC),* in which the behavior of a system during the first k execution steps is analyzed with the help of SAT or SMT solvers [7]. The insight that led to interpolation-based algorithms is that interpolants can be extracted for the formulas considered in BMC, this way turning *bounded* into *unbounded* model checking.

To introduce BMC, we fix a theory T and assume that states of the system to be verified are described by a vector $\bar{x} = (x_1, \ldots, x_n)$ of constants; in the original SAT-based formulation, $\bar{x}$ is a vector of propositional variables. We use the notation $\bar{x}' = (x'_1, \ldots, x'_n)$ for a fresh primed copy of the constants, and similarly $\bar{x}_i$ for copies indexed by i. BMC operates on *transition systems* represented using formulas that model, for instance, sequential hardware circuits. A transition system (B, TR) is given by two sentences $B[\bar{x}]$ and $TR[\bar{x}, \bar{x}']$ defining the *initial states* and the *transition relation,* respectively. The notation $B[\bar{x}]$ expresses that the constants $\bar{x}$ can occur in B, and $B[\bar{y}]$ that the constants $\bar{x}$ are replaced by another vector $\bar{y}$ of terms. Intuitively, any valuation of $\bar{x}$ satisfying $B[\bar{x}]$ is considered an initial state of the system, and valuations of $\bar{x}, \bar{x}'$ that satisfy $TR[\bar{x}, \bar{x}']$ represent a transition from $\bar{x}$ to $\bar{x}'$.

A further sentence $E[\bar{x}]$ is introduced to represent *error states.* The goal of verification is to decide whether any error states are reachable from the initial states. The verification of partial correctness properties, such as the ones discussed in Section 1.1, can be reduced to this form, modeling the states in which postconditions or assertions are violated using the formula E. In BMC, the reachability of error states is characterized by unfolding the transition relation a bounded number of times: error states are reachable if and only if there is a natural number $k \in \mathbb{N}$ so that the sentence

$$B[\bar{x}_0] \;\wedge\; TR[\bar{x}_0, \bar{x}_1] \wedge TR[\bar{x}_1, \bar{x}_2] \wedge \cdots \wedge TR[\bar{x}_{k-1}, \bar{x}_k] \;\wedge\; \big(E[\bar{x}_0] \vee E[\bar{x}_1] \vee \cdots \vee E[\bar{x}_k]\big) \quad (2)$$

is T-satisfiable. As it turns out, applying SAT or SMT solvers to (2) is a very effective way to discover systems in which errors can occur, thanks to the performance of such solvers. If the formula (2) is T-unsatisfiable, in contrast, it is not (at least not in the general case) possible

to tell whether the system can indeed not encounter any errors, or whether the chosen k was simply too small to find an error.

3.2 IMC: Interpolation and SAT-Based Model Checking

3.2.1 The IMC Algorithm

The first interpolation-based model checking algorithm extends BMC with the computation of inductive invariants through Craig interpolation, making it possible to tell conclusively that (2) is T-unsatisfiable *for every* value of $k \in \mathbb{N}$. We refer to the algorithm as the *IMC* algorithm, following the name given in [6]. To prove that the formula (2) is unsatisfiable for every $k \in \mathbb{N}$, it is enough to find a sentence $R[\bar{x}]$, the *inductive invariant,* with the properties discussed in Section 1.1:

- *Initiation:* $B[\bar{x}] \models_T R[\bar{x}]$;
- *Consecution:* $R[\bar{x}] \wedge TR[\bar{x}, \bar{x}'] \models_T R[\bar{x}']$;
- *Sufficiency:* $R[\bar{x}] \wedge E[\bar{x}] \models_T \bot$.

Indeed, assuming that formula (2) holds, due to *Initiation* and *Consecution* it follows that R is true for all states $\bar{x}_0, \ldots, \bar{x}_i$, and due to *Sufficiency* a contradiction with the disjunction of E-formulas follows.

For reasons of presentation, we make the simplifying assumption that the relation TR is *total*, i.e., $\forall \bar{y}.\, \exists \bar{z}.\ TR[\bar{y}, \bar{z}]$ is valid. To obtain candidates for the inductive invariant $R[\bar{x}]$, the IMC algorithm constructs a sequence $R_0, R_1, R_2, \ldots$ of sentences, in such a way that each R_i is an over-approximation of the states that are reachable in at most i steps from the initial states. Naturally, we can set $R_0 = B$. To construct the other sentences, we fix constants $j, k \in \mathbb{N}$ with $j \leq k$, which are the parameters of IMC. Each formula R_{i+1} is then derived from R_i by considering the following conjunction:

$$\underbrace{R_i[\bar{x}_0] \wedge TR[\bar{x}_0, \bar{x}_1]}_{A[\bar{x}_0, \bar{x}_1]} \ \wedge \ \underbrace{\bigwedge_{l=1}^{k-1} TR[\bar{x}_l, \bar{x}_{l+1}] \wedge \bigvee_{l=j}^{k} E[\bar{x}_l]}_{B[\bar{x}_1, \ldots, \bar{x}_k]} \ . \tag{3}$$

Formula (3) corresponds to the BMC query (2), with the difference that the initial states B have been replaced with the ith approximation R_i, and that the range of the disjunction of the $E[\bar{x}_l]$ is parameterized in j. Three cases are possible:

- **Case (i):** Formula (3) is satisfiable and $i = 0$. This implies that $R_i = R_0 = B$ and error states are reachable in the transition system. Formula (2) is then satisfiable as well for the same k.
- **Case (ii):** Formula (3) is satisfiable and $i > 0$. This can happen either because error states are reachable (like in (i)), or because the approximation R_i is too imprecise. Nothing can be said in this situation and the algorithm has to abort.

- **Case (iii):** Formula (3) is unsatisfiable. We can then extract a T-separator $I_i[\bar{x}_1]$ from the conjunction $A[\bar{x}_0, \bar{x}_1] \wedge B[\bar{x}_1, \ldots, \bar{x}_k]$, since $\bar{x}_1$ are the constants that occur in both $A[\bar{x}_0, \bar{x}_1]$ and $B[\bar{x}_1, \ldots, \bar{x}_k]$. By renaming $\bar{x}_1$ to $\bar{x}$, we obtain a sentence $I_i[\bar{x}]$ that is an over-approximation of the states that can be reached in one step from R_i. The next approximation is set to $R_{i+1}[\bar{x}] = R_i[\bar{x}] \vee I_i[\bar{x}]$, weakening R_i to take one further step of the transition system into account.

It turns out that the sequence $R_0, R_1, R_2, \ldots$ often contains some R_i, for small values of i, that are inductive invariants. Intuitively, this is because the sentences R_i are constructed to approximate the reachable states of the transition system. The use of Craig interpolation has the effect of *accelerating* the fixed-point construction, focusing on properties of the states that are sufficient to stay inconsistent with the B-part of (3) (i.e., not reaching any error states within k steps) and ignoring other details that might be part of the formulas B, TR, E but irrelevant for ruling out errors. This effect of acceleration is achieved by obtaining interpolants from proofs, utilizing the fact that a proof is likely to only refer to those facts and constraints that are necessary to show the unsatisfiability of (3), which are exactly the properties needed to avoid error states.

By construction, the sentences $R_0, R_1, R_2, \ldots$ all satisfy the *Initiation* condition of inductive invariants. To check whether any of the sentences R_i satisfies *Consecution*, this condition can be verified explicitly ($R_i[\bar{x}] \wedge TR[\bar{x}, \bar{x}'] \models_T R_i[\bar{x}']$); alternatively, as a sufficient condition, it can also be checked whether it is ever the case that $R_{i+1} \equiv_T R_i$, which implies *Consecution* thanks to the definition of T-separators. *Sufficiency* of a sentence R_i can similarly be verified explicitly ($R_i[\bar{x}] \wedge E[\bar{x}] \models_T \bot$). IMC can also be set up in such a way that all constructed formulas R_i satisfy *Sufficiency*: for this, it has to be verified that none of the initial states $R_0 = B$ are error states, and j must have value 1 [49]. (At this point, the assumption that T is total is required as well.)

To make the approach effective, it is common to require the formulas B, TR, E to be quantifier-free and the considered theory T to admit plain quantifier-free interpolation; this ensures that also the sentences $R_0, R_1, R_2, \ldots$ are quantifier-free and the required checks remain decidable. The original IMC algorithm [49] made use of Craig interpolants extracted from propositional resolution proofs (see Chapter 1).

The parameters j, k can be used to control the behavior of the algorithm. Bigger values of k and smaller values of j lead to more precise over-approximations, since the B-part of (3) ensures that no error states are reachable from the separator $I_i[\bar{x}_1]$ in at least $j - 1$ and at most $k - 1$ steps. Increasing k, in particular, will lead to stronger formulas $I_i[\bar{x}_1]$ being computed, which will reduce the probability of ending up in Case (ii) in later iterations. It can be shown that IMC is even *complete* for computing inductive invariants under certain assumptions [49]:

- The transition system has *bounded reverse depth,* which means that there is a number $d \in \mathbb{N}$ such that whenever E is reachable from some state, then there is also a path from the

state to E that has length at most d.

- The sequence of computed invariant candidates $R_0, R_1, \ldots$ is guaranteed to become stationary, i.e., there is some $i \in \mathbb{N}$ such that $R_i \equiv_T R_{i'}$ for every $i' \geq i$.

If the first assumption is satisfied, it is enough to choose a large enough k and set j to 1 to prevent the algorithm from failing due to Case (ii). If, moreover, the second assumption is satisfied, there will be some R_i such that $R_i \equiv_T R_{i+1}$, which is then guaranteed to pass the *Consecution* condition and give rise to an inductive invariant.

Both assumptions are satisfied by transition systems and invariants in propositional logic. For the second condition, observe that each R_i entails the next R_{i+1}, so that the candidates $R_0, R_1, \ldots$ form an ascending chain. Since there are only finitely many propositional formulas R_i (up to equivalence), every chain $R_0, R_1, \ldots$ eventually becomes stationary. In contrast, for systems with infinite state space, which are modeled using theories T that allow infinite universes, the conditions are typically violated. Empirically, the IMC algorithm tends to perform well even for such systems [6].

3.2.2 Related Interpolation-Based Algorithms

Many different versions and improvements of the SAT-based model checking algorithm IMC have been proposed. In the original formulation, IMC was defined for transition systems and interpolation in propositional logic [49]. IMC was later extended to theories and the setting of software verification [6], a setting that we adopted for this chapter as well.

The interpolation sequence-based model checking algorithm *ISB* uses a similar fixed-point loop as IMC, but applies sequence interpolants (Definition 10) instead of binary interpolants [63]. The key difference between IMC and ISB is in the way how interpolation queries are defined: while IMC includes the previous approximation R_i in the query (3) used to compute the next R_{i+1}, the queries applied by ISB only refer to the sentences B, TR, E defining the transition system and the error states, but ISB increases the considered number of transitions (copies of TR) in each iteration.

Verification algorithms for software programs often take the control structure of a program into account. Instead of a monolithic transition system (B, TR), we would then start from a program described by a set of transition relations arranged by a control-flow automaton. This setting leads to smaller interpolation queries, since only one program path at a time has to be considered. The *Impact* algorithm follows this strategy, using sequence interpolants to obtain invariant candidates for particular program paths [52].

The *IC3* algorithm is even more local and analyzes individual system states instead of the whole transition relation to obtain invariant candidates [9]. While IC3 has similarities with the IMC and ISB algorithms, IC3 does not rely on Craig interpolation anymore but directly infers clauses that are inductive invariants. The *Spacer* algorithm, which is an extension of IC3, uses interpolation as an optional abstraction step [42].

3.3 CEGAR: Counterexample-Guided Abstraction Refinement

Predicate abstraction [24, 4] and *Counterexample-Guided Abstraction Refinement (CEGAR)* [15] provide another perspective on the use of Craig interpolation for verifying systems. Both notions predate the IMC algorithm and the use of interpolation in verification but turned out to be very suitable for a combination with interpolation [34, 50].

3.3.1 Existential Abstractions

We again fix a theory T. For the sake of presentation, we assume that the theory comes with a uniquely defined structure S_T, i.e., that the set $\mathcal{S}_T = \{S_T\}$ in Definition 1 is a singleton set. As the states of a system are described by a vector $\bar{x} = (x_1, \ldots, x_n)$ of constants with sorts $\bar{\sigma} = (\sigma_1, \ldots, \sigma_n)$, the state space of the system is the Cartesian product $D = D_{\sigma_1} \times \cdots \times D_{\sigma_n}$. Since the theory has a unique structure S_T, with slight abuse of notation we can write $B[s]$ to express that a state $s \in D$ satisfies the sentence B, i.e., that s is an initial state, and similarly state by $TR[s, s']$ that there is a transition from s to s'.

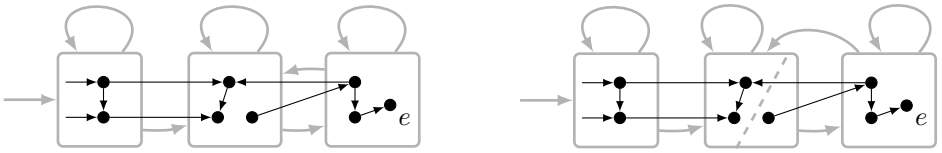
To verify properties of the transition system (B, TR), for instance the reachability of error states E, it is frequently useful to consider an *abstract* version of the transition system with a smaller number of states. We can define abstractions by choosing a non-empty set $\hat{D}$ as the abstract state space, together with an *abstraction function* $h : D \to \hat{D}$ that induces the abstract transition system in terms of a set $\hat{B}$ of abstract initial states and an abstract transition relation $\hat{TR}$:

$$\hat{B} = \{\hat{s} \in \hat{D} \mid \text{there is } s \in D \text{ with } h(s) = \hat{s} \text{ and } B[s]\} \ ,$$
$$\hat{TR} = \{(\hat{s}, \hat{s}') \in \hat{D}^2 \mid \text{there are } s, s' \in D \text{ with } h(s) = \hat{s}, h(s') = \hat{s}' \text{ and } TR[s, s']\} \ .$$

This construction is known as *existential abstraction* [28]. It is useful since non-reachability of states in existential abstractions implies non-reachability also in the concrete system: whenever $M \subseteq D$ and the abstract states $h(M)$ are not reachable from initial states in $(\hat{B}, \hat{TR})$, the concrete states M are not reachable from the initial states in (B, TR) either. More generally, existential abstractions can be used to prove ACTL* properties of the concrete system [28].

Existential abstractions are illustrated in Fig. 8a. Black states and edges represent the concrete transition system (B, TR), with initial states marked with a further incoming edge. The concrete states are grouped together to gray abstract states by the function h. Abstract states are connected by transitions whenever any of the included concrete states are, leading to over-approximation of the behavior. For instance, the concrete transition system in Fig. 8a does not contain any transitions of a state to itself, while the abstract transition system has such transitions.

As a result, the reachability of states in $(\hat{B}, \hat{TR})$ does not imply the reachability of the corresponding states in (B, TR), since the abstract transition system can contain *spurious* paths that have no equivalent paths in (B, TR). For instance, in Fig. 8a there is an abstract



(a) Initial abstraction. **(b)** Abstraction after refinement.

Figure 8 A transition system (black) and two existential abstractions (gray).

path from the left-most initial to the right-most state, but there is no concrete path from an initial state to the right-most state e.

3.3.2 Predicate Abstraction

A convenient way to define existential abstractions is *predicate abstraction* [24]. For this, we assume a finite set $\mathcal{P} = \{P_1[\bar{x}], \ldots, P_k[\bar{x}]\}$ of sentences that can refer to the constants $\bar{x}$ representing system states. The transition system defined using *Boolean abstraction* [4], one of multiple different versions of predicate abstraction, groups together states that cannot be distinguished by the predicates in $\mathcal{P}$. Abstract states can be represented as bit-vectors, $\hat{D} = \{0,1\}^k$, storing for each of the predicates in $\mathcal{P}$ whether the predicate holds for some concrete state $s \in D$ or not:

$$h(s) = (p_1(s), \ldots, p_k(s)), \qquad p_i(s) = \begin{cases} 1 & \text{if } P_i[s] \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i \in \{1, \ldots, k\} \ .$$

For an abstraction with k predicates, this results in a transition system $(\hat{B}, \hat{TR})$ with 2^k abstract states, which can also be constructed explicitly in terms of the sentences B, TR defining the concrete transition system. For this, we define $p^{-1}(\hat{s})[\bar{x}]$ as the conjunction of the (positive or negative) predicates corresponding to an abstract state $\hat{s} = (b_1, \ldots, b_k) \in \hat{D}$:

$$p^{-1}(\hat{s})[\bar{x}] \ = \ \bigwedge_{i=1}^{k} (P_i[\bar{x}] \leftrightarrow b_i) \ .$$

The abstract transition system is then given by:

$$\begin{aligned} \hat{B} \ &= \ \{\hat{s} \in \hat{D} \mid B[\bar{x}] \wedge p^{-1}(\hat{s})[\bar{x}] \text{ is } T\text{-satisfiable}\} \ , \\ \hat{TR} \ &= \ \{(\hat{s}, \hat{s}') \in \hat{D}^2 \mid p^{-1}(\hat{s})[\bar{x}] \wedge TR[\bar{x}, \bar{x}'] \wedge p^{-1}(\hat{s}')[\bar{x}'] \text{ is } T\text{-satisfiable}\} \ . \end{aligned}$$

3.3.3 Counterexample-Guided Abstraction Refinement

The key idea of the CEGAR algorithm is to *refine* abstractions when they are too coarse to prove some desired property of the concrete transition system; this way, the verification

of some system can start with a very small, imprecise abstract transition system (e.g., the system obtained for an empty set $\mathcal{P}_0 = \emptyset$ of predicates) and gradually refine the abstraction until it is possible to prove the property [15]. Refinement can be seen as sub-dividing the states of an abstract transition system, so that spurious paths are eliminated. For instance, in Fig. 8b, the abstract state in the middle has been split into two states, this way removing abstract paths and making the abstract state containing e unreachable from the initial state. In general, multiple abstract states have to be split to remove paths in the abstract transition system. Abstraction refinement can be achieved with the help of Craig interpolation [34].

For this, suppose $E[\bar{x}]$ is a sentence representing error states, and $\hat{E} = \{\hat{s} \in \hat{D} \mid \text{there is } s \in D \text{ with } h(s) = \hat{s} \text{ and } E[s]\}$ is the corresponding abstract set. Suppose that abstract analysis with the predicates $\mathcal{P}_l$ has identified a path $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_m$ of states such that $\hat{s}_1 \in \hat{B}$, $(\hat{s}_i, \hat{s}_{i+1}) \in \hat{TR}$ for all $i \in \{1, \ldots, m-1\}$, and $\hat{s}_m \in \hat{E}$, but no corresponding path exists in the concrete transition system. The latter assumptions means that the formula

$$\underbrace{B[\bar{x}_1]}_{A_1} \wedge \underbrace{TR[\bar{x}_1, \bar{x}_2]}_{A_2} \wedge \underbrace{TR[\bar{x}_2, \bar{x}_3]}_{A_3} \wedge \cdots \wedge \underbrace{TR[\bar{x}_{m-1}, \bar{x}_m]}_{A_m} \wedge \underbrace{E[\bar{x}_m]}_{A_{m+1}}$$

is T-unsatisfiable. The path $\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_m$ is also called a "spurious counterexample" that can be eliminated by adding further predicates, increasing the precision of predicate abstraction. Let $I_0, I_1, \ldots, I_{m+1}$ be a sequence interpolant for the conjunction $A_1 \wedge \cdots \wedge A_{m+1}$. By definition, each I_i for $i \in \{1, \ldots, m\}$ only refers to the symbols $\bar{x}_i$, so that we obtain m new sentences $I_1[\bar{x}], \ldots, I_m[\bar{x}]$ by renaming; those sentences give rise to the refined abstract transition system, defined by $\mathcal{P}_{l+1} = \mathcal{P}_l \cup \{I_1[\bar{x}], \ldots, I_m[\bar{x}]\}$.

In the refined abstract system, the spurious counterexample does not exist any more, in the sense that there is no abstract path $\hat{s}'_1, \hat{s}'_2, \ldots, \hat{s}'_m$ of length m to an error state. This is due to the new predicates: in every initial state $\hat{s} \in \hat{B}$, the bit corresponding to the new predicate I_1 has to be 1; therefore, in every state reachable in one step from an initial state, the bit for I_2 has to be 1, etc. Finally, the predicate I_m, which is present in every state reachable in $m-1$ steps, is inconsistent with the sentence E defining error states, implying that no paths of length m to error states are possible. There might be, of course, paths of different length to error states, which can be removed by further refinement steps.

3.3.4 Variants of Predicate Abstraction and CEGAR

The explicit construction of abstract transition systems tends to be a time-consuming step and is often avoided in newer implementations of CEGAR using the *lazy abstraction* paradigm [35]. Instead of working explicitly with abstract states, lazy abstraction constructs a *reachability graph* in which every node is labeled with a Boolean combination of predicates in $\mathcal{P}$, summarizing a whole set of abstract states. An edge from a node n to n' implies that the formulas ϕ, ϕ' labeling n, n', respectively, satisfy $\phi[\bar{x}] \wedge TR[\bar{x}, \bar{x}'] \models_T \phi'[\bar{x}']$. The reachability graph is built on-the-fly during exploration, starting from the initial states, and refined when error states

are encountered. The Impact algorithm does not even store and reuse predicates, but instead generates new predicates (using Craig interpolation) each time a new path is considered [52].

A further optimization, utilized in most papers on CEGAR, is to take again the control structure of software programs into account (e.g., [35, 52, 33]). Besides leading to smaller interpolation queries, since only one path is considered at a time, sensitivity to control flow enables verification tools to maintain a separate set of predicates for each control location. The even more general framework of constrained Horn clauses, described in the next section, can handle many different kinds of control structure, including procedure calls and concurrency.

3.4 CHC: Craig Interpolation and Constrained Horn Clauses

The framework of *Constrained Horn Clauses* (CHCs) rephrases program verification in terms of model construction in an extension of a theory T with additional predicates [26, 8]. CHCs can be seen as an intermediate language that lets verification front-ends communicate in a standardized way with verification back-ends. CHCs also subsume the extended versions of Craig interpolation (parallel interpolants, sequence interpolants, tree interpolants) that were defined in Section 2.1.3 and let us state a corresponding, more general result.

3.4.1 Definitions and Basic Properties

▶ **Definition 17** (Constrained Horn Clause [59]). *Suppose T is a theory and $\mathcal{R}$ a set of relation symbols disjoint from the signature Σ_T of T. A* Constrained Horn Clause (CHC) *is a sentence $\forall \bar{x}.\ C \wedge B_1 \wedge \cdots \wedge B_n \to H$ in which:*

- *C is a formula over Σ_T that can refer to the variables $\bar{x}$;*
- *each B_i is an application $p(t_1, \ldots, t_k)$ of a relation symbol $p \in \mathcal{R}$ to terms over Σ_T and $\bar{x}$;*
- *H is either $\bot$ or an application $p(t_1, \ldots, t_k)$ of $p \in \mathcal{R}$ to terms over Σ_T and $\bar{x}$.*

We call C the *constraint,* H the *head,* and $C \wedge B_1 \wedge \cdots \wedge B_n$ the *body* of the clause. Constraints $C = \top$ are usually left out. When modeling the partial correctness of a program, the relation symbols $\mathcal{R}$ are used to represent verification artifacts like inductive invariants or contracts, while the clauses model conditions those artifacts have to satisfy. Verification is carried out by *constructing a model* of the clauses, which then includes all the necessary artifacts witnessing the correctness of the program. The algorithms we have encountered in Section 3.2 and Section 3.3 can be adapted to work on sets of CHCs.

This process is formalized using two notions of solvability of clauses. For the sake of presentation, we focus again on theories T that come with a uniquely defined structure S_T, i.e., for which the set $\mathcal{S}_T = \{S_T\}$ in Definition 1 is a singleton set. A set of CHCs is called *semantically solvable* if the structure S_T can be extended to a structure over the signature $\Sigma_T \cup \mathcal{R}$ that satisfies all clauses, i.e., if the set of CHCs is T-satisfiable according

to Definition 1.[1] A set of CHCs is called *syntactically solvable* if all clauses can be made T-valid by uniformly substituting formulas over the signature Σ_T for the relation symbols from $\mathcal{R}$ [59]. Syntactic solvability is related to the classical *solution problem* in logic [64].

▶ **Example 18.** We model the Fibonacci example from Fig. 1a using CHCs. For this, we introduce a relation symbol *loop* that represents the loop invariant of the function `fib` and formulate clauses for the *Initiation, Consecution,* and *Sufficiency* conditions. Clause (4) tells that the invariant should hold when entering the loop, i.e., for any value n with $n \geq 0$ and $a = 0, b = 1, i = 0$. Clause (5) ensures that the invariant is preserved by the loop body: whenever *loop* holds for values n, a, b, i and the loop condition $i < n$ is satisfied, *loop* should also hold for the state after one loop iteration. Clause (6) ensures that whenever *loop* holds for values n, a, b, i and the loop terminates due to $i \geq n$, the postcondition $a \geq 0$ is not violated.

$$\forall n.\ n \geq 0 \rightarrow \mathit{loop}(n, 0, 1, 0) \tag{4}$$

$$\forall n, a, b, i.\ i < n \wedge \mathit{loop}(n, a, b, i) \rightarrow \mathit{loop}(n, b, a + b, i + 1) \tag{5}$$

$$\forall n, a, b, i.\ i \geq n \wedge a \not\geq 0 \wedge \mathit{loop}(n, a, b, i) \rightarrow \bot \tag{6}$$

Every syntactic solution of the clauses corresponds to an inductive loop invariant. The existing CHC solvers, which are often interpolation-based, can easily compute such solutions; one possible solution is the assignment $\{\mathit{loop}(n, a, b, i) \mapsto a \geq 0 \wedge b \geq 0\}$, proving that the program is partially correct. Note that $a \geq 0 \wedge b \geq 0$ coincides with the inductive invariant that was discovered in Section 1.1. ⌟

If a set of CHCs is syntactically solvable, then it is also semantically solvable. The converse is not true in general, because a set of CHCs could only have models that cannot be defined explicitly in the theory T [59]. For the fragment of *recursion-free* CHCs, however, the notions of semantic and syntactic solvability coincide thanks to the Craig interpolation theorem. To illustrate the relationship between CHCs and T-separators, consider formulas $A[\bar{x}, \bar{y}], B[\bar{x}, \bar{z}]$ that only have the variables $\bar{x}$ and symbols from Σ_T in common, while $\bar{y}, \bar{z}$ are local variables that only occur in $A[\bar{x}, \bar{y}]$ or $B[\bar{x}, \bar{z}]$, respectively. Consider then the CHCs $\forall \bar{x}, \bar{y}.\ A[\bar{x}, \bar{y}] \rightarrow I(\bar{x})$ and $\forall \bar{x}, \bar{z}.\ B[\bar{x}, \bar{z}] \wedge I(\bar{x}) \rightarrow \bot$. By definition, any syntactic solution of the two clauses is a T-separator[2] for the conjunction $A[\bar{x}, \bar{y}] \wedge B[\bar{x}, \bar{z}]$, and vice versa, implying that the two CHCs are syntactically solvability if and only if $A[\bar{x}, \bar{y}] \wedge B[\bar{x}, \bar{z}]$ is T-unsatisfiable. In a similar way, sequence interpolants and tree interpolants can be represented using sets of CHCs.

---

[1] For the general case of theories T with multiple structures, two definitions of semantic solvability are commonly used. In *loose semantics* [59, 8], a set of CHCs is considered semantically solvable if *every* T-structure can be extended to a structure satisfying all clauses. Existing CHC solvers more commonly apply the *strict semantics* also used by SMT solvers, which asks whether there exists *some* T-structure that can be extended to a structure satisfying all clauses. CHCs are often formulated in such a way that both notions of semantics lead to the same result.

[2] We consider the free variables $\bar{x}, \bar{y}, \bar{z}$ as constant symbols at this point.

To state this result more generally, for a set $\mathcal{C}$ of CHCs over the relation symbols $\mathcal{R}$ we introduce a *dependence relation* $\rightarrow_{\mathcal{C}} \subseteq \mathcal{R}^2$, defining $p \rightarrow_{\mathcal{C}} q$ if there is a CHC in $\mathcal{C}$ that contains p in the head and q in the body. The set $\mathcal{C}$ is called *recursion-free* if $\rightarrow_{\mathcal{C}}$ is acyclic, and *recursive* otherwise.

▶ **Theorem 19** (Solvability of CHCs [59]). *Let $\mathcal{C}$ be a finite and recursion-free set of CHCs over a theory T. Then $\mathcal{C}$ is semantically solvable*[3] *if and only if it is syntactically solvable.*

To prove the direction "⇒" of this result, the recursion-free CHCs can be expanded to obtain a disjunctive interpolation problem [58], which can be translated further to a tree interpolation problem; the syntactic solution can be extracted from the tree interpolant modulo T. Recursion-free CHCs are exponentially more succinct than tree interpolation problems, however. The problem of determining whether a recursion-free set of CHCs over the theory LIA has solutions is co-NEXPTIME-complete, while checking whether a formula is LIA-unsatisfiable (and interpolants exist) is only co-NP-complete.

The relationship between semantic and syntactic solvability also holds in the quantifier-free setting, provided that the theory T admits quantifier-free interpolation:

▶ **Lemma 20** (Quantifier-free solvability of CHCs). *Suppose T is a theory that admits plain quantifier-free interpolation. Let $\mathcal{C}$ be a finite and recursion-free set of CHCs over T in which the constraints do not contain quantifiers. Then $\mathcal{C}$ is semantically solvable*[4] *if and only if it is syntactically solvable, with a solution that does not contain quantifiers.*

3.4.2 Encoding of Recursive Programs using CHCs

Constrained Horn clauses can be used also for inference tasks beyond just inductive invariants, making it possible, among others, to automatically verify recursive or concurrent programs [26]. Consider the program in Fig. 9, which is a recursive implementation of the Fibonacci function from Section 1.1. Properties of the recursive implementation can be verified by applying the principle of *Design by Contract* [53], describing the behavior of the function using pre- and postconditions that serve a similar purpose as an inductive loop invariant: every recursive call to the function has to satisfy the precondition and can then rely on the postcondition being established by the function upon return.

```
1 int fibR(int n) {
2   if (n == 0)
3     return 0;
4   else if (n == 1)
5     return 1;
6   else
7     return fibR(n-1) + fibR(n-2);
8 }
```

Figure 9 Recursive program computing the nth element of the Fibonacci sequence.

[3] For theories with multiple models, applying *loose semantics* as discussed in Footnote 1.
[4] As in Theorem 19, for theories with multiple models, applying *loose semantics* as discussed in Footnote 1.

We illustrate how this style of reasoning can be used to prove a property about the Fibonacci implementation: whenever $n \geq 3$, the result of the function is a number $\texttt{fibR}(n) \geq 2$. There are several possible ways to formulate CHCs for this recursive function [8]. Choosing an encoding that is close to the classical Design by Contract methodology, we can reason about the function by introducing two relation symbols *pre* and *post*, representing the pre- and postcondition of `fibR`, respectively. The precondition $pre(n)$ is a formula about the function argument n, whereas the postcondition $post(n, r)$ is a relation between the argument n and the result r. We then formulate CHCs that state the properties those symbols have to satisfy:

$$\forall n.\ n \geq 3 \rightarrow pre(n) \tag{7}$$

$$\forall n.\ n = 0 \wedge pre(n) \rightarrow post(n, 0) \tag{8}$$

$$\forall n.\ n \neq 0 \wedge n = 1 \wedge pre(n) \rightarrow post(n, 1) \tag{9}$$

$$\forall n.\ n \neq 0 \wedge n \neq 1 \wedge pre(n) \rightarrow pre(n-1) \tag{10}$$

$$\forall n.\ n \neq 0 \wedge n \neq 1 \wedge pre(n) \rightarrow pre(n-2) \tag{11}$$

$$\forall n, r, s.\ n \neq 0 \wedge n \neq 1 \wedge pre(n) \wedge post(n-1, r) \wedge post(n-2, s) \rightarrow post(n, r+s) \tag{12}$$

$$\forall n, r.\ n \geq 3 \wedge r \not\geq 2 \wedge post(n, r) \rightarrow \bot \tag{13}$$

Clause (7) states that the contract to be derived should include inputs $n \geq 3$, while (13) expresses that the result r for an input $n \geq 3$ should not be less than 2. The other clauses encode the recursive function itself. Clauses (8) and (9) model the `return` statements in lines 3 and 5 by stating that the postcondition has to hold for the result 0 and 1, respectively. The third `return` statement in line 7 performs two recursive calls and gives rise to somewhat more complicated clauses: clauses (10) and (11) ensure that the recursive calls satisfy the precondition of the function, while (12) states that the computed sum $r + s$ has to satisfy the postcondition, assuming that the values r, s produced by the recursive calls satisfy the postcondition. A solution of the clauses is given by the assignment $\{pre(n) \mapsto n \geq 0$, $post(n, r) \mapsto n \geq 0 \wedge r \geq 0 \wedge (r < 2 \rightarrow n = 0 \vee (n = 1 \wedge r = 1) \vee (n = 2 \wedge r = 1))\}$.

The clause (12) is called a *non-linear* clause, since its body contains three atoms $pre(n), post(n-1, r), post(n-2, s)$, whereas all other clauses are *linear* and have at most one body atom. Non-linear clauses naturally occur when translating recursive or concurrent programs to CHCs and are handled by CHC solvers with the help of *tree interpolation.*

4 Conclusions

The chapter has explored the use of Craig interpolation in program verification, as well as some of the techniques that have been developed for effectively computing Craig interpolants in SMT solvers. Craig interpolation is today a standard method applied in verification tools and there are many more directions that could be discussed, but which are beyond the scope of the chapter, including:

- Techniques for deriving Craig interpolants in *combined theories,* for instance in the (signature-disjoint) combination of arrays and LRA. As a sufficient criterion for quantifier-free interpolation in combined theories, the notion of theories being *equality interpolating* has been identified [66]. Equality interpolation was later shown to be equivalent to the model-theoretical notion of *strong sub-amalgamation* [14].
- The use of *uniform interpolants* in program verification, which are in this context also called *covers* [29, 23]. Uniform interpolation can be used to define a model checking algorithm that is similar in spirit to the IMC algorithm (Section 3.2), but in which the computed invariant candidates R_i are exact.
- Methods for *controlling* the result of Craig interpolation. In applications like verification, not all interpolants work equally well, and the ability of algorithms like IMC to compute inductive invariants strongly depends on the right interpolants being found by the applied interpolation procedure. To address this, methods have been developed to control the logical strength [19, 31], size [1], or the vocabulary [18, 47] of interpolants.

Acknowledgments

The author would like to thank Jean Christoph Jung, Sibylle Möhle-Rotondi, and Christoph Wernhard for the careful proof-reading and helpful discussions while preparing this chapter. The work on the chapter been supported by the Swedish Research Council (VR) under grant 2021-06327, the Wallenberg project UPDATE, the Ethereum Foundation under grant FY25-2124, and the EuroProofNet COST action.

References

1 Leonardo Alt, Grigory Fedyukovich, Antti E. J. Hyvärinen, and Natasha Sharygina. A proof-sensitive approach for small propositional interpolants. In Arie Gurfinkel and Sanjit A. Seshia, editors, *Verified Software: Theories, Tools, and Experiments (VSTTE)*, volume 9593 of *LNCS*, pages 1–18. Springer, 2015. doi:10.1007/978-3-319-29613-5_1.

2 Leo Bachmair, Ashish Tiwari, and Laurent Vigneron. Abstract congruence closure. *J. Autom. Reason.*, 31(2):129–168, 2003. doi:10.1023/B:JARS.0000009518.26415.49.

3 Peter Backeman, Philipp Rümmer, and Aleksandar Zeljic. Interpolating bit-vector formulas using uninterpreted predicates and Presburger arithmetic. *Formal Methods Syst. Des.*, 57(2):121–156, 2021. doi:10.1007/S10703-021-00372-6.

4 Thomas Ball, Andreas Podelski, and Sriram K. Rajamani. Boolean and Cartesian abstraction for model checking C programs. In Tiziana Margaria and Wang Yi, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2031 of *LNCS*, pages 268–283. Springer, 2001. doi:10.1007/3-540-45319-9_19.

5 Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1267–1329. IOS Press, 2021. doi:10.3233/FAIA201017.

6 Dirk Beyer, Nian-Ze Lee, and Philipp Wendler. Interpolation and SAT-based model checking revisited: Adoption to software verification. *J. Autom. Reason.*, 69(1):5, 2025. doi:10.1007/S10817-024-09702-9.

7 Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In W. Rance Cleaveland, editor, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1579 of *LNCS*, pages 193–207. Springer, 1999. doi:10.1007/3-540-49059-0_14.

8 Nikolaj S. Bjørner, Arie Gurfinkel, Kenneth L. McMillan, and Andrey Rybalchenko. Horn clause solvers for program verification. In Lev D. Beklemishev, Andreas Blass, Nachum Dershowitz, Bernd Finkbeiner, and Wolfram Schulte, editors, *Fields of Logic and Computation II*, volume 9300 of *LNCS*, pages 24–51. Springer, 2015. doi:10.1007/978-3-319-23534-9_2.

9 Aaron R. Bradley. SAT-based model checking without unrolling. In Ranjit Jhala and David Schmidt, editors, *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 6538 of *LNCS*, pages 70–87. Springer, 2011. doi:10.1007/978-3-642-18275-4_7.

10 Aaron R. Bradley and Zohar Manna. *The calculus of computation - decision procedures with applications to verification.* Springer, 2007. doi:10.1007/978-3-540-74113-8.

11 Angelo Brillout, Daniel Kroening, Philipp Rümmer, and Thomas Wahl. Beyond quantifier-free interpolation in extensions of Presburger arithmetic. In Ranjit Jhala and David A. Schmidt, editors, *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 6538 of *LNCS*, pages 88–102. Springer, 2011. doi:10.1007/978-3-642-18275-4_8.

12 Angelo Brillout, Daniel Kroening, Philipp Rümmer, and Thomas Wahl. An interpolating sequent calculus for quantifier-free Presburger arithmetic. *J. Autom. Reason.*, 47(4):341–367, 2011. doi:10.1007/S10817-011-9237-Y.

13 Roberto Bruttomesso, Silvio Ghilardi, and Silvio Ranise. Rewriting-based quantifier-free interpolation for a theory of arrays. In Manfred Schmidt-Schauß, editor, *Rewriting Techniques and Applications (RTA)*, volume 10 of *LIPIcs*, pages 171–186. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPICS.RTA.2011.171.

14 Roberto Bruttomesso, Silvio Ghilardi, and Silvio Ranise. Quantifier-free interpolation in combinations of equality interpolating theories. *ACM Trans. Comput. Log.*, 15(1):5:1–5:34, 2014. doi:10.1145/2490253.

15 Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003. doi:10.1145/876638.876643.

16 Liyun Dai, Bican Xia, and Naijun Zhan. Generating non-linear interpolants by semidefinite programming. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 364–380. Springer, 2013. doi:10.1007/978-3-642-39799-8_25.

17 Isil Dillig, Thomas Dillig, and Alex Aiken. Cuts from proofs: a complete and practical technique for solving linear inequalities over integers. *Formal Methods Syst. Des.*, 39(3):246–260, 2011. doi:10.1007/S10703-011-0127-Z.

18 Vijay D'Silva. Propositional interpolation and abstract interpretation. In Andrew D. Gordon, editor, *European Symposium on Programming (ESOP)*, volume 6012 of *LNCS*, pages 185–204. Springer, 2010. doi:10.1007/978-3-642-11957-6_11.

19 Vijay Victor D'Silva, Daniel Kroening, Mitra Purandare, and Georg Weissenbacher. Interpolant strength. In Gilles Barthe and Manuel V. Hermenegildo, editors, *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 5944 of *LNCS*, pages 129–145. Springer, 2010. doi:10.1007/978-3-642-11319-2_12.

20 Bruno Dutertre and Leonardo Mendonça de Moura. A fast linear-arithmetic solver for DPLL(T). In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification (CAV)*, volume 4144 of *LNCS*, pages 81–94. Springer, 2006. doi:10.1007/11817963_11.

21 Cormac Flanagan and James B. Saxe. Avoiding exponential explosion: generating compact verification conditions. In Chris Hankin and Dave Schmidt, editors, *Principles of Programming Languages (POPL)*, pages 193–205. ACM, 2001. doi:10.1145/360204.360220.

22 Alexander Fuchs, Amit Goel, Jim Grundy, Sava Krstic, and Cesare Tinelli. Ground interpolation for the theory of equality. *Log. Methods Comput. Sci.*, 8(1), 2012. doi:10.2168/LMCS-8(1:6)2012.

23 Alessandro Gianola. Uniform interpolation for the automated verification of data-aware business processes. In Luca Geatti, Guido Sciavicco, and Alessandro Umbrico, editors, *Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis*, volume 3311 of *CEUR Workshop Proceedings*, pages 73–79. CEUR-WS.org, 2022.

24 Susanne Graf and Hassen Saïdi. Construction of abstract state graphs with PVS. In Orna Grumberg, editor, *Computer Aided Verification (CAV)*, volume 1254 of *LNCS*, pages 72–83. Springer, 1997. doi:10.1007/3-540-63166-6_10.

25 Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science, 2nd Ed.* Addison-Wesley, 1994.

26 Sergey Grebenshchikov, Nuno P. Lopes, Corneliu Popeea, and Andrey Rybalchenko. Synthesizing software verifiers from proof rules. In Jan Vitek, Haibo Lin, and Frank Tip, editors, *Programming Language Design and Implementation (PLDI)*, pages 405–416. ACM, 2012. doi:10.1145/2254064.2254112.

27 Alberto Griggio, Thi Thieu Hoa Le, and Roberto Sebastiani. Efficient interpolant generation in satisfiability modulo linear integer arithmetic. *Log. Methods Comput. Sci.*, 8(3), 2010. doi:10.2168/LMCS-8(3:3)2012.

28 Orna Grumberg. Abstraction and refinement in model checking. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever, editors, *Formal Methods for Components and Objects (FMCO)*, volume 4111 of *LNCS*, pages 219–242. Springer, 2005. doi:10.1007/11804192_11.

29 Sumit Gulwani and Madan Musuvathi. Cover algorithms and their combination. In Sophia Drossopoulou, editor, *European Symposium on Programming (ESOP)*, volume 4960 of *LNCS*, pages 193–207. Springer, 2008. doi:10.1007/978-3-540-78739-6_16.

30 Ashutosh Gupta, Corneliu Popeea, and Andrey Rybalchenko. Solving recursion-free Horn clauses over LI+UIF. In Hongseok Yang, editor, *Programming Languages and Systems (APLAS)*, volume 7078 of *LNCS*, pages 188–203. Springer, 2011. doi:10.1007/978-3-642-25318-8_16.

31 Arie Gurfinkel, Simone Fulvio Rollini, and Natasha Sharygina. Interpolation properties and SAT-based model checking. In Dang Van Hung and Mizuhito Ogawa, editors, *Automated Technology for Verification and Analysis (ATVA)*, volume 8172 of *LNCS*, pages 255–271. Springer, 2013. doi:10.1007/978-3-319-02444-8_19.

32 John Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.

33 Matthias Heizmann, Jochen Hoenicke, and Andreas Podelski. Nested interpolants. In Manuel V. Hermenegildo and Jens Palsberg, editors, *Principles of Programming Languages (POPL)*, pages 471–482. ACM, 2010. doi:10.1145/1706299.1706353.

34 Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Kenneth L. McMillan. Abstractions from proofs. In Neil D. Jones and Xavier Leroy, editors, *Principles of Programming Languages (POPL)*, pages 232–244. ACM, 2004. doi:10.1145/964001.964021.

35 Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Grégoire Sutre. Lazy abstraction. In John Launchbury and John C. Mitchell, editors, *Principles of Programming Languages (POPL)*, pages 58–70. ACM, 2002. doi:10.1145/503272.503279.

36 Jochen Hoenicke and Tanja Schindler. Efficient interpolation for the theory of arrays. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning (IJCAR)*, volume 10900 of *LNCS*, pages 549–565. Springer, 2018. doi:10.1007/978-3-319-94205-6_36.

37 Hossein Hojjat and Philipp Rümmer. Deciding and interpolating algebraic data types by reduction. In Tudor Jebelean, Viorel Negru, Dana Petcu, Daniela Zaharie, Tetsuo Ida, and Stephen M. Watt, editors, *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 145–152. IEEE Computer Society, 2017. doi:10.1109/SYNASC.2017.00033.

38 Himanshu Jain, Edmund M. Clarke, and Orna Grumberg. Efficient Craig interpolation for linear Diophantine (dis)equations and linear modular equations. *Formal Methods Syst. Des.*, 35(1):6–39, 2009. doi:10.1007/S10703-009-0069-X.

39 Ranjit Jhala and Kenneth L. McMillan. Interpolant-based transition relation approximation. *Log. Methods Comput. Sci.*, 3(4), 2007. doi:10.2168/LMCS-3(4:1)2007.

40 Dejan Jovanovic and Bruno Dutertre. Interpolation and model checking for nonlinear arithmetic. In Alexandra Silva and K. Rustan M. Leino, editors, *Computer Aided Verification (CAV)*, volume 12760 of *LNCS*, pages 266–288. Springer, 2021. URL: https://doi.org/10.1007/978-3-030-81688-9_13, doi:10.1007/978-3-030-81688-9_13.

41 Deepak Kapur, Rupak Majumdar, and Calogero G. Zarba. Interpolation for data structures. In Michal Young and Premkumar T. Devanbu, editors, *Foundations of Software Engineering (FSE)*, pages 105–116. ACM, 2006. doi:10.1145/1181775.1181789.

42 Anvesh Komuravelli, Arie Gurfinkel, and Sagar Chaki. SMT-based model checking for recursive programs. *Formal Methods Syst. Des.*, 48(3):175–205, 2016. doi:10.1007/S10703-016-0249-4.

43 George Kourousias and David Makinson. Parallel interpolation, splitting, and relevance in belief change. *Journal of Symbolic Logic*, 72(3):994–1002, 2007. doi:10.2178/jsl/1191333851.

44 Laura Kovács and Andrei Voronkov. Interpolation and symbol elimination. In Renate A. Schmidt, editor, *Automated Deduction (CADE)*, volume 5663 of *LNCS*, pages 199–213. Springer, 2009. doi:10.1007/978-3-642-02959-2_17.

45 Daniel Kroening, Jérôme Leroux, and Philipp Rümmer. Interpolating quantifier-free Presburger arithmetic. In Christian G. Fermüller and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 6397 of *LNCS*, pages 489–503. Springer, 2010. doi:10.1007/978-3-642-16242-8_35.

46 Serge Lang. *Algebra (3. ed.)*. Addison-Wesley, 1993.

47 Jérôme Leroux, Philipp Rümmer, and Pavle Subotic. Guiding Craig interpolation with domain-specific abstractions. *Acta Informatica*, 53(4):387–424, 2016. doi:10.1007/S00236-015-0236-Z.

48 John McCarthy. Towards a mathematical science of computation. In C. M. Popplewell, editor, *Information Processing 1962: Proceedings IFIP Congress 62*, pages 21–28, Amsterdam, 1963. North Holland.

49 Kenneth L. McMillan. Interpolation and SAT-based model checking. In Warren A. Hunt Jr. and Fabio Somenzi, editors, *Computer Aided Verification (CAV)*, volume 2725 of *LNCS*, pages 1–13. Springer, 2003. doi:10.1007/978-3-540-45069-6_1.

50 Kenneth L. McMillan. Applications of Craig interpolants in model checking. In Nicolas Halbwachs and Lenore D. Zuck, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 3440 of *LNCS*, pages 1–12. Springer, 2005. doi:10.1007/978-3-540-31980-1_1.

51 Kenneth L. McMillan. An interpolating theorem prover. *Theor. Comput. Sci.*, 345(1):101–121, 2005. doi:10.1016/J.TCS.2005.07.003.

52 Kenneth L. McMillan. Lazy abstraction with interpolants. In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification (CAV)*, volume 4144 of *LNCS*, pages 123–136. Springer, 2006. doi:10.1007/11817963_14.

53 Bertrand Meyer. Applying "Design by Contract". *IEEE Computer*, 25(10):40–51, 1992. doi:10.1109/2.161279.

54 Rohit Parikh. Beth definability, interpolation and language splitting. *Synth.*, 179(2):211–221, 2011. doi:10.1007/S11229-010-9778-3.

55 Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symb. Log.*, 62(3):981–998, 1997. doi:10.2307/2275583.

56 William Pugh. The Omega test: a fast and practical integer programming algorithm for dependence analysis. In Joanne L. Martin, editor, *Supercomputing (SC)*, pages 4–13, New York, NY, USA, 1991. ACM.

57 Philipp Rümmer. A constraint sequent calculus for first-order logic with linear integer arithmetic. In Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 5330 of *LNCS*, pages 274–289. Springer, 2008. doi:10.1007/978-3-540-89439-1_20.

58 Philipp Rümmer, Hossein Hojjat, and Viktor Kunčak. Disjunctive interpolants for Horn-clause verification. In *Computer Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 347–363. Springer, 2013. doi:10.1007/978-3-642-39799-8_24.

59 Philipp Rümmer, Hossein Hojjat, and Viktor Kunčak. On recursion-free Horn clauses and Craig interpolation. *Formal Methods Syst. Des.*, 47(1):1–25, 2015. doi:10.1007/S10703-014-0219-7.

60 Andrey Rybalchenko and Viorica Sofronie-Stokkermans. Constraint solving for interpolation. *J. Symb. Comput.*, 45(11):1212–1233, 2010. doi:10.1016/J.JSC.2010.06.005.

61 Alexander Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, 1998.

62 Nishant Totla and Thomas Wies. Complete instantiation-based interpolation. In Roberto Giacobazzi and Radhia Cousot, editors, *Principles of Programming Languages (POPL)*, pages 537–548. ACM, 2013. doi:10.1145/2429069.2429132.

63 Yakir Vizel and Orna Grumberg. Interpolation-sequence based model checking. In Armin Biere and Carl Pixley, editors, *Formal Methods in Computer-Aided Design (FMCAD)*, pages 1–8. IEEE, 2009. doi:10.1109/FMCAD.2009.5351148.

64 Christoph Wernhard. The Boolean solution problem from the perspective of predicate logic. In Clare Dixon and Marcelo Finger, editors, *Frontiers of Combining Systems (FroCoS)*, volume 10483 of *LNCS*, pages 333–350. Springer, 2017. doi:10.1007/978-3-319-66167-4_19.

65 Hao Wu, Jie Wang, Bican Xia, Xiakun Li, Naijun Zhan, and Ting Gan. Nonlinear Craig interpolant generation over unbounded domains by separating semialgebraic sets. In André Platzer, Kristin Yvonne Rozier, Matteo Pradella, and Matteo Rossi, editors, *Formal Methods (FM)*, volume 14933 of *LNCS*, pages 92–110. Springer, 2024. doi:10.1007/978-3-031-71162-6_5.

66 Greta Yorsh and Madanlal Musuvathi. A combination method for generating interpolants. In Robert Nieuwenhuis, editor, *Automated Deduction (CADE)*, volume 3632 of *LNCS*, pages 353–368. Springer, 2005. doi:10.1007/11532231_26.

Chapter 14

Interpolation and Query Rewriting

Michael Benedikt
University of Oxford, UK

Abstract

We overview applications of Craig interpolation and Beth definability to simplifying logical expressions or database queries. From the perspective of the theory of interpolation and definability the results give a number of new angles. First, they give a different take on what it means to make definability or interpolation results effective, looking at algorithms that take a proof as input and return an interpolant or explicit definition as output. Secondly, they relate interpolation and definability to preservation theorems in model theory: interpolation and definability theorems are the basis for many "semantics-to-syntax" results, relating a semantic property of a formula to its equivalence with a certain syntactic form. Thirdly, they motivate new forms of interpolation and definability, focusing on syntactic forms that are of interest in databases.

Contents

1 Overview 504

2 Preliminaries 508

3 First Example: Query Reformulation over a Subvocabulary 511
3.1 Reformulating Queries over Restricted Vocabularies: Definitions 511
3.2 From a Semantic Property to a First-Order Reformulation 511
3.3 The Interpolation Theorem We Need: Relativized-Quantifier Interpolation . . 514

4 Vocabulary-Based Reformulation with Positive Existential Queries . . . 515

5 Vocabulary-Based Existential Reformulation 518

6 Rewriting with Access Patterns 520
6.1 Basics of Target Restrictions Based on Access Methods 521
6.2 The Interpolation Result Behind the Rewriting Results: Access Interpolation 528
6.3 Variants of the Results for $USPJ^{\neq}$-Plans 533

7 State of Play and Future Directions 534

8 Bibliographic Remarks 535

References 536



© Michael Benedikt;
licensed under Creative Commons License CC BY 4.0

In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 14, pages 503–537. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.n

1 Overview

There are a number of applications of Craig interpolation in data management, but this chapter will focus on one type of application, to *rewriting queries*. The idea is that we want to translate a *declarative source query* into a declarative query or procedural plan in a *target language* that abides by certain *interface restrictions*. By a declarative source query we will usually mean a request for information from a database, where the request is specified using a formula of first-order logic, or its equivalent in the database query language SQL. We often focus on source queries given in the language of *conjunctive queries*, which corresponds to a very simple subset of SQL. In logical terms these correspond to formulas built up using only conjunction and existential quantification. By a plan in a target language we may mean either another declarative query, or something more operational, such as a relational algebra expression, possibly using some restricted set of physical operators.

Translating from a declarative source query to a target in this broad sense captures a huge number of activities that take place within data management. One of the basic features of modern database systems is to decouple the vocabulary employed by users to ask questions about a dataset from the data structures that are used to implement data access: the "logical model" and the "physical model". In relational databases, the logical model might be a set of tables given an SQL schema: users interact with the data by sending SQL queries that mention these tables. The database manager performs a translation of the queries written over this high-level vocabulary into a program that interacts with the data using some set of data access functions.

In this "classical query evaluation" scenario, the relationship of the source vocabulary to the target vocabulary is very simple: every source relation corresponds to one or more "physical datasource" relations or functions. A more complicated example comes from data integration systems. The goal there is to mask a diverse set of datasources by providing a single unified schema. Suppose we have a multitude of "local databases" with information about different hotel chains. A data integration system would present to users a "global schema" — an easier-to-understand vocabulary that can model information in all of the sources. The system accepts queries written over the global schema and translates them into queries over the various local sources. The "global database" is virtual, implicitly defined by its relationship with local data.

Having talked generally about what we mean by sources and targets, we mention several flavors of languages that represent targets for a rewriting algorithm. The most basic kind of restriction we look at is a *vocabulary-based restriction*. There the source is a query represented as a logical formula, and the target is also a logical formula, but we limit the target's vocabulary. We begin with a logic-based query Q written using some relations $R_1 \dots R_j$, and want to convert it to another logic-based query $Q_{\vec{V}}$ making use of a different set of relations $V_1 \dots V_k$. If the queries Q and $Q_{\vec{V}}$ mention different relations, we cannot expect Q to give the same answers as $Q_{\vec{V}}$ on arbitrary instances. But our schema will come with *integrity*

constraints — usually given as a set of first-order sentences — which restrict the possible instances of interest. We will thus be considering equivalence only on instances satisfying the constraints.

An example of vocabulary-based restriction comes from *reformulating queries over views*. We have a collection of view relations $V_1 \dots V_k$, and each V_i is associated with a query Q_i defined using some other set of relations $R_1 \dots R_j$. Given a query Q defined over $R_1 \dots R_j$, the goal is to find an equivalent query $Q_{\vec{V}}$ that mentions only $V_1 \dots V_k$. Generally, additional restrictions will be put on $Q_{\vec{V}}$. For example, it should be a conjunctive query, or a union of conjunctive queries, or a relational algebra query. The view-based query reformulation problem can thus be seen as a special case of vocabulary-based restriction, where the integrity constraints are of the form:

$$\forall x_1 \dots \forall x_n \; V_i(x_1 \dots x_n) \leftrightarrow Q_i(x_1 \dots x_n)$$

for $1 \leq i \leq k$. Here the constraint with $\leftrightarrow$ could be rewritten to be a conjunction of two constraints with $\rightarrow$, one in each direction, and we will use this version below.

▶ **Example 1.** A university database has a table Professor containing ids and last names of professors, along with the name of the professor's department. It also has a table Student listing the id and last name of each student, along with their advisor's id.

The database does not allow users to access the Professor and Student tables directly, but instead exposes a view Professor′ where the id attribute is dropped, and a table Student′ where the advisor's id is replaced with the advisor's last name.

That is, Professor′ is a view defined by the formula:

$$\{ \mathsf{lname},\ \mathsf{dname} \mid \exists\ \mathsf{profid}\ \mathsf{Professor}(\mathsf{profid},\ \mathsf{lname},\ \mathsf{dname})\}$$

or equivalently by the constraints:

$$\forall \mathsf{profid}\ \forall \mathsf{lname}\ \forall \mathsf{dname}\ \mathsf{Professor}(\mathsf{profid}, \mathsf{lname}, \mathsf{dname}) \rightarrow \mathsf{Professor}'(\mathsf{lname}, \mathsf{dname})$$
$$\forall \mathsf{lname}\ \forall \mathsf{dname}\ \mathsf{Professor}'(\mathsf{lname}, \mathsf{dname}) \rightarrow \exists \mathsf{profid}\ \mathsf{Professor}(\mathsf{profid}, \mathsf{lname}, \mathsf{dname})$$

Student′ is a view defined by the formula:

$$\{\mathsf{studid}, \mathsf{lname}, \mathsf{profname} \mid$$
$$\exists \mathsf{profid}\ \exists \mathsf{dname}\ \mathsf{Student}(\mathsf{studid}, \mathsf{lname}, \mathsf{profid}) \wedge \mathsf{Professor}(\mathsf{profid}, \mathsf{profname}, \mathsf{dname})\}$$

or equivalently by constraints:

$$\forall \mathsf{studid}\ \forall \mathsf{lname}\ \forall \mathsf{dname}\ \forall \mathsf{profid}\ \forall \mathsf{profname}\ [\mathsf{Professor}(\mathsf{profid}, \mathsf{profname}, \mathsf{dname}) \wedge \mathsf{Student}(\mathsf{studid}, \mathsf{lname}, \mathsf{profid}) \rightarrow \mathsf{Student}'(\mathsf{studid}, \mathsf{lname}, \mathsf{profname})]$$

$$\forall \mathsf{studid}\ \forall \mathsf{lname}\ \forall \mathsf{profname}\ [\mathsf{Student}'(\mathsf{studid}, \mathsf{lname}, \mathsf{profname}) \rightarrow$$
$$\exists \mathsf{profid}\ \exists \mathsf{dname}\ \mathsf{Professor}(\mathsf{profid}, \mathsf{profname}, \mathsf{dname}) \wedge \mathsf{Student}(\mathsf{studid}, \mathsf{lname}, \mathsf{profid})]$$

Consider the query asking for the names of the advisors of a given student. We can answer this by simply using the $\mathsf{Student}'$ view, returning the $\mathsf{profname}$ attribute of a tuple in the view. A formula over relational expression that uses $\mathsf{Student}'$ to answer the query is called a *reformulation of the query over the views*. More generally it is an example of a *reformulation of the query over the subvocabulary* $\mathsf{Student}'$*, relatively to the background theory consisting of constraints relating* $\mathsf{Student}'$ *to the* $\mathsf{Student}$.

But a query asking for the last names of all students that have an advisor in the history department can *not* be answered using the information in the views: knowing the advisor's name is not enough to identify the department.

Integrity constraints need not be restricted to expressing view definitions. A natural use of constraints is to represent *relationships between sources*, such as overlap in the data. This overlap can be exploited to take a query specified over a source that *a priori* does not have sufficient data, and reformulate it over a source that provides the necessary data. ⌟

Access Methods. A finer notion of interface than vocabulary-based restrictions is based on *access methods*, which state that a relation can only be accessed by lookups where certain arguments must be provided. One example of a restricted interface based on access methods comes from web forms. Thinking of the form as exposing a virtual table, the mandatory fields must be filled in by the user, while submitting the form returns all tuples that match the entered values. Other examples include web services and legacy databases.

▶ **Example 2.** Consider a $\mathsf{Profinfo}$ table available via a web form, containing information about faculty, including their last names, office number, and employee id, but with a restricted interface that requires giving an employee id as an input. The interface could be implemented as a web form that requires entering an employee's id and then pressing a submit button to get the matching records. The query Q asking for ids of faculty named "Smith" cannot be "completely answered" over this schema: that is, there is no function over the available data in this schema which is equivalent to Q.

But suppose another web form provides access to a $\mathsf{Udirectory}$ table containing the employee id and last name of every university employee, with an interface that allows one to access the entire contents of the table. Then we can reason that Q is answerable using the information in this schema: a plan would pull tuples from the $\mathsf{Udirectory}$ form and check them within the $\mathsf{Profinfo}$ form to see if they correspond to a faculty member. ⌟

In Example 2, reasoning about access considerations was straightforward, but in the presence of more complex schemas, we may have to chain several inferences, resulting in a plan that makes use of several auxiliary accesses.

▶ **Example 3.** Suppose we have two directory data sources with overlapping information. One source exposes information from $\mathsf{Udirectory}_1(\mathsf{uname}, \mathsf{addr}, \mathsf{uid})$ via an access method requiring a uname and uid. There is also a "web table" $\mathsf{Ids}(\mathsf{uid})$ with no access restriction, that makes available the set of uids (hence we have a "referential integrity constraint" saying that every uid

in $\mathsf{Udirectory}_1$ matches a uid in Ids). The other source exposes $\mathsf{Udirectory}_2(\mathsf{uname}, \mathsf{addr}, \mathsf{phone})$, requiring a uname and addr, and also a web table $\mathsf{Names}(\mathsf{uname})$ with no access restriction that reveals all unames in $\mathsf{Udirectory}_2$ (that is, a constraint that each uname in $\mathsf{Udirectory}_2$ appears in Names). There is also a constraint saying that each uname and addr in $\mathsf{Udirectory}_2$ appears in $\mathsf{Udirectory}_1$.

A query asking for all phone numbers in the second directory could be written:

$$Q = \{\mathsf{phone} \mid \exists \mathsf{uname}\ \exists \mathsf{addr}\ \mathsf{Udirectory}_2(\mathsf{uname}, \mathsf{addr}, \mathsf{phone})\}.$$

There is a plan that implements this query: it gets all the uids from Ids and unames from Names first, puts them into the access on $\mathsf{Udirectory}_1$, then uses the uname and addr of the resulting tuples to get the phone numbers in $\mathsf{Udirectory}_2$. ⌟

We emphasize that the notion of rewriting we consider here is getting target queries which give the same answers as the original query. This means that if we have a query asking for the office number of all professors with last name "Smith", the plan produced should return all tuples in the answer, even if access to the $\mathsf{Professor}$ relation is limited. This contrasts with a line of work in data integration that considers the broader question of how to answer any query "as much as possible given the available data": how to get the *certain answers* or how to compute the *maximally contained query* [21].

Rewriting via Interpolation: the Meta-Algorithm. We will now explain the connection of rewriting to interpolation and Beth definability. There is a general template for solving a rewriting problem:

1. Isolate a *semantic property* that any input query Q must have with respect to the target $\mathcal{T}$ and constraints Σ in order to have a reformulation of the desired type.
2. Express this property as a *proof goal* (in the language we use later on, an *entailment*): a statement that a certain formula follows from another formula.
3. Search for a proof of the entailment within a given proof system. For example, one could use tableau proofs or resolution proofs, both well-known proof systems within computational logic.
4. From the proof, extract an *interpolant* using an interpolation algorithm. We will review some standard interpolation algorithms and also present new ones.
5. Convert the interpolant to a reformulation.

As we will see below, the properties of an interpolant will play a role in the ability to convert to a reformulation, and different notions of interpolation, which control how similar an interpolant is to the statements we interpolate over, will be needed for different variations of reformulation. This approach is very general and can be applied to a variety of proof systems and restrictions on the target, with different target languages corresponding to different entailments. We prove theorems saying that the method is complete: there is a

reformulation exactly when there is a proof of the semantic property. These completeness theorems give as a consequence a *definability or preservation theorem*: a query Q has a certain kind of reformulation if and only if it has a certain semantic property. Such theorems are well-known in model theory, and indeed our theorems can be seen as "database versions" of the preservation theorems that are known from classical model theory textbooks, e.g. [12].

The fact that interpolation theorems can be used to prove preservation theorems is certainly not new. As mentioned earlier in this volume, William Craig used interpolation to prove Beth's definability theorem, and all of our results can be seen as generalizations of Craig's technique: see the bibliographic remarks at the end of this chapter for further details.

What we emphasize in this chapter, as in [6] is that the connection between reformulation, interpolation, and preservation can be made effective: *interpolation algorithms yield reformulation algorithms*. Actually, the interpolation technique yields many of the algorithmic results about reformulation that had appeared in the database setting previously.

2 Preliminaries

We start with some preliminaries about relational databases and logic. A good reference for this is [2].

Structures, Instances, and Range of Quantification. A *schema* is a finite set of relation symbols, each associated to a number (the arity), along with a set of constant symbols. A database *instance* I for a schema Sch assigns to every relation R in Sch of arity n a collection of n-tuples $[\![R]\!](\mathsf{I})$.

We call $[\![R]\!](I)$ the *interpretation* of R in I. An association of a database relation R with a tuple $\vec{c}$ of the proper arity will be referred to as a *fact*. A database instance can equivalently be seen as a collection of facts. An instance of a schema that has only a single relation R is a *relation instance*. The *active domain* of an instance I, denoted $\mathsf{adom}(\mathsf{I})$, is the union of the one-dimensional projections of all interpretations of relations: that is, all the elements that participate in some fact of I.

We can extend the notion of a schema to allow constant symbols: now an instance also maps each constant symbol to a value. We can also extend the notion of a schema to allow *integrity constraints*: in general these are an arbitrary subset of the instances. An instance of a schema with constraints is one where all integrity constraints of the schema are satisfied.

▶ **Example 4.** Suppose our schema consists of only one unary relation $\mathsf{UEmployee}$, containing the ids of university employees. One possible instance I interprets $\mathsf{UEmployee}$ by the singleton set $\{e_0\}$. We can alternatively define I by the set of facts $\{\mathsf{UEmployee}(e_0)\}$. In this case we have $\mathsf{adom}(\mathsf{I}) = \{e_0\}$. ⌟

A *query* (over a given vocabulary) and output arity m is a function from instances of the schema to a set of m-tuples. A *Boolean query* is the case where $m = 0$: thus the output is a

set of 0-tuples. There are only two 0-tuples: the empty set, which we identify with False, and the singleton 0-tuple, which we identify with True. If $m > 0$ we talk of a *non-Boolean query*.

Classical logic considers *structures* rather than *instances*. A structure consists of a set, the *domain* of the structure, interpretations for each relation as sets of tuples with values in the domain, and an interpretation for each constant as a single element of the domain.

▶ **Example 5.** Returning to the schema in Example 4, one structure M that conforms to it consists of a two-element domain $\{e_0, f_0\}$, with UEmployee interpreted as $\{e_0\}$. ⌟

In the *classical semantics* of first-order logic, quantifiers range over the domain of the structure. In databases, quantifiers are usually given the *active domain* semantics, in which the quantified variable ranges over the union of the values in the interpretations of relations. The active domain semantics can be used to give a meaning to a sentence in an instance, since the meaning only depends on the instance, not some domain in which the instance sits.

A *relational atom* is an expression $R(u_1 \dots u_n)$ where R is an n-ary relation symbol and the u_i are either constants or variables. We will use *first-order logic with relativized quantifiers*, RQFO. RQFO is built up from equality and relational atoms via the Boolean operators and the quantifications:

$$\exists \vec{x}\ R(\vec{s}, \vec{x}) \wedge \varphi(\vec{s}, \vec{x}, \vec{t})$$

and

$$\forall \vec{x}\ R(\vec{s}, \vec{x}) \rightarrow \varphi(\vec{s}, \vec{x}, \vec{t})$$

for R a relational symbol and φ an RQFO formula. Those familiar with active domain semantics will be able to see that it is subsumed by RQFO, in the sense that an active domain semantics formula can be converted to an equivalent RQFO formula under classical semantics: the active domain interpretation of the quantification $\exists x\ \varphi(x, \vec{y})$ is a disjunction $\bigvee_i \exists \vec{u}_i\ R_i(\vec{u}_i) \wedge \varphi(x, \vec{y})$, where x is one of the variables in each $\vec{u}_i$. Similarly a quantification $\forall x\ \varphi$ under active domain semantics translates into a conjunction of relativized universal quantifications.

We can similarly talk about equality-free RQFO formulas, by disallowing equality atoms. By convention, we allow the prefix of existential quantifiers $\exists x_1\ \dots\ \exists x_n$ to be empty, and similarly for universal quantifiers. In this way we can capture an atom $R(x_1 \dots x_n)$ and negated atoms $\neg R(x_1 \dots x_n)$ as specialized RQFO formulas $R(\vec{x}) \wedge \mathsf{True}$ and $R(\vec{x}) \rightarrow \mathsf{False}$, respectively. Thus for equality-free formulas we can in fact assume that the only base formulas are True and False.

The truth value of an RQFO sentence without constants is well-defined over an instance, since from the instance we can determine the range of each relativized quantifier. Similarly the truth value for an RQFO formula without constants is well-defined given an instance and a mapping of the free variables to some values (a *variable binding* or just *binding* for short).

An RQFO formula is in *Negation Normal Form* (NNF) if negation is only applied to relational atoms. By simple rules, one can convert any RQFO formula into NNF.

Relational Algebra. RQFO gives a natural way of defining Boolean queries over finite instances. For non-Boolean queries we review another standard formalism. Given a relational vocabulary, we can build up *relational algebra* a language for defining queries. A relational algebra expression is associated with an output type, which is a finite set of attribute names. The atomic expressions are the relation symbols for names in the vocabulary, and for a relation symbol R of arity n, the output type has names $\{1 \dots n\}$. The language is closed under the operations:

- difference, intersection, and union: if E and F are expressions having the same output T, $E - F$, $E \cup F$ and $E \cap F$ are new expressions also having type T
- projection: if E is an expression with output type $a_1 \dots a_n$ and S is a subset of $a_1 \dots a_n$ then $\pi_S(E)$ is an expression with output type S
- renaming: if E is an expression with output type S and F is a bijection from S to S' then $\rho_F(E)$ is an expression with output type S'
- product: if E and F are expressions with disjoint output types S and S', then $E \times F$ is an expression with output type $S \cup S'$
- selection: if E is an expression with output type $\{a_1 \dots a_k\}$ then $\sigma_{a_i=a_j} E$ is an expression with the same output type.

The semantic function for relational algebra maps each term E with output type S to a query whose input is an instance over the input vocabulary and whose output is an instance of the signature consisting of an S-ary relation symbol, along with a mapping of positions to elements of S. For atomic terms it is the identity. The inductive definition of each operator is also straightforward.

We note a superficial difference between relational algebra and logic-based formalisms like RQFO. Relational algebra uses *named notation*: it deals with instances where in each tuple of a relation, the different components are referred to via attribute names. As opposed to logical formalisms, where we have *positional notation*: each relation has some arity m and different components are referred to by a position $1 \leq i \leq m$. It is easy to map between named and positional notation. Below we will abuse notation slightly and talk about a formalism in one notation being equivalent in expressiveness to a formalism in the other: this is up to some canonical way of mapping between names and positions. One can find the formalization of such mappings in [2].

A term with empty output type returns either the empty instance or the instance consisting of a single empty tuple: it can thus be considered as a Boolean query.

▶ **Proposition 6.** *A relational algebra expression of Boolean type can be mapped to an equivalent* RQFO *expression, and vice versa.*

▶ **Proposition 7.** *The output of a relational algebra expression on a finite instance is a finite relation.*

A query is *safe* if on every finite instance, there are only finitely many variable bindings that satisfy it.

There is a converse to Proposition 7: for any RQFO formula, if the query it defines is safe, then there is a relational algebra expression that represents the same query.

A relational algebra expression that does not use the difference operator is said to be a $USPJ^{\neq}$ expression (for union-selection-projection-join). If further it does not use union, it is an $SPJ^{\neq}$ expression.

3 First Example: Query Reformulation over a Subvocabulary

We now drill down into the first example of reformulation-via-interpolation: vocabulary-based reformulation.

3.1 Reformulating Queries over Restricted Vocabularies: Definitions

Let $\mathcal{S}$ be a collection of relations, Σ a set of integrity constraints, and $\mathcal{T}$ a subset of $\mathcal{S}$. Given a query Q specified by a logical formula with n free variables over $\mathcal{S}$, our first goal will be to get a *relativized-quantifier first-order reformulation of Q over $\mathcal{T}$ with respect to* Σ. This means a query $Q_{\mathcal{T}}$ given by a relativized-quantifier first-order formula using only the relations in $\mathcal{T}$ such that for every instance I satisfying Σ,

$$\forall x_1 \dots x_n \; Q(x_1 \dots x_n) \leftrightarrow Q_{\mathcal{T}}(x_1 \dots x_n)$$

holds in I. We use the same notation when Q is a relational algebra query, identifying its output attributes with free variables. Thus a reformulation is another query $Q_{\mathcal{T}}$ that is *equivalent to Q w.r.t.* Σ or *answers Q w.r.t.* Σ, meaning that they have the same output on instances satisfying Σ.

3.2 From a Semantic Property to a First-Order Reformulation

Let us return to the "meta-algorithm" for reformulating a query Q with respect to a target language and a set of constraints, mentioned in the introduction. Recall that step (1) is to identify a property that Q must have in order to admit the desired reformulation. Clearly, for Q to have any reformulation over $\mathcal{T}$, its output should depend only on the interpretations of the relations in $\mathcal{T}$. This notion of "implicit definability" has been formalized by Segoufin and Vianu in [25], as the notion of *determinacy* for database views and queries. We extend the definition here to the setting where there are constraints given by RQFO sentences.

If Σ is a collection of RQFO constraints, we say that an RQFO query Q over $\mathcal{S}$ is *determined over $\mathcal{T}$ relative to* Σ if:

> For any two instances I and I' that satisfy Σ and have the same interpretation of all relations in $\mathcal{T}$ (that is, they have the same T-facts for each $T \in \mathcal{T}$) $[\![Q]\!](I) = [\![Q]\!](I')$.

▶ **Example 8.** We consider the case where the vocabulary consists of the Department and Employee relations, the constraints Σ are inclusion dependencies from Employee to Department

(on deptid) and from Department to Employee (on mgrid and deptid), while the target signature $\mathcal{T}$ consists of only Department.

Suppose our query Q asks for the ids of all employees. We claim that Q is *not* determined over $\mathcal{T}$ relative to Σ. To see this, consider two instances I and I', such that I consists of facts:

$$\{\mathsf{Employee}(123, \text{“Jones”}, 1112), \mathsf{Employee}(134, \text{“Smith”}, 1112),$$
$$\mathsf{Department}(1112, \mathsf{mathematics}, 134)\}$$

and I' consists of facts

$$\{\mathsf{Employee}(134, \text{“Smith”}, 1112), \mathsf{Department}(1112, \mathsf{mathematics}, 134)\}$$

I and I' both satisfy the schema constraints, and they have the same restriction to Department, but the output of Q on I includes employee id 123, while Q evaluated on I' does not.

We have shown via witnesses I and I' that Q is not determined over $\mathcal{T}$ *w.r.t.* Σ. ⌟

A special case of interest is where the relations in the restricted vocabulary $\mathcal{T}$ are “virtual tables” defined by *view definitions*. For each $V \in \mathcal{T}$ of arity n there is a corresponding query $Q_V(x_1 \dots x_n)$, while the constraints Σ consist only of the statements that a view contains exactly those tuples that satisfy its definition: $\forall \vec{x}\ V(\vec{x}) \leftrightarrow Q_V(\vec{x})$. In this case, determinacy of another query Q with respect to $\mathcal{T}$ and Σ can be rephrased as:

> For any two instances I and I', if I and I' yield the same results for each Q_V, then they yield the same results for Q.

Segoufin and Vianu use the terminology “Q is determined by the views $\{Q_V : V \in \mathcal{T}\}$”. We will apply our methodology to the semantic property of determinacy, turning it into a proof goal, and showing that reformulations can be read off from these proofs witnessing the proof goal.

Translating the Semantic Property to an Entailment. Returning to Step (2) of our “meta-algorithm” from the introduction, we write out the semantic property (in this case, determinacy) as a proof goal. We now formalize what this means.

Recall that an *entailment* is a statement of the form $\lambda(\vec{x}) \models \rho(\vec{x})$, which means that for every instance I and any binding vbind of $\vec{x}$, if instance I with binding vbind satisfies λ then it satisfies ρ. By convention, we write $\models \rho$ to mean True $\models \rho$: that is, ρ holds in every structure. Let us extend our original signature for the constraints Σ and the query Q by making a copy R' of every relation $R \in \mathcal{S}$. Let Q' be the copy of Q on the new relations, and Σ' be the copy of the constraints Σ on the new relations. Our assumption of determinacy of Q can be restated as an entailment:

$$\models \forall \vec{x}\ [\Sigma \wedge \Sigma' \wedge (\bigwedge_{T \in \mathcal{T}} \forall \vec{y}\ T(\vec{y}) \leftrightarrow T'(\vec{y})) \wedge Q(\vec{x}) \rightarrow Q'(\vec{x})]$$

Or, rewriting,

$$\Sigma \wedge Q(\vec{x}) \models [(\bigwedge_{T \in \mathcal{T}} \forall \vec{y}\, T(\vec{y}) \leftrightarrow T'(\vec{y})\ ) \wedge \Sigma'] \rightarrow Q'(\vec{x})$$

This is the *entailment associated with determinacy.*

How Interpolation for the Entailment Gives a Reformulation. Going back to our general plan:

1. We have isolated a *semantic property* that an input query Q must have with respect to the target $\mathcal{T}$ and constraints Σ in order to have a relativized-quantifier first-order reformulation: it must be determined over $\mathcal{T}$ w.r.t.Σ.
2. We have expressed this property as an entailment:

$$\Sigma \wedge Q(\vec{x}) \models [(\bigwedge_{T \in \mathcal{T}} \forall \vec{y}\, T(\vec{y}) \leftrightarrow T'(\vec{y})) \wedge \Sigma'] \rightarrow Q'(\vec{x})$$

We now relate this to *interpolants*.

If $\lambda \models \rho$ is an entailment, an *interpolant* is a formula χ, such that:

- $\lambda \models \chi$ and $\chi \models \rho$
- Every relation in χ occurs in both λ and ρ.

Interpolation theorems state that any entailment for formulas in a logic L has an interpolant in logic L, possibly satisfying additional conditions.

To instantiate the meta-algorithm for this setting, we need to:

- show that an interpolant for this entailment gives the desired reformulation
- develop an algorithm to extract these interpolants

We start with the first item, the argument that an interpolant for the entailment above represents the first-order reformulation we want:

▶ **Proposition 9** (Variation of [13]). *Suppose Q is a first-order logic formula and Σ is a first-order sentence. Let χ be any interpolant for the entailment*

$$\Sigma \wedge Q \models [(\bigwedge_{T \in \mathcal{T}} \forall \vec{y}\, T(\vec{y}) \leftrightarrow T'(\vec{y})) \wedge \Sigma'] \rightarrow Q'$$

Then χ uses only the relations in $\mathcal{T}$, and is equivalent to Q for structures that satisfy Σ.

In particular, if χ is an interpolant in relative-quantifier first-order logic, we have found an RQFO reformulation of Q relative to Σ.

The proof of the proposition is quite straightforward: by the definition of an interpolant, χ uses only the relations common to the left and right, which are those in $\mathcal{T}$. And it is easy

to see that Σ entails that χ is equivalent to Q. For one direction we use that $\Sigma \wedge Q \models \chi$, by the definition of an interpolant. For the other part of the equivalence, using the fact that χ entails the right side, and setting the primed and unprimed relations equal we see that $\Sigma \wedge \chi \models Q$.

We have thus completed all the steps of the meta-algorithm, provided that we have a way to get RQFO interpolants from an entailment between RQFO formulas.

3.3 The Interpolation Theorem We Need: Relativized-Quantifier Interpolation

We discuss the modification of Craig interpolation needed to take a proof witnessing an entailment for relativized-quantifier first-order formulas and produce an RQFO interpolant. This will allow us to instantiate the meta-algorithm for RQFO constraints:

The interpolation result we need is a relativized-quantifier version of Craig interpolation, a variant of a result proven originally by Martin Otto.

▶ **Theorem 10** ([24]). *[Relativized-Quantifier Craig Interpolation Theorem] If λ and ρ are* RQFO *formulas such that $\lambda \models \rho$, then there is an interpolant χ in* RQFO*. Furthermore, if λ and ρ do not use equality, neither does χ.*

In [6], an effective version of this was proven, where we assume a proof of the entailment in a certain proof system. In [6] we use analytic tableaux as a proof system: this proof system and the interpolation algorithm for it are discussed in Chapter 2. But in the results we will just refer to a *suitable proof system*, leaving the details to the references.

▶ **Theorem 11** (Effective Relativized-Quantifier Craig Interpolation Theorem). *If λ and ρ are* RQFO *formulas* and *we have a suitable proof of $\lambda \models \rho$, then* in polynomial time in the proof *we can find an interpolant χ in* RQFO*. Furthermore, if λ and ρ do not use equality, neither does χ.*

Putting It All Together. We are ready to give the instantiation of the general methodology. Instantiating it to this setting, we see that to find an RQFO reformulation of Q with respect to subvocabulary $\mathcal{T}$ and constraints Σ we should:

- Search for a proof that the semantic property determinacy holds. That is, find a proof witnessing

$$\Sigma \wedge Q(\vec{x}) \models [(\bigwedge_{T \in \mathcal{T}} \forall \vec{y}\, T(\vec{y}) \leftrightarrow T'(\vec{y})) \wedge \Sigma'] \rightarrow Q'(\vec{x})$$

- Use the relativized-quantifier tableau-based interpolation algorithm to produce an interpolant $Q_{\mathcal{T}}(\vec{x})$.
- As shown in Proposition 9, such an interpolant will give the reformulation we want.

We also know that to have an RQFO reformulation, Q must be determined over $\mathcal{T}$ *w.r.t.* Σ, and thus there must be a proof of the entailment for determinacy. This means we have proven the following *effective reformulation theorem*:

▶ **Theorem 12.** *To find a reformulation of an* RQFO *query Q with respect to subvocabulary $\mathcal{T}$ and* RQFO *constraints Σ, it is sufficient to find a proof (in a suitable proof system) witnessing the entailment:*

$$\Sigma \wedge Q \models [(\bigwedge_{T \in \mathcal{T}} \forall \vec{y}\ T(\vec{y}) \leftrightarrow T'(\vec{y})) \wedge \Sigma'] \rightarrow Q'$$

where Σ' is formed from Σ by replacing each relation R with R', and Q' is formed similarly from Q.

From any such proof, we can effectively produce a reformulation.

Theorem 12 reduces searching for a reformulation to searching for a proof. Since the existence of a reformulation implies that the entailment above holds, and the entailment above captures determinacy:

▶ **Corollary 13** (Equivalence of Reformulations, Entailments, and Semantic Properties)**.** *An* RQFO *query Q has an* RQFO *reformulation with respect to subvocabulary $\mathcal{T}$ and constraints Σ if and only if Q is determined over subvocabulary $\mathcal{T}$ with respect to Σ if and only if the entailment in Theorem 12 holds.*

4 Vocabulary-Based Reformulation with Positive Existential Queries

Expanding on our general program outlined in the introduction, we show what happens when we restrict the target language for a rewriting. Recall that a *positive existential formula with inequalities* ($\exists^{+,\neq}$ formula) is a formula built up using only $\exists, \wedge, \vee$ from relational atoms and inequalities. We also consider the formula False to be positive existential with inequalities.

Given an RQFO formula Q, restricted vocabulary $\mathcal{T}$, and constraints Σ given by RQFO sentences, we are interested in getting a $\exists^{+,\neq}$ reformulation of Q over $\mathcal{T}$ with respect to Σ. This means we want a $\exists^{+,\neq}$ formula over $\mathcal{T}$ that agrees with Q for instances satisfying the constraints.

The Semantic Property for $\exists^{+,\neq}$ Reformulation. Following the "meta-algorithm" from the introduction, we start by finding the appropriate semantic property that an input Q must have to admit a $\exists^{+,\neq}$ reformulation. [23] isolated such a property, which we call monotonic determinacy[1]. We say that a query Q is *monotonically-determined over* $\mathcal{T}$ relative to Σ if:

> for any two instances $\mathsf{I}_1, \mathsf{I}_2$ that satisfy Σ and such that for all relations $T \in \mathcal{T}$, $[\![T]\!](\mathsf{I}_1) \subseteq [\![T]\!](\mathsf{I}_2)$, then $[\![Q]\!](\mathsf{I}_1) \subseteq [\![Q]\!](\mathsf{I}_2)$.

[1] [23] used the term "monotone"

The Entailment Corresponding to the Semantic Property. Proceeding to the second step of the meta-algorithm, we will express this semantic property as an entailment. Let Σ' be a copy of the constraints Σ where each occurrence of a relation R in $\mathcal{S}$ has been replaced by a copy R'.

Monotonic determinacy of a first-order query Q over $\mathcal{T}$ relative to Σ can be restated as saying that the following sentence holds on all instances:

$$\forall \vec{x}\ [\Sigma \wedge \Sigma' \wedge (\bigwedge_{T \in \mathcal{T}} \forall \vec{y}\ T(\vec{y}) \rightarrow T'(\vec{y})) \wedge Q(\vec{x})] \rightarrow Q'(\vec{x})$$

Above, Q' and Σ' are again the result of changing unprimed relations R to their primed counterparts R' within Q and within Σ respectively. Observe that if a $\exists^{+,\neq}$ formula over $\mathcal{T}$ is true on an instance I, then it is true on any instance I' which only adds tuples to the relations in $\mathcal{T}$. It is thus easy to see that a sufficient condition for monotonic determinacy of Q over $\mathcal{T}$ w.r.t.Σ is that Σ implies the sentence $\forall \vec{x}\ Q(\vec{x}) \leftrightarrow \varphi(\vec{x})$, where φ is a $\exists^{+,\neq}$ formula mentioning only relations in $\mathcal{T}$.

We highlight the difference from the entailment for determinacy: here we only have implication in the "forward" direction, from unprimed to primed, while for determinacy we have implications in both directions.

Generating $\exists^{+,\neq}$ Reformulations from Proofs of the Entailment. We will show that by applying interpolation to proofs of the entailment for monotonic determinacy, we get $\exists^{+,\neq}$ reformulations. In doing so, we will prove the following analog of Theorem 10:

▶ **Theorem 14.** *If the constraints Σ are in* RQFO *and* RQFO *query Q is monotonically-determined over $\mathcal{T}$, then there is a $\exists^{+,\neq}$ formula $\varphi(\vec{x})$ using only relations in $\mathcal{T}$ such that $\Sigma \models \forall \vec{x}\ [Q(\vec{x}) \leftrightarrow \varphi(\vec{x})]$. Furthermore, if the constraints and the query Q do not make use of equality, then χ can be taken to be positive existential (without inequalities).*

We refer to this as the *Projective Monotone Preservation Theorem*. The adjective "Projective" emphasizes that we are dealing with a subset of the signature, as opposed to many preservation theorems one encounters in logic textbooks and papers, which deal with syntactically characterizing a semantic property involving the entire signature.

The steps in proving Theorem 14 will follow the meta-algorithm. We will prove a modification of the Craig Interpolation Theorem, the Relativized-Quantifier Lyndon Interpolation Theorem, and our proof will provide an algorithm for generating interpolants of a special form from an entailment. We then show that the interpolants produced by this algorithm, when applied to a re-arrangement of the entailment corresponding to the semantic property of monotonic determinacy, will give us the $\exists^{+,\neq}$ reformulation of our query Q with respect to the constraints.

The Interpolants Required for $\exists^{+,\neq}$ Reformulation. Before giving the argument, we provide some motivation. We can restate the entailment for monotonic determinacy as:

$$\Sigma \wedge (\bigwedge_{T \in \mathcal{T}} \forall \vec{y}\, T(\vec{y}) \rightarrow T'(\vec{y})) \wedge Q(\vec{x}) \models (\Sigma' \rightarrow Q'(\vec{x}))$$

As before, Σ' is a copy of the constraints on primed versions of the relations, and Q' is a copy of the query Q on primed versions of the relations. The common vocabulary on the left and right consists of exactly the relations T' for $T \in \mathcal{T}$.

Further, we note that these common relations only occur on the right of an implication on the left-hand side. Writing out the implication $A \rightarrow B$ as $\neg A \vee B$, we see that the common relations would not occur within a negation on the left side. Informally, we can say that these relations "occur positively" on the left hand side in the original formula. Thus we want to show that the interpolant will also contain these common relations positively. To do this, we need a formal definition of "occurring positively" that applies to arbitrary first-order formulas, and a version of interpolation that connects the relations occurring positively in the interpolant with those that occur positively in both sides of the entailment.

Formally, we say that a relation *occurs positively* in a first-order formula if it occurs in the scope of an even number of negations, when we rewrite the formula to use only the quantifiers and the connectives $\wedge, \vee, \neg$. A relation occurs negatively in a formula if it occurs in the scope of an odd number of negations. Intuitively if a relation occurs only positively, then the set of solutions to the formula can only increase or stay the same as tuples are added to that relation (this is easy to check if the relation does not appear under any negations at all).

The interpolation theorem that tracks which occurrences are positive is the following result, which is a strengthening of the Relativized-quantifier Craig Interpolation Theorem, Theorem 10:

▶ **Theorem 15** (Relativized-quantifier Lyndon Interpolation Theorem)**.** *Suppose λ and ρ are in* RQFO *and $\lambda \models \rho$. Then there is an* RQFO *interpolant χ for the entailment such that a relation occurs positively in χ only if it occurs positively in both λ and ρ, and a relation occurs negatively in χ only if it occurs negatively in both λ and ρ. Furthermore if equality does not occur in λ or ρ, then it does not occur in χ, so in particular χ can not contain inequalities.*

Again, there is a well-known version of this for the classical semantics of first-order logic, Lyndon's interpolation theorem [22].

The construction that witnesses the Relativized-quantifier Lyndon Interpolation Theorem, Theorem 15, is a variant of the one for the Relativized-quantifier CIT, which is discussed Chapter 2.

Getting $\exists^{+,\neq}$ and Positive Existential Reformulations via Interpolants. Theorem 15 shows that we can extract a certain kind of interpolant from a tableau proof witnessing an entailment corresponding to monotonic determinacy. We are now ready to instantiate the last step of

our meta-algorithm, extracting a reformulation from an interpolant. This will complete the proof of Theorem 14.

Apply Theorem 15 to the entailment

$$\Sigma \wedge (\bigwedge_{T \in \mathcal{T}} \forall \vec{y}\, T(\vec{y}) \rightarrow T'(\vec{y})) \wedge Q(\vec{x}) \models (\Sigma' \rightarrow Q'(\vec{x}))$$

We can conclude that there is an RQFO interpolant γ mentioning only relations in the primed copy of $\mathcal{T}$, where these relations only occur positively. We can assume γ is built up from True and False via connectives and relativized quantifiers $\forall \vec{x}\ R(\vec{x}) \rightarrow \varphi$ and $\exists \vec{x}\ R(\vec{x}) \wedge \varphi$. We claim that any RQFO formula in which all relations occur positively must be equivalent to a positive existential formula. To see this, convert an RQFO formula to NNF. If the resulting formula γ had any relativized universal quantifier, then consider an outermost quantification of the form $\forall \vec{x}\ (R(\vec{x}) \rightarrow \varphi)$. R must occur negatively in this formula. But then it must occur negatively within γ, since existential quantification and the positive Boolean operators preserve the polarity of a subformula, and this contradicts the assumption on γ. Hence the Negation Normal Form of γ can not contain any relativized universal quantifier, and thus must be $\exists^{+,\neq}$.

We say $\exists^{+,\neq}$ above, rather than positive existential, because the argument applies only to relations R of the schema, not equality. If the equality symbol does not appear in the entailment we know that it is not generated in the interpolant. Therefore when equality does not occur in the constraints or the query, we can strengthen the conclusion to be that the interpolant is positive existential.

This completes the proof of the Projective Monotone Preservation Theorem, Theorem 14.

Application to View-Based Query Reformulation. Let us look at the setting where there are no constraints other than those that come from views defined by conjunctive queries. As a corollary of Theorem 14 we have:

▶ **Corollary 16.** *Suppose Q is a CQ and $V_1 \dots V_n$ are views defined by arbitrary equality-free* RQFO *formulas. Then, Q is monotonically-determined in $V_1 \dots V_n$ if and only if there is a positive existential reformulation of Q in terms of $V_1 \dots V_n$.*

5 Vocabulary-Based Existential Reformulation

At this point, we have proven a statement about first-order reformulations and one about positive existential reformulations. What about queries that can be reformulated using *existential formulas*? That is, formulas that are built up from atoms and negated atoms by positive Boolean operators and existential quantification. There are conjunctive queries that are equivalent to existential formulas but not to positive existential ones. For example, in the absence of any constraints $\exists x\ S(x) \wedge \neg R(x)$ is not equivalent to a positive existential formula. Can we use a similar technique to detect which formulas are equivalent to an existential

formula with respect to a set of constraints, and if so find such a reformulation? We give a positive answer to this below, restricting for simplicity to equality-free RQFO.

The Semantic Property for Existential Reformulation. As before, let Σ be a set of integrity constraints in equality-free RQFO, and $\mathcal{T}$ a subset of the relations of Sch. We start by isolating a property that Q must have in order to possess an existential reformulation.

We say that a query Q is *induced-subinstance-monotonically-determined over* $\mathcal{T}$ relative to Σ if:

Whenever we have two instances $\mathsf{I}_1, \mathsf{I}_2$ that satisfy Σ and such that I_1 is an induced subinstance of I_2 then $[\![Q]\!](\mathsf{I}_1) \subseteq [\![Q]\!](\mathsf{I}_2)$.

An instance I_1 is an induced subinstance of I_2 means that I_2 contains all facts of I_1 and I_2 does not add any facts over the active domain of I_1.

Note that if an existential formula over $\mathcal{T}$ is true on an instance I, then it is true on any instance I' which only adds tuples to the relations in $\mathcal{T}$ and never "destroys a negated assertion about a relation of $\mathcal{T}$ holding in I". From this we see that if a formula is equivalent to an existential formula under a set of constraints Σ, then the formula is induced-subinstance-monotonically-determined over $\mathcal{T}$ *w.r.t.* Σ.

The Entailment Corresponding to the Semantic Property. As in the previous cases, we instantiate our meta-algorithm by writing out the semantic property as an entailment.

Let $\mathsf{InDomain}_{\mathcal{T}}(x)$ abbreviate the formula:

$$\bigvee_{T \in \mathcal{T}} \bigvee_{j} \exists w_1 \ldots \exists w_{j-1}\ \exists w_{j+1} \ldots w_{\mathsf{arity}(T)}\ T(w_1, \ldots w_{j-1}, x, w_{j+1}, \ldots, w_{\mathsf{arity}(T)})$$

So $\mathsf{InDomain}_{\mathcal{T}}$ states that x is in the domain of a relation in $\mathcal{T}$. The entailment we need is:

$$Q(\vec{x}) \wedge \Sigma \wedge \Sigma' \wedge \bigwedge_{T \in \mathcal{T}} (\forall \vec{y}\ T(\vec{y}) \rightarrow T'(\vec{y})) \wedge \bigwedge_{T \in \mathcal{T}} (\forall \vec{y}\ \bigwedge_{i} \mathsf{InDomain}_{\mathcal{T}}(y_i) \wedge T'(\vec{y}) \rightarrow T(\vec{y})\ ) \models Q'(\vec{x})$$

Comparing with the two previous entailments, we have the forward implication as before, and a restriction of the backward implication. It is clear that induced-subinstance-monotonicity is equivalent to this entailment holding.

Extracting Interpolants from a Proof of the Entailment: Statement of Result. We will show later that if we have a "suitable" interpolant for the entailment corresponding to induced-subinstance-monotonicity, then we can extract an existential reformulation from it, completing another instantiation of the meta-algorithm.

This will give us a proof of another analog of Theorem 10:

▶ **Theorem 17.** *If the constraints Σ are in equality-free* RQFO*, and* RQFO *query Q is induced-subinstance-monotonically-determined over $\mathcal{T}$, then there is an existential first-order formula $\varphi(\vec{x})$ using only relations in $\mathcal{T}$ such that $\Sigma \models \forall \vec{x}\ Q(\vec{x}) \leftrightarrow \varphi(\vec{x})$.*

A similar theorem will hold if the constraints are in RQFO with equality, with the conclusion being that φ is an existential formula with inequalities. There is also an effective variant of the theorem above, an analog of Theorem 12. It states that the existential φ can be generated efficiently from a suitable proof of the entailment for induced-subinstance-monotonicity.

We defer the proof of Theorem 17 for the moment. It will follow from a more general theorem, Theorem 27, proved later. The reformulation we need will come from applying an interpolation procedure to the entailment above. But we need a new interpolation result to guarantee that the interpolant will be existential, and this extended interpolation theorem is "Access Interpolation", described later.

Łos-Tarski. The theorem above is closely related to another result in logic, the Łos-Tarski preservation theorem. This states that a formula is preserved under extensions exactly when it is equivalent to an existential formula. Roughly speaking, the Łos-Tarski theorem is a special case of Theorem 17, where Σ is empty and $\mathcal{T}$ contains all relations in the schema. The "roughly speaking" disclaimer is because the Łos-Tarski theorem deals with classical first-order logic formulas rather than for RQFO, and the notion of induced subinstance must be replaced by the analogous notion for structures.

6 Rewriting with Access Patterns

Thus far the target of rewriting was specified through vocabulary restrictions. We wanted a query that used a fixed set of target relations, perhaps restricted to be positive existential or existential. We now consider a finer notion of reformulation, where the target has to satisfy *access restrictions*.

We begin with the definitions of access methods along with a programming language, the RA-plans, that combines data access by means of a fixed set of access methods with data manipulation using relational algebra queries. Recall from Section 2 that relational algebra represents an algebraic programming language that is equivalent in expressiveness to first-order logic formulas that are *safe*, in that for every finite instance, the number of satisfying assignments is finite. Implementation of logic-based languages proceeds by translating a logical formula into relational algebra. RA-plans can be thought of as a variation of relational algebra in which we want to abide by a fixed set of data interfaces given by access methods. They will thus be the new target of reformulation.

We return to our program of going from semantic properties to rewritings, following the methodology outlined in the introduction. We present semantic properties that must hold for a query to be implemented using the interface given by a set of access methods, and show that these properties can be captured by entailments. In the remainder of the section we

explain how proofs of these entailments can be converted into plans within our plan languages. The conversion from proof to plan will again proceed via interpolation. But in this case we need a new interpolation theorem tailored to the setting of access methods.

6.1 Basics of Target Restrictions Based on Access Methods

We now look at a notion of interface that is closer to the traditional notion in programming languages: a set of functions that access the data. A specification of this interface will be an extended set of metadata describing both the format of the data (e.g. the vocabulary that would be used in queries and constraints) and the access methods (functions that interact with the stored data).

An *access schema* consists of:

- A collection of relations, each of a given arity.
- A finite collection C of schema constants ("Smith", 3, ...). Schema constants represent a fixed set of values that will be known to a user prior to interacting with the data. Values that can be used in queries and constraints should be schema constants, as before. In addition, any fixed values that might be used in plans that implement queries should come from the set of schema constants. For example, a plan to answer a query about the mathematics department might involve first putting the string "mathematics" into a university directory service.
- For each relation R, a collection (possibly empty) of *access methods*[2]. Recall that a *position* of a relation R is a number between 1 and $\mathsf{arity}(R)$. Each access method mt is associated with a collection (possibly empty) of positions of R — the *input positions* of mt.
- Integrity constraints, which are sentences in relativized-quantifier first-order logic as before.

An *access* (relative to a schema as above) consists of an access method of the schema and a *method binding* — a function assigning values to every input position of the method. If mt is an access method on relation R with arity n, I is an instance for a schema that includes R, and AccBind is a method binding on mt, then the *output* or *result* of the access (mt, AccBind) on I is the set of n-tuples $\vec{t} \in [\![R]\!](I)$ such that $\vec{t}$ restricted to the input positions of mt is equal to AccBind.

An access method may have an empty collection of input positions. In this case, the only access that can be performed using the method is with the empty method binding. When a method has no input positions, we say that the access method is "input-free". In Example 2 of Section 1, the Udirectory table was assumed to have such an input-free access method.

[2] Our definition of "access methods" is a variant of the terminology "access patterns" or "binding patterns" found in the database literature.

The goal is to reformulate source queries in a target language that represents the kind of restricted computation done over an interface given by an access schema. We first formalize this operationally, as a language of *plans*. Plans are straight-line programs that can perform accesses and manipulate the results of accesses using relational algebra expressions. This language could model, at a high-level, the plans used internally in a database management system. It could also describe the computation done within a data integration system, which might access remote data via a web form or web service and then combine data from different sources using SQL within its own database management system.

All of our plan languages have as a primitive an *access command*. Over a schema Sch with access methods, an access command is of the form:

$$T \Leftarrow_{\mathsf{OutMap}} \mathsf{mt} \Leftarrow_{\mathsf{InMap}} E$$

where:

- E is a relational algebra expression, the *input expression*, over some set of relations not in Sch (henceforward "temporary tables");
- mt is a method from Sch on some relation R;
- InMap, the *input mapping* of the command, is a function from the output attributes of E onto the input positions of mt;
- T, the *output table* of the command, is a temporary table;
- OutMap, the *output mapping* of the command, is a bijection from positions of R to attributes of T.

Note that an access command using an input-free method must take the empty relation algebra expression $\emptyset$ as input.

The manipulation of data retrieved by an access is modeled with the other primitive of our languages, a *middleware query command*. These are of the form $T := Q$, where Q is a relational algebra expression over temporary tables and T is a temporary table. We use the qualifier "middleware" to emphasize that the queries are performed on temporary relations created by other commands, rather than on relations of the input schema.

A *relational algebra-plan* (or simply, *RA-plan*) consists of a sequence of access and middleware query commands, ending with at most one *return command* of the form Return E, where E is an RA expression.

▶ **Example 18.** We return to Example 2 of Section 1, where we had two sources of information. One was Profinfo, which was available through an access method $\mathsf{mt}_{\mathsf{Profinfo}}$ requiring input on the first position. The second was Udirectory, which had an access method $\mathsf{mt}_{\mathsf{Udirectory}}$ requiring no input. Our query Q asked for ids of faculty named "Smith". One plan that is

equivalent to Q would be represented as follows

$T_1 \Leftarrow \mathsf{mt}_{\mathsf{Udirectory}} \Leftarrow \emptyset$
$T_2 := \pi_{\mathsf{eid}}(\sigma_{\mathsf{lname}=\text{“}\mathsf{Smith}\text{”}} T_1)$
$T_3 \Leftarrow \mathsf{mt}_{\mathsf{Profinfo}} \Leftarrow T_2$
Return $\pi_{\mathsf{eid}}(T_3)$

Above we have omitted the mappings in writing access commands, since they can be inferred from the context. We will often do this in plans for brevity. ⌟

Semantics of Plans. A temporary table is *assigned* in a plan if it occurs on the left side of a command, and otherwise is said to be *free*. The semantics of plans is defined as a function that takes as input an instance I for Sch and interpretations of the free tables. If the plan has no Return statement, the output consists of interpretations for each assigned temporary table. If the plan contains a statement Return E, the output is an interpretation of a relation with attributes for each output attribute of E. In the latter case, we refer to this as the *output* of the plan.

An access command $T \Leftarrow_{\mathsf{OutMap}} \mathsf{mt} \Leftarrow_{\mathsf{InMap}} E$ is executed by evaluating the expression E on I and "accessing mt on every result tuple". That is, each output tuple of E is mapped to a tuple $t_{j_1} \dots t_{j_m}$ using the input mapping InMap. For each tuple $\vec{t} = t_1 \dots t_n \in R$ that "matches" (i.e.that extends) $t_{j_1} \dots t_{j_m}$, $\vec{t}$ is transformed to a tuple $\vec{t'}$ using the output mapping OutMap. The interpretation of T is then the union of all such tuples $\vec{t'}$. A middleware query command $T := E$ executes query E on the contents of the temporary tables mentioned in E, and then assigns the result to temporary table T.

A plan is executed by evaluating its commands in sequence, with each command operating on the instance formed from the input instance, adding the interpretations of assigned tables produced by earlier commands. For a plan having as its final command Return E, the output of the plan is the evaluation of E on the instance formed as above.

We usually assume (without loss of generality) that each table is only assigned once within a plan. Given plan PL, temporary table T that occurs in PL, and an instance I for the schema Sch, we let $[\![T|\ \mathsf{PL}]\!]_I$ be the content of T when PL is run on I. For plan PL including a Return statement, we let $[\![\mathsf{PL}]\!](I)$ be the output of PL on I. Similarly, given a relational algebra expression E over temporary tables $T_1 \dots T_n$ of PL and instance I, $[\![E|\ \mathsf{PL}]\!]_I$ represents the result of E when run on $[\![T_1|\ \mathsf{PL}]\!]_I \dots [\![T_n|\ \mathsf{PL}]\!]_I$.

Fragments of the Plan Language. We now define fragments of our plan language, analogs of the fragments of relational algebra and first-order logic. In RA-plans, we allowed arbitrary relational algebra expressions in both the inputs to access commands and the middleware query commands. We can similarly talk about $SPJ^{\neq}$*-plans*, where the expressions in access and middleware query commands are built up from the $SPJ^{\neq}$ relational algebra operators

and $USPJ^{\neq}$*-plans* that allow the union operator of relational algebra in addition to $SPJ^{\neq}$ operators. We define $USPJAD^{\neq}$*-plans* as RA-plans in which relational algebra's difference operator only occurs in a *non-membership check*, which tests whether the tuples in a projection of a temporary table are not in a given relation R. Formally, a non-membership check is a sequence of two commands:

$$T' \Leftarrow_{\mathsf{OutMap}} \mathsf{mt} \Leftarrow_{\mathsf{InMap}} \pi_{a_{j_1} \ldots a_{j_m}}(T)$$
$$T'' := T - (T \bowtie T')$$

where in the first command:

- mt is an access method on some relation R with input positions $j_1 \ldots j_m$;
- the input mapping InMap maps attribute a_{j_i} to position j_i;
- the attributes of the output table T' are a subset of the attributes of T containing each a_{j_i};
- the output mapping OutMap maps j_i back to a_{j_i}.

In the second command, the join condition identifies attributes that have the same name. SPJ-plans, $USPJ$-plans, and $USPJAD$-plans are defined analogously to the classes above, but not allowing inequality conditions in selections or joins.

Plans that Answer Queries. We now define what it means for a plan to correctly implement a query. Given an access schema Sch, a plan *answers* a query Q *(over all instances)* if for every instance I satisfying the constraints of Sch, the output of the plan on I is the same as the output of Q. We often omit the schema from our notation, since it is usually clear from context, saying that a plan PL answers Q.

From Plans to Specialized Formulas. We have stated our target language for reformulation as a procedural language. But to make use of interpolation-based methods, it will be useful to have a logic, consisting of formulas. We will be able to state an equivalence of plans in the case of sentences — that is, Boolean queries. In this case, we can show that a plan using a given set of access patterns is equivalent to a logical sentence where the quantification structure is compatible with the access patterns. This approach can be bootstrapped to general logical formulas, but we defer the discussion here, see [6].

An RQFO sentence is *executable* (relative to an access schema Sch) if it is built up from equalities and the formula True using arbitrary Boolean operations and the quantifiers:

$$\forall \vec{y}\ [R(\vec{x}, \vec{y}) \rightarrow \varphi(\vec{x}, \vec{y}, \vec{z})]$$
$$\exists \vec{y}\ R(\vec{x}, \vec{y}) \wedge \varphi(\vec{x}, \vec{y}, \vec{z})$$

and for any such quantification above, if R is a Sch relation, then R has an access method mt such that all of the input positions of mt are occupied by some x_i (that is: by a free variable or constant).

This definition captures the informal idea that the sentence is compatible with the access methods: we should be quantifying only over the output positions, while the values input positions should be provided.

The next proposition says that executable RQFO sentences we can find a plan that filters an input table down to the subset satisfying the formula.

▶ **Proposition 19.** *There is a linear time procedure that takes as input an executable* RQFO *sentence φ and produces an equivalent Boolean RA-plan* Plan_φ.

Furthermore, if the RQFO *sentence is existential with inequalities (resp. existential) the result is a $USPJAD^{\neq}$-plan (resp. $USPJAD$-plan). If the sentence is positive existential with inequalities (resp. positive existential) the result is a $USPJ^{\neq}$-plan (resp. $USPJ$-plan).*

In the other direction, we can convert every Boolean plan (of the given form) into the corresponding logic.

This result is not difficult, and the inductive arguments are given in [6]. It will allow us to deal with logical formulas having these "access-restricted quantifiers" from now on.

The Accessible Part. In order to begin our instantiation of the meta-algorithm for plans, we need a semantic property corresponding to existence of a plan. The property should say that formula "only depends on the accessible data", and thus we need to formalize what accessible data means.

▶ **Definition 20.** *Given an instance* I *for schema* Sch *the* accessible part of I, *denoted* $\mathsf{AccPart}(\mathsf{I})$, *and the* accessible values of I, *denoted* $\mathsf{accessible}(\mathsf{I})$. *Informally the accessible part consists of all the facts over* I *that can be obtained by starting with empty relations and iteratively entering values into the access methods. If* Sch *contains no schema constants, this will be an instance containing a set of facts* $\mathsf{Accessed}R(v_1 \dots v_n)$, *where R is a relation and $v_1 \dots v_n$ are values in the domain of* I *such that $R(v_1 \dots v_n)$ holds in* I, *obtained by starting with relations* $\mathsf{Accessed}R_0$ *and* $\mathsf{accessible}_0$ *empty and then iterating the following process until a fixpoint is reached:*

$$\mathsf{accessible}_{i+1} = \mathsf{accessible}_i \cup \bigcup_{\substack{R \text{ a relation} \\ j \leq \mathsf{arity}(R)}} \pi_j(\mathsf{Accessed}R_i)$$

and

$$\mathsf{Accessed}R_{i+1} = \mathsf{Accessed}R_i \cup \bigcup_{\substack{(R,\{j_1,\dots,j_m\}) \\ \text{there is a method of } \mathsf{Sch} \text{ on } R \text{ with inputs } j_1,\dots,j_m}} \{(v_1 \dots v_n) \in [\![R]\!](I) \mid v_{j_1} \dots v_{j_m} \in \mathsf{accessible}_i\}$$

Above $\pi_j(\mathsf{Accessed}R_i)$ denotes projection of $\mathsf{Accessed}R_i$ on the j^{th} position. For a finite instance, this induction will reach a fixpoint after $|\mathsf{I}|$ iterations, where $|\mathsf{I}|$ denotes the number

of facts in I. *For an arbitrary instance the union of these instances over all* i *will be a fixpoint. Assuming* Sch *does include schema constants, we modify the definition by starting with an* $\mathsf{accessible}_0$ *consisting of the schema constants, rather than being empty.*

Above we consider $\mathsf{AccPart}(I)$ as a database instance for the schema with relations $\mathsf{accessible}$ and $\mathsf{Accessed}R$. The accessible values are the union of the $\mathsf{accessible}_i$.

In the case of vocabulary-based access-restrictions, the accessible part of an instance just represents the restriction of the instance to the relations in the subsignature $\mathcal{T}$. Thus access determinacy of a query Q in the case of vocabulary-based restrictions is the same as determinacy of Q with respect to the subsignature.

The Semantic Property and Entailment for RA-Plans. We now give the analogous property and entailment for RA-plans. Q is said to be *access-determined* over Sch if for all instances I and I' satisfying the constraints of Sch with $\mathsf{AccPart}(I) = \mathsf{AccPart}(I')$ we have $[\![Q]\!](I) = [\![Q]\!](I')$. If a query is *not* access-determined, it is obvious that it cannot be answered through any plan, since it is easy to see that any plan can only read tuples in the accessible part.

▶ **Example 21.** We return to the setting of Example 2, where we have a $\mathsf{Profinfo}$ table available via a web form, containing information about faculty, including their last names, office number, and employee id, but with only an access method $\mathsf{mt}_{\mathsf{Profinfo}}$ that requires giving an employee id as an input. We consider a query Q asking for ids of faculty named "Smith", where "Smith" is a schema constant.

We show that Q is not access-determined. For this, take I to be any instance that contains exactly one tuple, with $\mathsf{lastname}$ "Smith", but with no schema constant as its employee id. Let I' be the empty instance. The accessible parts of I and I' are empty, since in both cases when we enter all the constants we know about in $\mathsf{mt}_{\mathsf{Profinfo}}$, we get the empty response. But Q has an output on I but no output on I'.

I and I' witness that Q is not access-determined. From this we see that Q can not be implemented by any plan using $\mathsf{mt}_{\mathsf{Profinfo}}$. ⌟

We now turn to the corresponding entailment for access-determinacy. The accessible part itself is not a first-order definable object — it requires an infinite disjunction or recursion. But it will turn out that "agreeing on the accessible part" can be expressed in a first-order way.

We start with a schema Sch consisting of relation symbols and RQFO sentences — integrity constraints, as well as access restrictions. The *bidirectional accessible schema for* Sch, denoted $\mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch})$, is a vocabulary without access restrictions, and also a set of integrity constraints in the form of RQFO sentences. It is defined as follows:

- The constants are those of Sch.
- The relations are those of Sch, a unary relation $\mathsf{accessible}(x)$ ("x is an accessible value") plus a copy of each relation R of Sch called $\mathsf{InfAcc}R$ (the "inferred accessible version of R").

- The constraints are those of Sch (referred to as "Sch constraints" below) for each access method mt on relation R of arity n with input positions $j_1 \dots j_m$ along with the following constraints (dropping universal quantifiers on the outside for brevity)
  - *forward accessibility axioms*: we have a rule:

$$\mathsf{accessible}(x_{j_1}) \wedge \ldots \wedge \mathsf{accessible}(x_{j_m}) \wedge R(x_1 \dots x_n) \rightarrow \mathsf{InfAcc}R(x_1 \dots x_n) \wedge \bigwedge_j \mathsf{accessible}(x_j)$$

  - *backward accessibility axioms*

$$\bigwedge_{i \leq m} \mathsf{accessible}(x_{j_i}) \wedge \mathsf{InfAcc}R(x_1 \dots x_n) \rightarrow R(x_1 \dots x_n) \wedge \bigwedge_i \mathsf{accessible}(x_i)$$

    In addition, we have $\mathsf{accessible}(c)$ for each constant c of Sch.
- A copy of each of the original integrity constraints, with each relation R replaced by $\mathsf{InfAcc}R$, denoted "$\mathsf{InfAccCopy}$ constraints" below.

Given a query Q, its *inferred accessible version* $\mathsf{InfAcc}Q$ is obtained by replacing each relation R by $\mathsf{InfAcc}R$. Informally, $\mathsf{InfAcc}Q$ represents the fact that the existence of a witness to Q can be obtained through making accesses and reasoning.

We overload $\mathsf{AcSch}(\mathsf{Sch})$ to also refer to the conjunction of axioms in this schema $\mathsf{AcSch}(\mathsf{Sch})$.

A way of motivating the axioms is to think informally of the relations R $\mathsf{InfAcc}R$ as representing the instances considered in access determinacy. By having two copies of the constraints, we are stating that both instances satisfy the constraints of the schema. The forward and backward accessibility axioms will ensure that these two instances have the same accessible part. The entailment thus captures that if two instances have the same accessible part, and both satisfy the constrains, and one satisfies query Q, then so does the other.

Again, we show that this entailment captures the proposed preservation property, access-determinacy.

▷ Claim 22. The following are equivalent (for any Boolean RQFO query Q and access schema consisting of RQFO constraints):

1. Q entails $\mathsf{InfAcc}Q$ with respect to the rules in $\mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch})$
2. Q is access-determined over Sch

Proof. We prove that the first item implies the second. Fix I and I' satisfying the schema with the same accessible part, and assume I satisfies Q. Consider the instance I'' for $\mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch})$ formed by interpreting the relations R as in I, the relation $\mathsf{accessible}$ by the accessible values of I, and each $\mathsf{InfAcc}R$ by the interpretation of R in I'. Then one can verify that I'' satisfies the constraints of $\mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch})$. Since I (and hence I'') satisfies Q, and we are assuming that Q entails $\mathsf{InfAcc}Q$ with respect to $\mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch})$ we can conclude that I'' must satisfy $\mathsf{InfAcc}Q$. So Q holds in I' as required.

We complete the proof of the claim by arguing from the second item to the first. Suppose Q is not contained in $\mathsf{InfAcc}Q$ with respect to the rules in $\mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch})$. Hence there is an instance $I^{\mathsf{AcSch}^{\leftrightarrow}}$ satisfying the rules of $\mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch})$ and also satisfying $Q \wedge \neg \mathsf{InfAcc}Q$. Let I_1 consist of the restriction of $I^{\mathsf{AcSch}^{\leftrightarrow}}$ to the original schema relations. Let I_2 consist of the inferred accessible relations from $I^{\mathsf{AcSch}^{\leftrightarrow}}$, renamed to the original schema.

We claim that a fact $R(e_1 \dots e_n)$ in the accessible part of I_1 is in the accessible part of I_2. This is proven by induction on the point in which the fact goes into the accessible part, using the forward accessibility axioms. Arguing symmetrically (now using the backward axioms), we find that I_1 and I_2 have the same accessible part, and hence they contradict access-determinacy. ◀

We are now ready to state our main results on the relationship between semantic properties, entailments, and plans.

▶ **Theorem 23.** *For any Boolean* RQFO *query Q and access schema* Sch *with constraints specified in* RQFO*, there is an RA-plan answering Q (over instances satisfying* Sch*) if and only if $Q \wedge \mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch}) \models \mathsf{InfAcc}Q$. If the query and constraints do not include equality, then the RA-plan will not make use of equality in any of its RA expressions.*

Further, from any tableau proof witnessing $Q \wedge \mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch}) \models \mathsf{InfAcc}Q$ we can extract (in linear time) an RA-plan for Q over Sch*.*

Using Claim 22, we can restate Theorem 23:

> For any Boolean RQFO query Q and access schema Sch with constraints specified in RQFO, there is an RA-plan answering Q (over instances of Sch) if and only if Q entails $\mathsf{InfAcc}Q$ with respect to the rules in $\mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch})$ if and only if Q is access-determined.

In the direction from right to left we are again going from a preservation property to a syntactic restriction. Thus Theorem 23 can be considered as an "access-restricted variant" of the definability theorems given in the vocabulary-based case.

6.2 The Interpolation Result Behind the Rewriting Results: Access Interpolation

We have instantiated the first steps of our meta-algorithm for reformulation via relational algebra access plans. For RA-plans, we have stated a semantic property of a query that is required for it to have an equivalent target RA plan. And we have shown that the semantic property is equivalent to an entailment. The next ingredient in proving the main results relating plans to proofs of these entailments is an interpolation theorem that tracks the "access methods used in the interpolant".

Interpolation and Access Methods. Recall Proposition 19. It states that Boolean RA-plans have the same expressiveness as RQFO sentences that are executable for membership checks. For simplicity, we call these "executable FO Boolean queries" below.

By this corollary, we can prove our main results by finding reformulations that are executable FO Boolean queries, rather than RA-plans. To find these reformulations, we require a version of Craig interpolation that allows us to relate the "binding patterns" — i.e. the subset of variables that are quantified in each relativized-quantifier quantification — used in the interpolant χ of the entailment $\lambda \models \rho$ with those used in λ or ρ. When we apply this theorem to the entailment of $\mathsf{InfAcc}Q$ by Q, we can conclude that the interpolant is an executable FO Boolean query.

We associate to RQFO formulas the set of binding patterns used in quantification, where a binding pattern is a relation and a subset of the positions. This is done by structural induction on the formula, assuming that the formula is first put into negation normal form.

$$
\begin{array}{lcl}
\mathsf{BindPatt}(\top) \;=\; \mathsf{BindPatt}(x = y) & = & \emptyset \\
\mathsf{BindPatt}(R(t_1, \ldots, t_n)) & = & \{(R, \{1, \ldots, n\})\} \\
\mathsf{BindPatt}(\neg\varphi) & = & \mathsf{BindPatt}(\varphi) \\
\mathsf{BindPatt}(\varphi \wedge \psi) & = & \mathsf{BindPatt}(\varphi) \cup \mathsf{BindPatt}(\psi) \\
\mathsf{BindPatt}(\varphi \vee \psi) & = & \mathsf{BindPatt}(\varphi) \cup \mathsf{BindPatt}(\psi) \\
\mathsf{BindPatt}(\exists \vec{x}\ (R(t_1, \ldots, t_n) \wedge \varphi)) & = & \mathsf{BindPatt}(\varphi) \cup \{(R, \{i \mid t_i \notin \vec{x}\})\} \\
\mathsf{BindPatt}(\forall \vec{x}\ (R(t_1, \ldots, t_n) \rightarrow \varphi)) & = & \mathsf{BindPatt}(\varphi) \cup \{(R, \{i \mid t_i \notin \vec{x}\})\}
\end{array}
$$

For example,

$$\mathsf{BindPatt}(\exists x\ \exists y\ (R(x,y) \wedge \forall z\ (S(x,y,z) \rightarrow U(x,y,z)))) = \{(R, \emptyset), (S, \{1,2\}), (U, \{1,2,3\})\}$$

We can similarly talk about the *universal binding patterns* of a formula — those that arise from a subformula $\forall \vec{x}\ (R(t_1, \ldots, t_n) \rightarrow \varphi)$ — and the *existential binding patterns* — those that arise from a subformula $\exists \vec{x}\ (R(t_1, \ldots, t_n) \wedge \varphi)$. These can be defined formally by modifying the inductive definition above. Note that every binding pattern of a formula is either a universal pattern or an existential pattern, since a non-quantified relation can be considered as a vacuous case of existential quantification (if occurring positively) or universal quantification (if occurring negatively).

Intuitively, $\mathsf{BindPatt}(\varphi)$ describes the kind of access that is used if φ is evaluated in an instance using a straightforward inductive evaluation procedure. For a sentence φ, if for each pattern $(R\{s_1 \ldots s_j\})$ in $\mathsf{BindPatt}(\varphi)$ Sch contains an access method on R whose input positions are contained in $\{s_1 \ldots s_j\}$, then φ is an executable FO Boolean query. We say that a binding pattern $(R_1, \{s_1 \ldots s_j\})$ is *covered by* another binding pattern $(R_2, \{t_1 \ldots t_k\})$ if $R_1 = R_2$ and $\{s_1 \ldots s_j\}$ is a superset of $\{t_1 \ldots t_k\}$. To show that a sentence φ_1 is an executable FO Boolean query, it suffices to get another executable FO Boolean query φ_2 whose binding patterns cover those of φ_2.

Recall that a relation R *occurs positively* (*negatively*) in a formula φ if some occurrence of R in φ is in the scope of an even (odd) number of negations. For the purpose of this definition, we view the implication symbol as a shorthand: $\psi \to \chi$ stands for $\neg\psi \vee \chi$. For example, in the formula $\forall x\ (P(x) \to \exists y\ R(x, y))$, the relation P occurs negatively and the relation R occurs positively.

▶ **Theorem 24** (Access interpolation). *Let λ and ρ be* RQFO *sentences such that $\lambda \models \rho$. Then there exists an* RQFO *sentence χ such that*

1. *$\lambda \models \chi$ and $\chi \models \rho$.*
2. *A relation occurs positively (negatively) in χ only if it occurs positively (negatively) in both λ and ρ.*
3. *A constant occurs in χ only if it occurs both in λ and ρ.*
4. *Every existential binding pattern of χ is covered by an existential binding pattern of ρ, and the relation it binds occurs positively in λ. Every universal binding pattern of χ is covered by a universal binding pattern of λ, and the relation it binds occurs negatively in ρ.*
5. *If λ and ρ are both equality-free, then χ is equality-free.*

Furthermore, χ can be computed in polynomial time from a proof (in a suitable proof system) of the entailment $\lambda \models \rho$.

We show how Theorem 24 suffices to prove Theorem 23. Recall the theorem:

> For any Boolean RQFO query Q and access schema Sch containing constraints expressible in RQFO, there is an RA-plan answering Q (over instances in Sch) if and only if Q entails $\mathsf{InfAcc}Q$ with respect to the rules in $\mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch})$. If the query and constraints do not include equality, then the RA-plan will not make use of equality in any of its relational algebra expressions.

From Claim 22, we obtain the "plan-to-proof" direction of Theorem 23. Suppose Q does not imply $\mathsf{InfAcc}Q$ with respect to the rules in $\mathsf{AcSch}^{\leftrightarrow}(\mathsf{Sch})$. By Claim 22, Q is not access-determined, and it follows that no plan can answer Q.

For the "proof-to-plan" direction of Theorem 23, we assume Q entails $\mathsf{InfAcc}Q$ and construct an RA-plan that answers Q. We will use a slight modification of the axiom schema $\mathsf{AcSch}^{\leftrightarrow}$, denoted $\mathsf{AltAcSch}^{\leftrightarrow}$ in which the relation accessible does not appear. In every forward accessibility axiom an atom $\mathsf{accessible}(x)$ on the left is replaced by a relation $\mathsf{InfAcc}R(\vec{z})$, where $\vec{z}$ contains x in at least one position (and the other variables are universally quantified). Occurrences of accessible on the right are dropped. For example, the axiom $\mathsf{accessible}(x) \wedge R(x, y) \to \mathsf{InfAcc}R(x, y) \wedge \mathsf{accessible}(y)$ would be replaced by many axioms, including $\mathsf{InfAcc}S(x, w, z) \wedge R(x, y) \to \mathsf{InfAcc}R(x, y)$. We also allow all variants of these axioms in which free variables corresponding to input positions on the left are substituted by schema constants. Similarly, in every backward accessibility axiom we replace $\mathsf{accessible}(x)$ on the left by an atom in the original schema containing x, again dropping occurrences of accessible on the right.

▶ **Proposition 25.** *Q proves* $\mathsf{InfAcc}Q$ *using the axioms of* $\mathsf{AcSch}^{\leftrightarrow}$ *if and only if Q proves* $\mathsf{InfAcc}(Q)$ *in the modified schema* $\mathsf{AltAcSch}^{\leftrightarrow}$.

Proof. In one direction, suppose that Q does not prove $\mathsf{InfAcc}Q$ using the axioms of $\mathsf{AcSch}^{\leftrightarrow}$. Then by Claim 22, Q is not access-determined. Fix I and I' instances for the original schema satisfying Σ with the same accessible part, but disagreeing on the output of Q. Without loss of generality, we can assume that there is a tuple $\vec{d}$ in the output of Q on I but not in the output of Q on I'. By applying an isomorphism to non-accessible values of I' and I, we can assume that every non-accessible value of I' is not in I and vice versa. Let I^* be the instance in which relations R are interpreted as in I and relations $\mathsf{InfAcc}R$ are interpreted, as in I', by a new relation symbol.

We argue that I^* satisfies $\mathsf{AltAcSch}^{\leftrightarrow}$. Clearly both the original relations and the relations $\mathsf{InfAcc}R$ satisfy Σ. Consider a modified forward accessibility axiom (universal quantifiers omitted, and with no schema constants for simplicity):

$$\mathsf{InfAcc}R_{j_1}(\ldots x_{j_1}\ldots) \wedge \ldots \mathsf{InfAcc}R_{j_m}(\ldots x_{j_m}\ldots) \wedge R(\vec{x}) \rightarrow \mathsf{InfAcc}R(\vec{x})$$

where R has an access method on positions $j_1 \ldots j_m$. Suppose we have a tuple $c_1 \ldots c_m$ in I^* satisfying the left-hand side of this implication. Then $c_{j_1} \ldots c_{j_m}$ must be accessible values of I and I'. Since the fact $R(\vec{c})$ holds in I, it must be in the accessible part of I, and hence in the accessible part of I'. So $\mathsf{InfAcc}R(\vec{c})$ holds in I^* as required. The backward accessibility axioms are argued symmetrically.

The tuple $\vec{d}$ is in $[\![Q]\!](\mathsf{I}^*)$, since it is in $[\![Q]\!](\mathsf{I})$ and Q is a formula using the relations in the original schema. But $\vec{d}$ is not in $[\![Q]\!](\mathsf{I}')$. Therefore $\vec{d}$ can not be returned by $\mathsf{InfAcc}Q$ on I^*. So I^* witnesses that Q does not prove $\mathsf{InfAcc}Q$ in $\mathsf{AltAcSch}^{\leftrightarrow}$.

In the other direction, suppose we have I^* witnessing that Q does not prove $\mathsf{InfAcc}(Q)$ in $\mathsf{AltAcSch}^{\leftrightarrow}$. Expand I^* to an instance I^+ for the signature extended with $\mathsf{accessible}$, by interpreting $\mathsf{accessible}$ by all values of schema constants unioned with all values that lie in the domain of some $\mathsf{InfAcc}R$ relation and in the domain of some relation of the original schema. We show that the resulting instance satisfies the constraints of $\mathsf{AcSch}^{\leftrightarrow}$. The accessibility axioms follow directly from the corresponding axioms of $\mathsf{AltAcSch}^{\leftrightarrow}$. Similarly, the output of Q in I^+ is the same as the output of Q in I, while the output of $\mathsf{InfAcc}Q$ on I^+ is the same as the output of $\mathsf{InfAcc}Q$ on I^*. Therefore Q does not prove $\mathsf{InfAcc}Q$ in $\mathsf{AcSch}^{\leftrightarrow}$. ◀

We can rephrase the modified assumption on Q as:

$$Q \wedge \Sigma_1 \models \Sigma_2 \rightarrow \mathsf{InfAcc}Q$$

where Σ_1 contains the Sch constraints along with the "backwards" accessibility axioms going from $\mathsf{InfAcc}R$ relations to R relations within $\mathsf{AltAcSch}^{\leftrightarrow}$, while Σ_2 contains the forward accessibility axioms and the $\mathsf{InfAccCopy}$ constraints.

By the access interpolation theorem there is an RQFO formula χ in negation normal form such that:

1. $Q \wedge \Sigma_1 \models \chi$
2. $\chi \models \Sigma_2 \rightarrow \mathsf{InfAcc}Q$
3. a relation occurs positively (respectively negatively) in χ if and only if it occurs positively (resp. negatively) on both sides of the entailment.
4. for a binding pattern p on a relation U in χ if p is existential, then it is covered by an existential pattern in $\Sigma_2 \rightarrow \mathsf{InfAcc}Q$, and U occurs positively in $Q \wedge \Sigma_1$. If p is a universal pattern, then it is covered by a universal pattern of $Q \wedge \Sigma_1$ and U occurs negatively in $\Sigma_2 \rightarrow \mathsf{InfAcc}Q$.

Because we are dealing with the modified axioms $\mathsf{AltAcSch}^{\leftrightarrow}$, the only relations in the entailment are the original schema relations and their InfAcc copies. Occurrences of the original schema relations R on the right-hand side of the entailment are all in the forward accessibility axioms, and they correspond exactly to access methods of the schema. It follows that every existential pattern over the relations R in χ is of the proper form. Furthermore the relations R occur only positively on the right — since $\Sigma_2 \rightarrow \mathsf{InfAcc}Q$ in negation normal form is $(\sim \Sigma_2) \vee \mathsf{InfAcc}Q$, and $\sim$ applied to a forward accessibility axiom is of the form $\exists \vec{x}\ R(\vec{x}) \wedge \ldots \wedge \neg \mathsf{InfAcc}R(\vec{x})$. So relations R can occur only positively in χ, and hence can occur only in existential patterns of χ. We conclude that all the patterns in χ involving the relations of the original schema must be existential and of the proper form.

Now let us examine occurrences of relations of the form $\mathsf{InfAcc}R$ within χ. The last property of χ above implies that any existential pattern on relations of the form $\mathsf{InfAcc}R$ would need to correspond to a positive occurrence of $\mathsf{InfAcc}R$ in $Q \wedge \Sigma_1$. But there are no such occurrences. Any universal pattern on relations of the form $\mathsf{InfAcc}R$ would need to correspond to a universal pattern in $Q \wedge \Sigma_1$, and hence must be covered by the use of $\mathsf{InfAcc}R$ in a backward axiom. Hence all such occurrences are covered by an access method of the schema.

We conclude that χ is an executable FO Boolean query. Let $\mathsf{DeAcc}(\chi)$ be the result of changing all relations of the form $\mathsf{InfAcc}R$ in χ to R. We claim that $\mathsf{DeAcc}(\chi)$ is an executable rewriting of Q (and hence can be converted to an RA-plan using Proposition 19).

We justify this by proving containments between Q and $\mathsf{DeAcc}(\chi)$ in both directions. Suppose tuple $\vec{t}$ is returned by Q on an instance I of Sch. Let I' be the instance for the augmented schema in which both $\mathsf{InfAcc}R$ and R relations are interpreted as in I. We can see that I' satisfies the constraints in Σ_1 and Σ_2.

Applying the first condition on an interpolant above, we infer that χ holds of $\vec{t}$ in I', which implies that χ holds of $\vec{t}$ in I. But $\mathsf{DeAcc}(\chi)$ evaluated on I yields the same set of tuples as evaluating χ on I'. So $\vec{t}$ satisfies $\mathsf{DeAcc}(\chi)$ in I.

In the other direction, suppose tuple $\vec{t}$ satisfies $\mathsf{DeAcc}(\chi)$ in I. Then letting I' be as above, we have that $\vec{t}$ satisfies χ in I'. By the second property of an interpolant we have that $\Sigma_2 \rightarrow \mathsf{InfAcc}Q$ holds in I'. Since I' satisfies Σ_2, $\vec{t}$ is returned by $\mathsf{InfAcc}Q$ in I'. This tells us that $\vec{t}$ is returned by Q as required.

This completes the proof of the first assertion in Theorem 23. The assertion about equality-free queries and constraints follows since the Access interpolation theorem produces an equality-free interpolant when the entailment involves equality-free formulas, and the other transformations (e.g., from nested plans to RA-plans) do not introduce equality.

6.3 Variants of the Results for $USPJ^{\neq}$-Plans

We now state an analogous result for negation-free plans, which will be an access-related variant of the Projective Monotone Preservation Theorem, Theorem 14.

Here we use the schema $\mathsf{AcSch}(\mathsf{Sch})$ obtained from $\mathsf{AcSch}^{\leftrightarrow}$ by throwing away all the backward accessibility axioms, and keeping only the forward ones.

▶ **Theorem 26.** *For any Boolean* RQFO *query* Q *and access schema* Sch *containing constraints specified in* RQFO*, the following are equivalent:*

- *there is a* $USPJ^{\neq}$*-plan answering* Q *(over instances in* Sch*)*
- Q *entails* $\mathsf{InfAcc}Q$ *with respect to* $\mathsf{AcSch}(\mathsf{Sch})$ *(*$Q \wedge \mathsf{AcSch}(\mathsf{Sch}) \models \mathsf{InfAcc}Q$*).*

Furthermore, for every tableau proof witnessing $Q \wedge \mathsf{AcSch}(\mathsf{Sch}) \models \mathsf{InfAcc}Q$*, we can extract a* $USPJ^{\neq}$*-plan. If the query and the constraints of* Sch *are specified by* RQFO *formulas without any equalities (e.g., TGDs), then we can replace* $USPJ^{\neq}$*-plan with* $USPJ$*-plan in the above statement. So we do not need inequalities in the reformulation unless we have equalities in the constraints.*

The Schema $\mathsf{AcSch}^{\neg}$ and $USPJAD^{\neq}$-Plans. We now state an extension of the "Projective Łos-Tarski theorem", Theorem 17, to the setting of access methods, again focusing on the case of Boolean queries.

▶ **Theorem 27.** *For any Boolean* RQFO *query* Q *and access schema* Sch *containing constraints specified in* RQFO*, the following are equivalent:*

- *there is a* $USPJAD^{\neq}$*-plan answering* Q *(over instances for* Sch*)*
- $Q \wedge \mathsf{AcSch}^{\neg}(\mathsf{Sch}) \models \mathsf{InfAcc}Q$*. That is,* Q *entails* $\mathsf{InfAcc}Q$ *with respect to* $\mathsf{AcSch}^{\neg}(\mathsf{Sch})$*.*

Furthermore, from every tableau proof witnessing $Q \wedge \mathsf{AcSch}^{\neg}(\mathsf{Sch}) \models \mathsf{InfAcc}Q$*, we can effectively extract a* $USPJAD^{\neq}$*-plan. If the constraints of* Sch *and the query* Q *are specified by equality-free* RQFO *sentences (in particular, TGDs), then we can replace* $USPJAD^{\neq}$*-plan by* $USPJAD$*-plan.*

In the vocabulary-based setting, the entailment $Q \wedge \mathsf{AcSch}^{\neg}(\mathsf{Sch}) \models \mathsf{InfAcc}Q$ reduces to the entailment given in Section 5, which was shown there to be equivalent to the query being induced-subinstance-monotonically-determined. And clearly, in the vocabulary-based setting, a $USPJAD^{\neq}$-plan can be expressed as an $\exists^{\neq}$ formula. Thus Theorem 27 generalizes the

"Projective Łos-Tarski Theorem", Theorem 17, characterizing queries that can be reformulated using $\exists^{\neq}$ formulas.

Both of these results are proven in the same way as the result for RA plans. By Proposition 19, instead of dealing with plans as the target language, we deal with RQFO formulas with certain binding patterns. We apply the Access Interpolation Theorem, and then analyze the binding patterns in the resulting formulas, and show that we satisfy the requirements to convert to the associated plan.

7 State of Play and Future Directions

Here we have given a flavor of the use of interpolation in database rewriting problems, focusing on the cases of vocabulary-based restriction and access patterns.

Several other directions have been explored in the literature:

Richer Data Models. The examples we have gone through here both concern relational data - i.e. traditional set-based semantics. In databases there are tools and query languages for other collection types, like bags, nested sets, nested bags, sequences, and nested sequences. The main work we know of beyond relational data is about nested sets, summarized in [9]. The model starts with an infinite scalar type, and builds up objects via iterating tupling and powersets: thus one can have sets of sets of pairs of scalars. Pairs of sets of sets of scalars, etc. There is a standard query language in the literature for this data model, nested relational calculus (NRC). In [9] a notion of implicit definability/determinacy is defined for first-order logic formulas in a simple set-theoretic language. And it shown that any query over nested sets that is determined in this sense can be converted to NRC, effectively from a certain kind of proof. Although at first glance this appears to be a version of Beth Definability for a fragment of set theory, in fact the only set theoretic axiom used is extensionality: sets with equal members are equal. Thus implicit definability in the context of nested data is really a variation of standard implicit definability where the uniqueness is only "up to extensional equivalence". Extensional equivalence is definable in the language of set theory. Thus the question is whether there is a version of Beth's theorem that considers uniqueness up to definable equivalence. One of the key insights in [9] is that there is a variation of Beth that applies to this setting, originating in work of Gaifman [18] under the name of "rigid categoricity". While the results on rigid categoricity were proven model-theoretically in [18] (see [1]), in [9] a proof-theoretic argument is developed for the special case of extensional equivalence.

A natural question is what about the case of *bags*: finite multi-sets. While there is no standard semantics for first-order logic over bags, there is a semantics for conjunctive queries, and thus one can talk about determinacy of CQ views over bag semantics. Note that even for Boolean queries, it is not clear whether determinacy is decidable, or whether it implies rewritability in some natural query language for bags. The main results on this topic are in [20], where the case of Boolean views is demystified.

Richer Query Languages over Traditional Data Models. The work we overview deals with fragments of first-order logic. The only other work we are aware of in this space concerns fragments of the recursive query language Datalog, a fragment of fixpoint logic. The first work in this space was [17], which investigates the case of views in a fragment of Datalog, the "regular path queries": it is shown that determinacy implies rewritability, an analog of implicit definability. The case of general Datalog queries and views was investigated in [4, 3]. The only implicit definability results are for fragments of Datalog where containment is decidable — which means, very limited fragments. The main technique used is a variation of uniform interpolation, which hold for limited fixpoint languages [14]. See Chapter 9 for a discussion of uniform interpolation algorithms.

Other Proof Systems and Implementations. Toman and Weddell were the first to develop interpolation-methods for query rewriting. Their initial approach, based on a tableaux calculus, is presented in [26], with follow-up work in [27, 16]. A rewriting approach for access patterns was developed in [8, 7], but only in a restricted context where interpolation is not necessary. [5] developed an approach for interpolation-based rewriting where the underlying proof system is resolution. This made use of the standard algorithmic approach to resolution-based interpolation due to Huang [19], which is discussed in Chapter 12.

8 Bibliographic Remarks

The precursors of the interpolation-based approach to rewriting are in preservation or "semantics to syntax" theorems in model theory, such as the Łos-Tarski theorem, which states that preservation under extensions is equivalent to being rewritable as an existential formula: see for example [1, 12]. After the work of Lyndon [22], these results would be stated and proved model-theoretically, and effective aspects would be ignored. In addition, the presentations did not include the "projective versions", in which one looks for a restricted formula in a particular vocabulary: these variants are important for the applications in databases.

The idea that Craig interpolation is relevant to query reformulation in databases, from the point of view of theory, dates from the work of Segoufin and Vianu [25], later explored jointly with Nash [23]. Their investigation focused on the case of views — a special case of vocabulary-based reformulation. They phrase their results as showing that relational algebra was sufficient in seeking a reformulation over views.

Toman and Weddell [26] were the first to explore the use of effective interpolation to actually generate rewritings, in the context of query reformulation with constraints. Toman and Weddell also note connections with other query reformulation methods that have an interpolation-like feel, like the 'chase and backchase method", which is used to reformulate queries in the presence of a certain class of background theories, so-called Tuple-Generating Dependencies (TGDs): for an overview of the chase and backchase, see [15].

Much of the material in this chapter here is taken from [6], which extends earlier work [10, 11] looking systematically at interpolation for both vocabulary-based reformulation and rewriting with access methods. In this chapter we have given some samples of vocabulary-based reformulation for "first-order" (or relational algebra) queries, and on access-method based reformulation for the same queries, but the book covers several other cases. The book and the related papers explicitly connect these reformulation methods with preservation theorems and other semantics-to-syntax results in the earlier model theory literature.

The case of so-called "nested relations", which can be considered as a fragment of set theory, is taken from [9]. We have overviewed these results briefly in Section 7.

The case of Datalog (fixpoint) queries and views is considered in [17, 4, 3]. Here the results are related to uniform interpolation, and they work via automata-theoretic methods, as in [14].

References

1 Wilfrid Hodges. *Model Theory*. Cambridge University Press, 1993.

2 Serbe Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

3 Michael Benedikt, Stanislav Kikot, Johannes Marti, and Piotr Ostropolski-Nalewaja. Monotone rewritability and the analysis of queries, views, and rules. In *KR*, 2024.

4 Michael Benedikt, Stanislav Kikot, Piotr Ostropolski-Nalewaja, and Miguel Romero. On monotonic determinacy and rewritability for recursive queries and views. *TOCL*, 24(2):16:1–16:62, 2023.

5 Michael Benedikt, Egor V. Kostylev, Fabio Mogavero, and Efthymia Tsamoura. Reformulating queries: Theory and practice. In *IJCAI*, 2017.

6 Michael Benedikt, Julien Leblay, Balder ten Cate, and Efthymia Tsamoura. *Generating Plans from Proofs: the Interpolation-based approach to Query Reformulation*. Morgan Claypool, 2015.

7 Michael Benedikt, Julien Leblay, and Efi Tsamoura. PDQ: Proof-driven query answering over web-based data. In *VLDB*, 2014.

8 Michael Benedikt, Julien Leblay, and Efthymia Tsamoura. Querying with access patterns and integrity constraints. *PVLDB*, 8(6):690–701, 2015.

9 Michael Benedikt, Cécilia Pradic, and Christoph Wernhard. Synthesizing nested relational queries from implicit specifications: via model theory and via proof theory. *Log. Methods Comput. Sci.*, 20(3), 2024.

10 Michael Benedikt, Balder ten Cate, and Efi Tsamoura. Generating low-cost plans from proofs. In *PODS*, 2014.

11 Michael Benedikt, Balder Ten Cate, and Efthymia Tsamoura. Generating plans from proofs. *TODS*, 40(4), February 2016.

12 C. C. Chang and H. J. Keisler. *Model Theory*. North-Holland, 1990.

13 William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.

14 Giovanna D'Agostino and Marco Hollenberg. Logical questions concerning the μ-calculus: Interpolation, Lyndon and Łos-Tarski. *J. Symb. Log.*, 65(1):310–332, 2000.

15 Alin Deutsch, Lucian Popa, and Val Tannen. Query reformulation with constraints. *SIGMOD Record*, 35(1):65–73, 2006.

16 Eva Feng, David Toman, and Grant E. Weddell. Magic sets in interpolation-based rule driven query optimization. In *RuleML+RR*, 2022.

17 Nadime Francis, Luc Segoufin, and Cristina Sirangelo. Datalog rewritings of regular path queries using views. *Log. Methods Comput. Sci.*, 11(4), 2015.

18 Haim Gaifman. Operations on relational structures, functors and classes I. In *Proc. of the Tarski Symposium*, volume 25, page 20–40, 1974.

19 Guoxiang Huang. Constructing Craig interpolation formulas. In *COCOON*, 1995.

20 Jaroslaw Kwiecien, Jerzy Marcinkowski, and Piotr Ostropolski-Nalewaja. Determinacy of real conjunctive queries. the boolean case. In *PODS*, 2022.

21 Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, 2002.

22 Roger C. Lyndon. An interpolation theorem in the predicate calculus. *Pacific Journal of Mathemantics*, 9:129–142, 1959.

23 Alan Nash, Luc Segoufin, and Victor Vianu. Views and queries: Determinacy and rewriting. *TODS*, 35(3), 2010.

24 Martin Otto. An interpolation theorem. *Bulletin of Symbolic Logic*, 6(4):447–462, 2000.

25 Luc Segoufin and Victor Vianu. Views and queries: determinacy and rewriting. In *PODS*, 2005.

26 David Toman and Grant Weddell. *Fundamentals of Physical Design and Query Compilation*. Morgan Claypool, 2011.

27 David Toman and Grant E. Weddell. FO Rewritability for OMQ using Beth Definability and Interpolation. In *DL*, 2021.

Chapter 15

Interpolation in Knowledge Representation

Jean Christoph Jung
TU Dortmund University, Germany

Patrick Koopmann
Vrije Universiteit Amsterdam, Netherlands

Matthias Knorr
Universidade Nova de Lisboa, Portugal

Abstract

Craig interpolation and uniform interpolation have many applications in knowledge representation, including explainability, forgetting, modularization and reuse, and even learning. At the same time, many relevant knowledge representation formalisms do in general not have Craig or uniform interpolation, and computing interpolants in practice is challenging. We have a closer look at two prominent knowledge representation formalisms, description logics and logic programming, and discuss theoretical results and practical methods for computing interpolants.

Contents

1 Introduction 540
2 Foundations of Description Logics 544
3 Uniform Interpolation for Description Logic Ontologies 547
3.1 Existence and Size 548
3.2 Computing Uniform Interpolants in Practice 552
3.3 Related Notions and Applications 556
4 Craig Interpolation for Description Logic Concepts 558
4.1 Craig Interpolation 558
4.2 Beth Definability 564
4.3 Applications 567
4.4 Computing Interpolants 568



© Jean Christoph Jung, Patrick Koopmann, and Matthias Knorr;
licensed under Creative Commons License CC BY 4.0

In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 15, pages 539–588. Ubiquity Press, 2026. https://www.doi.org/10.5334/bdg.o

5 Interpolation in Logic Programming 571
5.1 Logic Programs and Answer Sets 571
5.2 Craig Interpolants 573
5.3 Uniform Interpolation and Forgetting 574
6 Conclusion 577
Acknowledgments 578
References 578

1 Introduction

The field of Knowledge Representation and Reasoning (KR) deals with the explicit representation and manipulation of knowledge in a format that is both *machine-processable* and *human-readable*, and plays a central role in symbolic AI [59, 21]. AI systems using KR are referred to as *knowledge-based systems*. Depending on the application, they use different *knowledge representation formalisms* satisfying the mentioned two conditions. Since interpolation is an inherently logical notion, in this chapter, we will concentrate on logic-based formalisms, and neglect, e.g., graph-based KR formalisms such as argumentation frameworks [14] or probabilistic graphical models [73]. Knowledge-based systems then implement inference procedures tailored to the underlying logic and use them to make decisions.

Let us illustrate the idea underlying KR with a simple example, formulated in terms of first-order logic. The following statements could be part of a formalization of our knowledge about cars that (1) every car has a prime mover, (2) the prime mover can be a diesel engine, a gas engine or an electric motor, and (3) every car with an electric motor is an electric car:

$$\forall x\ (\mathrm{Car}(x) \rightarrow \exists y\,(\mathrm{hasPart}(x,y) \wedge \mathrm{PrimeMover}(y))) \tag{1}$$

$$\forall x\ (\mathrm{PrimeMover}(x) \rightarrow (\mathrm{DieselEngine}(x) \vee \mathrm{GasEngine}(x) \vee \mathrm{ElectricMotor}(x))) \tag{2}$$

$$\forall x\ (\mathrm{Car}(x) \wedge \exists y\,(\mathrm{hasPart}(x,y) \wedge \mathrm{ElectricMotor}(y)) \rightarrow \mathrm{ElectricCar}(x)) \tag{3}$$

If we now find out that a car c has an electric motor, we can feed this knowledge and additional facts

$$\mathrm{Car}(c), \quad \mathrm{hasPart}(c,e), \quad \mathrm{ElectricMotor}(e)$$

into the inference procedure, which allows us to derive that c is an electrical car.

Interpolation plays a central role in many applications of KR. To ease the discussion, we recall the definition of Craig and uniform interpolants, formulated here for first-order logic (FO) with entailment relation $\models$. Here, and in the remainder of the introduction, a *signature* is a set of non-logical symbols.

▶ **Definition 1.** *A* Craig interpolant for *FO-formulae ϕ, ψ is a formula χ such that*
- *χ uses only non-logical symbols occurring in both ϕ and ψ, and*
- *$\phi \models \chi$, $\chi \models \psi$.*

Let Σ be signature. A uniform Σ-interpolant for *ϕ is an FO-formula ϕ_Σ such that:*
- *$\phi \models \phi_\Sigma$;*
- *ϕ_Σ uses only non-logical symbols from Σ;*
- *for any FO-formula ψ, if $\phi \models \psi$ and ψ uses only non-logical symbols from Σ, then $\phi_\Sigma \models \psi$.*

In applications, the formula ϕ from Definition 1 will typically be the modeled knowledge, which is why we will denote it with $\mathcal{K}$, for *knowledge base*. Alghough knowledge bases come in different forms for different KR formalisms, we refrain from specifying the exact formalism in the applications below, as we only want to convey the general ideas. We start with applications of uniform interpolants.

Forgetting and Information Hiding The most classical connection between interpolation and KR might be *forgetting*, originally introduced for first-order logic in a seminal paper by Lin and Reiter [92]. The idea of forgetting is that there are non-logical symbols that are considered "obsolete" and should be "forgotten" from the current knowledge in a way that preserves as much information as possible. There are different definitions of forgetting that can be found in the KR literature, see [33] for a survey, but the most common one is equivalent to uniform interpolation: forgetting a symbol X from a knowledge base $\mathcal{K}$ corresponds to computing a uniform interpolant of $\mathcal{K}$ for the signature $\mathsf{sig}(\mathcal{K}) \setminus \{X\}$. Forgetting is also useful for applications that require some form of *information hiding*, such as knowledge publication and exchange. For instance, if we were to publish $\mathcal{K}$ without disclosing anything about a private signature Σ, then a uniform $(\mathsf{sig}(\mathcal{K}) \setminus \Sigma)$-interpolant of $\mathcal{K}$ would be a good candidate, as is contains the "maximal" information contained in $\mathcal{K}$ about the remaining signature [57].

Abduction Abduction is a classical problem in KR that can be formalized as follows: given a knowledge base $\mathcal{K}$ and an *observation* ψ such that $\mathcal{K} \not\models \psi$, we want to find a *hypothesis* $\mathcal{H}$ satisfying $\mathcal{K} \wedge \mathcal{H} \models \psi$. The hypothesis $\mathcal{H}$ can be seen as a possible explanation for some unexpected phenomenon ψ, a diagnosis, or an indication of how to complete an incomplete knowledge base. To avoid trivial solutions, one typically formulates additional requirements on the hypothesis such as certain minimality conditions or a signature Σ from which the hypothesis ought to be constructed. It is immediate from the definitions that the negation of a Σ-uniform interpolant of $\mathcal{K} \wedge \neg\psi$ is a *logically weakest* hypothesis, that is, it is entailed by all other alternative hypotheses over signature Σ. This connection has been exploited in several abduction systems [43, 31, 82].

Modularisation and Reuse Knowledge bases often contain tens or even hundreds of thousands of statements (e.g. [32, 78]) which can make it challenging to work with them. At the same time, it could be that for a particular application, only a fragment of the knowledge base is

actually relevant. Uniform interpolants computed for a restricted, user-given signature can provide a more focussed view on the knowledge that is relevant to the application at hand, and can be used as a replacement of the original knowledge base. But uniform interpolants can also be used to determine whether a subset of the knowledge base preserves all entailments over a relevant signature, which can be used to *modularize* the knowledge base into different components based on a signature.

Analysis and Summarization Uniform interpolants for small signatures Σ can make hidden relations between the symbols in Σ explicit, and thus help knowledge engineers in analyzing how different symbols in the knowledge base relate to each other. We can also use uniform interpolation to summarize large knowledge bases, for instance by choosing for Σ a set of symbols that is central to the knowledge base (e.g. most frequently used). The interpolant then gives a high-level perspective on the content of the knowledge base, which can be seen as a summary.

Craig interpolants have a range of applications for KR as well.

Explaining Entailments One of the main benefits of KR is that the use of explicit representations of knowledge with a well-defined semantics leads to a higher transparency and understandability of the knowledge-based system. Still, for large knowledge bases and expressive KR formalisms, inferences may not be straightforward, which has motivated many researchers to investigate methods for explaining logical inferences for KR systems [106, 122, 35, 126, 3]. Craig interpolants are one way of explaining logical inferences: in particular, to explain an entailment $\mathcal{K} \models \psi$, we can provide a Craig interpolant for $\mathcal{K}, \psi$, which highlights the reason for the entailment in the common signature of $\mathcal{K}$ and ψ. Notice that the existence of a Craig interpolant for $\mathcal{K}, \psi$ is a weaker requirement than the existence of a uniform interpolant for $\mathcal{K}$, and indeed Craig interpolants can be computed in more cases.

Explicit Definitions It is well-known that Craig interpolation is closely related to the notion of *Beth definability*. In the context of KR, this connection is particularly relevant for the problem of finding *explicit definitions* of predicates which are *implicitly defined*. An explicit definition of a unary predicate A in knowledge base $\mathcal{K}$ is a formula $\chi(x)$ not mentioning A with $\mathcal{K} \models \forall x\,(A(x) \leftrightarrow \chi(x))$. That is, χ provides a direct description of the *meaning* of A, and we might add this definition to $\mathcal{K}$. A special kind of explicit definitions are *referring expressions*. Referring expressions are a concept originating in linguistics and are phrases that refer to specific objects by providing a unique description for them, as in "the current president of the USA" or "the capital of France". In a knowledge base $\mathcal{K}$, a referring expression for a constant c can be viewed as an explicit definition of $x = c$ over $\mathcal{K}$. The use of referring expressions in KR and data management has been advocated for instance in [7, 84, 19, 9].

Separating Examples and Learning Logical Formulas Consider the following separation problem. Suppose we are given two sets P and N of constants occurring in a knowledge base $\mathcal{K}$ that respectively represent *positive* and *negative* examples. We are then looking for a logical formula $\chi(x)$ that *separates* P from N over $\mathcal{K}$ in the sense that $\mathcal{K} \models \chi(a)$ for all $a \in P$ and $\mathcal{K} \models \neg\chi(b)$ for all $b \in N$. Such separation problems have been investigated thoroughly in KR in the context of learning logical formulas [70, 39]. Recently, it has been observed that in relevant cases, the problem of finding a separating formula is interreducible with the problem of computing Craig interpolants [8].

Motivated by these applications, there has been an enormous interest in studying interpolation in different formalisms. The baseline logical formalisms used in KR are certainly the classical *propositional logic* and *first-order logic*, which have the benefit of a clean and well-understood semantics. However, for many realistic applications, neither of them is well-suited, since they are either limited in their expressivity, or do not admit time-efficient reasoning. As a consequence, a wealth of other formalisms have been introduced that are tailored towards specific applications. Examples include description logics, modal and temporal logics, logic programs, default logic, existential rules, planning languages, and many more [59]. Interpolation is useful in many of these formalisms, but it has not been equally investigated in all of them. Interpolation in propositional logic is covered in Chapter 1, interpolation in modal logic is covered in Chapter 4, and interpolation in first-order logic is covered in Chapter 2 and Chapter 12. The majority of research on interpolation in the remaining formalisms has been conducted in description logics and logic programming, which is why we concentrate on these two formalisms in this chapter:

- *Description logics (DLs)* are a family of KR languages commonly used to formalize ontologies [11]. Here, an *ontology* is a formal specification of the concepts and their relations in some domain of interest such as biology or medicine. An example ontology (in first-order logic) is provided in Equations (1)–(3) above. Ontologies are important in information science, since they can be used by different parties to share knowledge about that domain, which has been exploited, for example, in biology, medicine, and artificial intelligence [78, 32, 44, 30]. DLs are also highly relevant in the *Semantic Web* [65]. Indeed, they form the logical basis of the W3C standard *web ontology language* OWL [66]. Typically, DLs are fragments of first-order logic with decidable inference problems, which makes them suitable for the mentioned applications.
- *Logic programming* is concerned with the use of *logical rules* to represent knowledge. One of the main features is that inference is *nonmonotonic*, that is, extending the knowledge base can invalidate previously made inferences. This is in contrast to *monotonic* logics such as first-order logic (and hence DLs as well). While logic programming has been studied since the 1960s, it is still relevant today, in particular in the form of *answer set programming* (ASP), which is a declarative approach to modeling and solving combinatorial problems [46]. It plays a central role in many applications, for instance for solving configuration problems.

It is particularly remarkable how much has been done on the topic of interpolation in DLs. We conjecture that this is due to the fact that the main applications of DLs are ontologies describing the conceptualization of a domain of discourse, a task that is ultimately linked with the signature, which is also central to interpolation. In the chapter, this will be reflected by the fact that we will mainly focus on interpolation for DLs, and only briefly discuss interpolation for logic programming. A peculiarity in the literature on interpolation in DLs, which has to do with the applications, is that uniform interpolation has been investigated mostly for the case where the knowledge base is an ontology, and Craig interpolation for DLs has been mostly investigated for concept descriptions, in which case the ontology is treated as background theory.

Looking at the applications, it appears that the central problem to be investigated is the *computation* of Craig and uniform interpolants. Unfortunately, their existence is by no means guaranteed in the relevant logics, so another focus of the chapter will be the corresponding *existence* problems, and the practically relevant question of what to do if no (Craig or uniform) interpolant exists. Given the technical similarity of description logics and modal logics, we will discuss the concrete relation (in terms of interpolation) when appropriate.

The chapter is structured as follows. In Section 2, we recall the necessary foundations of description logics. In Section 3, we discuss uniform interpolation for the case where the knowledge base is a description logic ontology, looking at both theoretical results and practical methods for computing interpolants. In Section 4, we discuss results on interpolation and Beth definability for DL concept descriptions from a more theoretical perspective. In Section 5, we discuss the role of Craig interpolants in logic programming, as well as uniform interpolation and forgetting. Finally, in Section 6, we conclude the chapter and provide an outlook for future directions.

2 Foundations of Description Logics

We first introduce the syntax and semantics of the basic description logic $\mathcal{ALC}$, discuss the extensions/restrictions relevant for the chapter, and introduce some model theory. We refer the reader to [11] for a comprehensive introduction to description logics. Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be mutually disjoint and countably infinite sets of *concept*, *role*, and *individual names*. An $\mathcal{ALC}$ *concept* is defined according to the syntax rule

$$C, D ::= \top \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall r.C \mid \exists r.C$$

where A ranges over concept names and r over role names. We use the abbreviations $C \to D$ and $C \leftrightarrow D$ for $\neg C \sqcup D$, and $(C \to D) \sqcap (D \to C)$, respectively. An $\mathcal{ALC}$ *concept inclusion ($\mathcal{ALC}$ CI)* takes the form $C \sqsubseteq D$ for $\mathcal{ALC}$ concepts C and D. An $\mathcal{ALC}$ *ontology* is a finite set of $\mathcal{ALC}$ CIs. We drop the reference to $\mathcal{ALC}$ if no confusion can arise.

The semantics is defined in terms of *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, called *domain* of $\mathcal{I}$, and $\cdot^{\mathcal{I}}$ is a function mapping every $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,

Table 1 Extensions of $\mathcal{ALC}$

acronym	syntax	semantics
$\mathcal{I}$	$\exists r^-.C$	$\{d \in \Delta^{\mathcal{I}} \mid \text{there exists } e \in C^{\mathcal{I}} : (e,d) \in r^{\mathcal{I}}\}$
$\mathcal{O}$	$\{a\}$	$\{a^{\mathcal{I}}\}$
$\mathcal{F}$	$(\leq 1\ r)$	$\{d \in \Delta^{\mathcal{I}} \mid \text{there is at most one } e \in \Delta^{\mathcal{I}} \text{ with } (d,e) \in r^{\mathcal{I}}\}$
$\mathcal{S}$	$\mathsf{trans}(r)$	$r^{\mathcal{I}}$ is transitive
$\mathcal{H}$	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

every $r \in \mathsf{N_R}$ to a subset $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every $a \in \mathsf{N_I}$ to an element in $\Delta^{\mathcal{I}}$. Moreover, the *extension* $C^{\mathcal{I}}$ *of a concept* C *in* $\mathcal{I}$ is defined as follows, where r ranges over role names:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}},\\
\neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},\\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}},\\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}},\\
(\exists r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \text{there exists } (d,e) \in r^{\mathcal{I}} \text{ with } e \in C^{\mathcal{I}}\ \},\\
(\forall r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \text{for all } (d,e) \in r^{\mathcal{I}} \text{ we have } e \in C^{\mathcal{I}}\ \}.
\end{aligned}$$

Let $\mathcal{O}$ be an ontology and $\mathcal{I}$ be an interpretation. Then $\mathcal{I}$ *satisfies* a CI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and $\mathcal{I}$ is a *model* of $\mathcal{O}$ if it satisfies all CIs in $\mathcal{O}$. The ontology $\mathcal{O}$ *entails* a CI $C \sqsubseteq D$, in symbols $\mathcal{O} \models C \sqsubseteq D$, if every model of $\mathcal{O}$ satisfies $C \sqsubseteq D$; it *entails* another ontology $\mathcal{O}'$, written $\mathcal{O} \models \mathcal{O}'$ if $\mathcal{O} \models C \sqsubseteq D$ for every CI $C \sqsubseteq D \in \mathcal{O}'$. In case $\mathcal{O} \models C \sqsubseteq D$, we also say that C *is subsumed by* D *under* $\mathcal{O}$. We drop $\mathcal{O}$ if it is empty and write $\models C \sqsubseteq D$ for $\emptyset \models C \sqsubseteq D$.

A *signature* Σ is a finite set of concept, role, and individual names, uniformly referred to as *symbols*. We use $\mathrm{sig}(X)$ to denote the set of symbols used in any syntactic object X such as a concept or an ontology. An $\mathcal{ALC}(\Sigma)$ *concept* is an $\mathcal{ALC}$ concept C with $\mathrm{sig}(C) \subseteq \Sigma$. The *size* of a (finite) syntactic object X, denoted $\|X\|$, is the number of symbols needed to represent it as a word. The *role depth* $\mathrm{rd}(C)$ of a concept C is the maximal nesting depth of existential and universal restrictions in C.

Extensions and Restrictions of $\mathcal{ALC}$ Depending on the application, one requires more or less expressive power to describe the domain knowledge, which is why different extensions and restrictions of $\mathcal{ALC}$ have been investigated. The most prominent restriction is certainly $\mathcal{EL}$, in which only the concept constructors $\top$, A, $C \sqcap D$, $\exists r.C$ are allowed (and thus no negation $\neg C$, disjunction $C \sqcup D$, and universal restriction $\forall r.C$) [10]. We also consider several important extensions, see [11] for more details on these. *Inverse roles* (abbreviated with the acronym $\mathcal{I}$)

take the form r^- for a role name r. They are interpreted as the inverse of the interpretation of the role, that is, $(r^-)^{\mathcal{I}} = \{(e,d) \mid (d,e) \in r^{\mathcal{I}}\}$ and can be used to travel edges in the converse direction. *Nominals* (acronym $\mathcal{O}$) take the form $\{a\}$ for individual names $a \in \mathsf{N_I}$ and are used to describe singleton concepts. *(Local) functionality restrictions* (acronym $\mathcal{F}$) take the form $(\leq 1\ r)$ and describe the set of all individuals that have at most one r-successor. *Role hierarchies* (acronym $\mathcal{H}$) take the form $r \sqsubseteq s$ and can be used to describe relations between roles. Finally, some of the roles may be declared as *transitive* (acronym $\mathcal{S}$), which means that they have to be interpreted as transitive relations. Table 1 displays an overview of the extensions and their semantics. The names of the extensions are obtained in a canonical way by appending the acronym of the additional constructor, e.g., $\mathcal{ALCHI}$ is the extension of $\mathcal{ALC}$ with role hierarchies and inverse roles. A bit of care has to be taken when combining certain features, e.g., transitive roles with functionality restrictions; we adopt the standard restrictions known from the literature [11].

▶ **Example 2.** The example from the introduction can be represented as follows as an $\mathcal{ALC}$ ontology. The first and last axioms are also in $\mathcal{EL}$.

$$\mathsf{Car} \sqsubseteq \exists \mathsf{hasPart}.\mathsf{PrimeMover} \qquad \mathsf{PrimeMover} \sqsubseteq \mathsf{DieselEngine} \sqcup \mathsf{GasEngine} \sqcup \mathsf{ElectricMotor}$$
$$\mathsf{Car} \sqcap \exists \mathsf{hasPart}.\mathsf{ElectricMotor} \sqsubseteq \mathsf{ElectricCar}$$

In $\mathcal{ALCFIO}$, we can additionally express that every engine belongs to at most one car, and that a German car is a car built in Germany.

$$\mathsf{PrimeMover} \sqsubseteq (\leq 1\ \mathsf{hasPart}^-) \qquad \mathsf{GermanCar} \equiv \mathsf{Car} \sqcap \exists \mathsf{madeIn}.\{\mathsf{Germany}\}$$ ⌟

Model Theory We next recall the model-theoretic notion of a bisimulation, which is very useful in the context of interpolation in DLs. Let Σ be a signature and $\mathcal{I}_1, \mathcal{I}_2$ be interpretations. A relation $Z \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ is an $\mathcal{ALC}(\Sigma)$-*bisimulation* between $\mathcal{I}_1$ and $\mathcal{I}_2$ if the following conditions are satisfied for all $(d,e) \in Z$:
Atom for all concept names $A \in \Sigma$: $d \in A^{\mathcal{I}_1}$ iff $e \in A^{\mathcal{I}_2}$,
Back for all role names $r \in \Sigma$ and all $(d,d') \in r^{\mathcal{I}_1}$, there is $(e,e') \in r^{\mathcal{I}_2}$ such that $(d',e') \in Z$,
Forth for all role names $r \in \Sigma$ and all $(e,e') \in r^{\mathcal{I}_2}$, there is $(d,d') \in r^{\mathcal{I}_1}$ such that $(d',e') \in Z$.
A *pointed interpretation* is a pair $\mathcal{I}, d$ with $\mathcal{I}$ an interpretation and $d \in \Delta^{\mathcal{I}}$. For pointed interpretations $\mathcal{I}_1, d_1$ and $\mathcal{I}_2, d_2$, we write $\mathcal{I}_1, d_1 \sim_{\mathcal{ALC},\Sigma} \mathcal{I}_2, d_2$ in case $\mathcal{I}_1, d_1$ and $\mathcal{I}_2, d_2$ are $\mathcal{ALC}(\Sigma)$-*bisimilar*, that is, if there is an $\mathcal{ALC}(\Sigma)$-bisimulation Z between $\mathcal{I}_1$ and $\mathcal{I}_2$ with $(d_1, d_2) \in Z$. Bisimulations are a powerful tool since they capture the expressive power of $\mathcal{ALC}$. Indeed, if $\mathcal{I}_1, d_1 \sim_{\mathcal{ALC},\Sigma} \mathcal{I}_2, d_2$, then d_1 and d_2 satisfy the same $\mathcal{ALC}(\Sigma)$ concepts, that is, $d_1 \in C^{\mathcal{I}_1}$ iff $d_2 \in C^{\mathcal{I}_2}$, for all $\mathcal{ALC}(\Sigma)$ concepts. The converse direction does not hold in general, but in relevant cases (for example, when $\mathcal{I}_1$ and $\mathcal{I}_2$ are finite). We refer the interested reader to [56] for a more detailed account on model theory.

For the purpose of this chapter, it is worth mentioning that for each extension/restriction $\mathcal{L}$ of $\mathcal{ALC}$ there is a corresponding notion of bisimulation, denoted $\sim_{\mathcal{L},\Sigma}$, that captures the

expressive power of $\mathcal{L}$. As a concrete example, an $\mathcal{ALCO}(\Sigma)$-bisimulation Z between $\mathcal{I}_1$ and $\mathcal{I}_2$ is an $\mathcal{ALC}(\Sigma)$-bisimulation that additionally satisfies the following for all $(d, e) \in Z$:
AtomI for all individual names $a \in \Sigma$: $d = a^{\mathcal{I}_1}$ iff $e = a^{\mathcal{I}_2}$.

3 Uniform Interpolation for Description Logic Ontologies

The central notion we are concerned with in this section is the following adaptation of Definition 1 to description logic ontologies.

▶ **Definition 3** (Uniform interpolant). *Let $\mathcal{L}$ be a DL, $\mathcal{O}$ an $\mathcal{L}$ ontology, and Σ a signature. Then, an $\mathcal{L}$ ontology $\mathcal{O}^\Sigma$ is called* uniform $\mathcal{L}(\Sigma)$-interpolant *of $\mathcal{O}$ if*
1. $\mathcal{O} \models \mathcal{O}^\Sigma$,
2. $\mathit{sig}(\mathcal{O}^\Sigma) \subseteq \Sigma$,
3. *for every $\mathcal{L}$ CI α such that $\mathit{sig}(\alpha) \subseteq \Sigma$ and $\mathcal{O} \models \alpha$, also $\mathcal{O}^\Sigma \models \alpha$.*

Conditions 1 and 3 imply that $\mathcal{O}$ and a uniform $\mathcal{L}(\Sigma)$-interpolant $\mathcal{O}^\Sigma$ of $\mathcal{O}$ entail precisely the same $\mathcal{L}(\Sigma)$ concept inclusions. This latter condition is called $\mathcal{L}(\Sigma)$-inseparability and of independent interest.

▶ **Definition 4** (Inseparability). *Let $\mathcal{O}_1$, $\mathcal{O}_2$ be $\mathcal{L}$ ontologies and Σ a signature. Then, $\mathcal{O}_1$ and $\mathcal{O}_2$ are* $\mathcal{L}(\Sigma)$-inseparable*, in symbols $\mathcal{O}_1 \equiv^{\mathcal{L}}_{\Sigma} \mathcal{O}_2$, if $\mathcal{O}_1 \models \alpha$ iff $\mathcal{O}_2 \models \alpha$ for every $\mathcal{L}(\Sigma)$ CI α.*

It is immediate from Definition 3 that uniform interpolants are unique modulo logical equivalence, which is why we often speak of *the* uniform $\mathcal{L}(\Sigma)$-interpolant of $\mathcal{O}$. In modal logic terminology, this form of interpolation could be classified as *turnstile* interpolation for global consequence. It is *uniform* in the sense that the interpolant has to work for all $\mathcal{L}(\Sigma)$ ontologies that are a consequence of $\mathcal{O}$; we refer the reader to Chapter 9 for more on uniform interpolation. To illustrate the notion, consider the following example [101, Example 2].

▶ **Example 5.** Consider the ontology $\mathcal{O}$ consisting of the following CIs:

$$\mathsf{Uni} \sqsubseteq \exists \mathsf{hasEnrolled}.\mathsf{Grad} \sqcap \exists \mathsf{hasEnrolled}.\mathsf{Undergrad} \tag{4}$$

$$\mathsf{Grad} \sqsubseteq \neg \mathsf{Undergrad} \tag{5}$$

$$\mathsf{Uni} \sqsubseteq \neg \mathsf{Grad} \tag{6}$$

$$\mathsf{Uni} \sqsubseteq \neg \mathsf{Undergrad} \tag{7}$$

Then, it can be verified that the ontology $\mathcal{O}'$ consisting of CI (7) and the CI

$$\mathsf{Uni} \sqsubseteq \exists \mathsf{hasEnrolled}.\mathsf{Undergrad} \sqcap \exists \mathsf{hasEnrolled}.(\neg \mathsf{UnderGrad} \sqcap \neg \mathsf{Uni})$$

is a uniform $\mathcal{ALC}(\Sigma)$-interpolant of $\mathcal{O}$ for $\Sigma = \{\mathsf{Uni}, \mathsf{hasEnrolled}, \mathsf{Undergrad}\}$. Sometimes $\mathcal{O}'$ is called the result of *forgetting* the concept name Grad in $\mathcal{O}$, since $\mathcal{O}'$ contains exactly the same information about the signature $\Sigma = \mathsf{sig}(\mathcal{O}) \setminus \{\mathsf{Grad}\}$. ⌟

Unfortunately, the existence of uniform interpolants is by no means guaranteed, as the following example illustrates already for very simple DL ontologies.

▶ **Example 6.** Consider the $\mathcal{EL}$ ontology $\mathcal{O} = \{A \sqsubseteq B,\ B \sqsubseteq \exists r.B\}$ and $\Sigma = \{A, r\}$. A uniform $\mathcal{EL}(\Sigma)$-interpolant would need to entail precisely the CIs

$$A \sqsubseteq \exists r.\top, \qquad A \sqsubseteq \exists r.\exists r.\top, \qquad A \sqsubseteq \exists r.\exists r.\exists r.\top, \qquad \dots$$

which is not possible for a finite set of $\mathcal{EL}$ or $\mathcal{ALC}$ CIs. (Actually, there is not even first-order sentence that entails precisely these CIs, c.f. Chapter 2 for uniform interpolation in first-order logic.) The reader may conjecture that the lack of a uniform interpolant is due to cyclicicty in $\mathcal{O}$, but this is not the entire story. Consider the $\mathcal{EL}$ ontology

$$\mathcal{O} = \{\ \ A \sqsubseteq \exists r.B, \qquad A_0 \sqsubseteq \exists r.(A_1 \sqcap B), \qquad E \equiv A_1 \sqcap B \sqcap \exists r.(A_2 \sqcap B)\ \ \}$$

and signature $\Sigma = \{A, A_0, A_1, r, E\}$. Reference [77] classifies $\mathcal{O}$ as Σ-*loop free* and shows that hence $\mathcal{O}$ has a uniform $\mathcal{EL}(\Sigma)$-interpolant. However, $\mathcal{O}$ fails to have a uniform $\mathcal{ALC}(\Sigma)$-interpolant [101, Example 3]. ⌟

As we have argued in the introduction, uniform interpolants are useful in a range of applications in KR. Since they do not always exist, it is interesting from a theoretical point of view to investigate existence as a decision problem. From a practical perspective, it is interesting to compute them (in case they exist), study their size, and to deal with the fact that they may not exist. We will address exactly these questions. More precisely, in Section 3.1, we discuss the complexity of the existence problem and the size of uniform interpolants. In Section 3.2, we discuss methods to compute uniform interpolants in practice, addressing also the question what to do if they do not exist. Finally, in Section 3.3, we discuss further related notions that are relevant to DL specific applications. Throughout the section, we focus on the DLs $\mathcal{EL}$ and $\mathcal{ALC}$, as these are the ones for which most results are known.

3.1 Existence and Size

As motivated in the previous section, the focus of this section will be the following *uniform interpolant existence problem* for DL $\mathcal{L}$:
Input $\mathcal{L}$ ontology $\mathcal{O}$, signature Σ.
Question Does there exist a uniform $\mathcal{L}(\Sigma)$-interpolant of $\mathcal{O}$?
The main result known here is that this problem is decidable for both $\mathcal{EL}$ and $\mathcal{ALC}$, but has complexity one exponential higher than standard reasoning tasks in these DLs.

▶ **Theorem 7** ([97, 101])**.** *Uniform interpolant existence is* ExpTime*-complete for* $\mathcal{EL}$ *and* 2ExpTime*-complete for* $\mathcal{ALC}$*.*

We are interested in the size of the uniform interpolants, if they exist, and tight bounds are known here as well. Interestingly, while the complexity of deciding existence is higher for $\mathcal{ALC}$ than for $\mathcal{EL}$, they exhibit the same bounds on the size of uniform interpolants.

▶ **Theorem 8** ([97, 101, 109]). *Let $\mathcal{L}$ be either $\mathcal{EL}$ or $\mathcal{ALC}$.*

1. *If some $\mathcal{L}$ ontology $\mathcal{O}$ has a uniform $\mathcal{L}(\Sigma)$-interpolant for signature Σ, then there is one of size at most triple exponential in the size of $\mathcal{O}$.*
2. *There is a family $\mathcal{O}_n$, $n \geq 1$ of $\mathcal{L}$ ontologies and a signature Σ such that the size of $\mathcal{O}_n$ is polynomial in n, a uniform $\mathcal{L}(\Sigma)$-interpolant exists for all $n \geq 1$, but any uniform $\mathcal{L}(\Sigma)$-interpolant has size at least triple exponential in n.*

It is beyond the scope of this chapter to present all proof details of these results, but we shall give some intuition on how they are obtained. We start with the size lower bounds, that is, Point 2 of Theorem 8, by recalling the family of $\mathcal{EL}$ ontologies from [109] that is used to prove this result. Let $n \geq 1$. Then $\mathcal{O}_n$ consists of the following CIs:

$$A_1 \sqsubseteq \overline{X_1} \sqcap \ldots \sqcap \overline{X_n} \qquad A_2 \sqsubseteq \overline{X_1} \sqcap \ldots \sqcap \overline{X_n} \tag{8}$$

$$\bigsqcap_{\sigma \in \{r,s\}} \exists \sigma.(\overline{X_i} \sqcap \overline{X_j}) \sqsubseteq \overline{X_i} \qquad \bigsqcap_{\sigma \in \{r,s\}} \exists \sigma.(X_i \sqcap \overline{X_j}) \sqsubseteq X_i \qquad 1 \leq j < i \leq n \tag{9}$$

$$\bigsqcap_{\sigma \in \{r,s\}} \exists \sigma.(\overline{X_i} \sqcap X_{i-1} \sqcap \ldots \sqcap X_1) \sqsubseteq X_i \qquad 1 \leq i \leq n \tag{10}$$

$$\bigsqcap_{\sigma \in \{r,s\}} \exists \sigma.(X_i \sqcap X_{i-1} \sqcap \ldots \sqcap X_1) \sqsubseteq \overline{X_i} \qquad 1 \leq i \leq n \tag{11}$$

$$X_0 \sqcap \ldots \sqcap X_n \sqsubseteq B \tag{12}$$

Intuitively, $\mathcal{O}_n$ constructs an n-bit binary counter via the concept names X_i, $\overline{X_i}$ for $1 \leq i \leq n$. The satisfaction of these names encodes a number between 0 and $2^n - 1$ using n bits: satisfaction of X_i means that the ith bit has value 1, and $\overline{X_i}$ represents a value of 0 at position i. Using this, elements can be assigned a counter value k. Specifically, the CIs in (8) make sure that instances of A_1 and A_2 have a counter value of 0 (all bits are 0). CIs (9)–(11) make sure that, if both an r and an s-successor of an element d has a counter value of k, then d has a counter value of $k + 1$. This is done by specifying how the bits on d should be set depending on the bits in the successors. Specifically, the CIs in (9) describe the situation where the bit value at position i should remain the same (at least one lower bit in the successor has value 0). CIs (10) and (11) describe that a bit at position i should flip if all lower bits in the successor have a value of 1. Finally, the CI (12) states that elements with a counter value of 2^{n-1} (all bits are 1) must satisfy B.

This construction has the following effect: any element d that is the root of a full binary r/s-tree of depth 2^n whose leaves satisfy A_1 or A_2, will be an instance of B. For $\Sigma = \{A_1, A_2, B, r, s\}$, any uniform Σ-interpolant will have to preserve this behaviour without using the counter encoded with the concept names $X_i, \overline{X}_i$. To construct the uniform $\mathcal{EL}(\Sigma)$-interpolant for $\mathcal{O}_n$, we define inductively sets $\mathcal{C}_i$ of concepts by setting $\mathcal{C}_0 = \{A_1, A_2\}$ and $\mathcal{C}_{i+1} = \{\exists r.C \sqcap \exists s.C \mid C \in \mathcal{C}_i\}$ for $i \geq 0$. It is not hard to see that the cardinality of each $\mathcal{C}_i$ is 2^{2^i}. Moreover, for each $C \in \mathcal{C}_{2^n}$, $\mathcal{O}_n \models C \sqsubseteq B$, and we cannot capture these entailments more concisely than by adding all those $C \sqsubseteq B$ to the uniform interpolant. It follows that the uniform interpolant contains at least $2^{2^{2^n}}$ CIs, and is thus of size triple exponential in n.

This finishes the proof sketch for $\mathcal{EL}$. The same construction does not directly yield a triple-exponential lower bound for $\mathcal{ALC}$, since we can here use the single concept $A_1 \sqcup A_2$ in case of the two concepts in $\mathcal{C}_0$, which reduces the size of the uniform interpolant to double exponential. The triple exponential lower bound on the size of uniform interpolants for $\mathcal{ALC}$ from [101, 50], is shown similarly, but relies on a 2^n-bit counter instead of an n-bit counter.

We now return to the existence problem, this time concentrating on $\mathcal{ALC}$. Let us fix an $\mathcal{ALC}$ ontology $\mathcal{O}$ and a signature Σ as input. We first give a syntactic characterization for the existence of a uniform interpolant. For $n \geq 0$, define $\mathcal{O}_n^\Sigma$ as the set of all $\mathcal{ALC}(\Sigma)$ CIs entailed by $\mathcal{O}$ whose role depth is at most n. If there exists a uniform interpolant, it must be equivalent to $\mathcal{O}_n^\Sigma$ for some n, since all its CIs would be contained in $\mathcal{O}_n^\Sigma$, and conversely $\mathcal{O} \models \mathcal{O}_n^\Sigma$ by construction. This means that non-existence of uniform interpolants implies that for every n, there exists some $k > n$ such that $\mathcal{O}_n^\Sigma \not\models \mathcal{O}_k^\Sigma$.

We characterize when some $\mathcal{O}_n^\Sigma$ is (not) a uniform interpolant using tree interpretations, which are interpretations $\mathcal{I}$ for which the relation $\bigcup\{r^\mathcal{I} \mid r \in \mathsf{N_R}\}$ forms a directed tree. The root of such an interpretation is then the root of that directed tree. For an integer m, we use $\mathcal{I}^m$ to refer to the interpretation obtained by removing all individuals which are not reachable from the root via a role path of length at most m. We then have the following lemma.

▶ **Lemma 9.** *Let $\mathcal{O}$ be an ontology, Σ a signature and $n > 0$. Then, $\mathcal{O}_n^\Sigma$ is not a uniform $\mathcal{ALC}(\Sigma)$-interpolant of $\mathcal{O}$ if there are tree interpretations $\mathcal{I}_1$, $\mathcal{I}_2$ with root d such that*

1. *$\mathcal{I}_1^n = \mathcal{I}_2^n$,*
2. *$\mathcal{I}_1, d \sim_{\mathcal{ALC},\Sigma} \mathcal{J}, d'$ for some model $\mathcal{J}$ of $\mathcal{O}$ and $d' \in \Delta^\mathcal{J}$,*
3. *$\mathcal{I}_2, d \not\sim_{\mathcal{ALC},\Sigma} \mathcal{J}, d'$ for all models $\mathcal{J}$ of $\mathcal{O}$ and $d' \in \Delta^\mathcal{J}$, and*
4. *for all role successors e of d, $\mathcal{I}_2, e \sim_{\mathcal{ALC},\Sigma} \mathcal{J}, d'$ for some model $\mathcal{J}$ of $\mathcal{O}$ and $d' \in \Delta^\mathcal{J}$.*

Intuitively, these conditions express that one has to look at a role depth beyond n to capture all Σ entailments of $\mathcal{O}$, or, equivalently, to capture all pointed interpretations that are Σ-bisimilar to some pointed model of $\mathcal{O}$. Together with Condition 2, Condition 1 ensures that the root individual in both pointed interpretations satisfies the same CIs up to role depth n, which means they satisfy all CIs in $\mathcal{O}_n^\Sigma$. However, $\mathcal{I}_2, d$ is not Σ-bisimilar to all pointed models of $\mathcal{O}$, while $\mathcal{I}_1$ is (Condition 3). Finally, Condition 4 states that the root is the point where the bisimulation breaks, since in $\mathcal{I}_2$, all successors of d are Σ-bisimilar to pointed models of $\mathcal{O}$ as required. In other words, $\mathcal{I}_2, d$ witnesses precisely that $\mathcal{O}_n^\Sigma$ is not a uniform Σ-interpolant of $\mathcal{O}$.

Set $M_\mathcal{O} = 2^{2 \cdot 2^{\|\mathcal{O}\|}} + 1$. We can show that if Lemma 9 applies for $n = M_\mathcal{O}$, then it also applies for all $n > M_\mathcal{O}$, meaning that then there does not exist a uniform $\mathcal{ALC}(\Sigma)$-interpolant. To do this, we use a construction based on types. Let Γ denote the set of all subconcepts that occur in $\mathcal{O}$, closed under single negation. Given an interpretation $\mathcal{I}$ and some individual $d \in \Delta^\mathcal{I}$, we then define the *type* $\mathsf{tp}_\mathcal{I}(d)$ *of* d *in* $\mathcal{I}$ as

$$\mathsf{tp}_\mathcal{I}(d) = \{C \in \Gamma \mid d \in C^\mathcal{I}\}.$$

To capture the types of elements that are bisimilar to d, we define the *extension set of* d as

$$\mathsf{Ext}(d, \mathcal{I}) = \{\mathsf{tp}_{\mathcal{J}}(d') \mid \mathcal{J} \models \mathcal{O}, d' \in \Delta^{\mathcal{J}} \text{ and } \mathcal{J}, d' \sim_{\Sigma} \mathcal{I}, d\}$$

There are at most $2^{2^{\|\mathcal{O}\|}}$ extension sets. Now assume Lemma 9 applies for $n = M_{\mathcal{O}}$, and let $\mathcal{I}_1$ and $\mathcal{I}_2$ be the witnessing tree interpretations. We have $\mathcal{I}_1^n = \mathcal{I}_2^n$, and now look at the successors in the leafs of $\mathcal{I}_1^n$, i.e. the elements in $\mathcal{I}_1^n$ without a role successor. Specifically, for such a pair, we define the set $D_n(\mathcal{I}_1, \mathcal{I}_2)$ to contain all leafs in $\mathcal{I}_1^n$ which differ in $\mathcal{I}_1^{n+1}$ and $\mathcal{I}_2^{n+1}$ in their sets of role successors, by which we mean that they either have a different set of role successors or some role successor that satisfies a different set of concept names in one interpretation compared to the other. Fix a pair $\mathcal{I}_1$, $\mathcal{I}_2$ of interpretations that satisfy the conditions in Lemma 9 for $n = M_{\mathcal{O}}$. If $D_n(\mathcal{I}_1, \mathcal{I}_2) = \emptyset$, then lemma also applies for $n = M_{\mathcal{O}} + 1$, and we are done. Otherwise, we pick some $f \in D_n(\mathcal{I}_1, \mathcal{I}_2)$, and consider the path $d = e_1, e_2, \ldots, e_n = f$ of elements such that, for $1 \leq i < n$, $\langle e_i, e_{i+1} \rangle \in r_i^{\mathcal{I}_1}$ for some role name r_i. Because $n = M_{\mathcal{O}} = 2^{2 \cdot 2^{\|\mathcal{O}\|}} + 1$, and there are at most $(2^{2^{\|\mathcal{O}\|}})^2 = 2^{2 \cdot 2^{\|\mathcal{O}\|}}$ pairs of extension sets, there must be some $1 \leq i < j \leq n$ such that $\mathsf{Ext}(e_i, \mathcal{I}_1) = \mathsf{Ext}(e_j, \mathcal{I}_1)$ and $\mathsf{Ext}(e_i, \mathcal{I}_2) = \mathsf{Ext}(e_j, \mathcal{I}_2)$. In both $\mathcal{I}_1$ and $\mathcal{I}_2$, we now replace the subtree below e_j with a copy of the subtree below e_i, resulting in two new interpretations $\mathcal{K}_1$ and $\mathcal{K}_2$. One can show that this construction preserves the conditions in Lemma 9, so that $\mathcal{K}_1$ and $\mathcal{K}_2$ also witness it. In addition, we have $D_n(\mathcal{K}_1, \mathcal{K}_2) \subsetneq D(\mathcal{I}_1, \mathcal{I}_2)$. We can repeat this operation until we obtain a pair $\mathcal{J}_1$, $\mathcal{J}_2$ of interpretations for which $D_n(\mathcal{J}_1, \mathcal{J}_2) = \emptyset$, and thus $\mathcal{J}_1^{n+1} = \mathcal{J}_2^{n+1}$, and obtain that Lemma 9 also holds for $n + 1$. This gives us the following lemma.

▶ **Lemma 10.** *There is a uniform $\mathcal{L}(\Sigma)$-interpolant of $\mathcal{O}$ iff $\mathcal{O}_n^{\Sigma} \models \mathcal{O}_{n+1}^{\Sigma}$ for $n = M_{\mathcal{O}}$.*

A consequence of lemma 10 is that if a uniform interpolant exists, then $\mathcal{O}_n^{\Sigma}$ is one, where $n = M_{\mathcal{O}}$. Thus, it gives a bound on the role depth of uniform interpolants. Since, up to equivalence, there are only finitely many $\mathcal{ALC}(\Sigma)$ CIs of bounded role depth, this also shows that the existence problem is decidable. Unfortunately, the bounds we get in this way are non-elementary. To establish the 2ExpTime-upper bound claimed in Theorem 7, [101] uses an automata-based approach to decide existence of uniform interpolants. The idea is to construct a tree automaton that decides the existence of interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ that satisfy the conditions in Lemma 9 for $n = M_{\mathcal{O}}$. In the same paper, they show the triple exponential upper bound on the size of uniform $\mathcal{ALC}$-interpolants claimed in Point 1 of Theorem 8. The proof is by a reduction of the problem of computing uniform interpolants of ontologies to the problem of computing uniform interpolants of *concepts* and apply known results for the latter [27]. For the complexity lower bounds, we refer the reader to [101] as well.

Interestingly, the strategy for deciding existence of uniform interpolants for $\mathcal{EL}$ is similar [97]: it relies on a characterization in the style of Lemma 9 (using simulations instead of bisimulations), and shows a bound on the role depth of uniform interpolants in style of Lemma 10 (which is single exponential in this case). Automata are then used as well, though in a different way, to obtain the complexity upper bounds claimed in Theorem 7. The size bounds for $\mathcal{EL}$ in Theorem 8 are shown in [109].

Table 2 Overview of tools for computing uniform interpolants of ontologies in practice.

Name of Tool	Logic	Technique	Dealing with Non-Existence
NUI [77]	$\mathcal{EL}$	TBox unfolding	–
- [94]	$\mathcal{ALC}$	resolution	approximation
LETHE [80]	$\mathcal{ALC}$	resolution + Ackermann's Lemma	fixpoints/auxiliary symbols
FAME [134]	$\mathcal{ALC}$	Ackermann's Lemma	fixpoints/auxiliary symbols

3.2 Computing Uniform Interpolants in Practice

Despite these high complexity bounds, algorithms for computing uniform interpolants have been described for many description logics, ranging from light-weight DLs such as DL-Lite [130] and $\mathcal{EL}$ [77, 95, 93] to $\mathcal{ALC}$ and more expressive DLs [131, 94, 80, 134, 79]. Many of these were implemented (see Table 2), and are able to compute results even for larger realistic ontologies. In this section, we discuss how implemented methods compute uniform interpolants of $\mathcal{ALC}$ ontologies, describing the method used by LETHE in detail.

The first practical issue is of course that uniform interpolants do not always exist. In many cases, a pragmatic solution could be to just extend the desired signature by the problematic symbols. For the case where this is not an option, there are three solutions, all of which are implemented by some tools.

Option 1: We approximate the uniform interpolant up to a given role depth, which might be sufficient if we are only interested in entailments of bounded role depth. The resolution based approach proposed in [94] provides guarantees on the role depth of preserved entailments—unfortunately these guarantees require an exponential increase in the role depth in the uniform interpolant.

Option 2: We extend the DL by *greatest fixpoint operators*. As argued above, for the ontology $\mathcal{O} = \{A \sqsubseteq B, B \sqsubseteq \exists r.B\}$, a uniform $\mathcal{ALC}(\{A, r\})$-interpolant does not exist. However, there does exist one in $\mathcal{ALC}\mu$: $\mathcal{O}_\Sigma = \{A \sqsubseteq \nu X.\exists r.X\}$. Indeed, $\mathcal{ALC}\mu$ does have the uniform interpolation property, which follows from the corresponding result for the modal μ-calculus [29]; see also Chapter 11. $\mathcal{ALC}\mu$ extends $\mathcal{ALC}$ by concepts of the form $\nu X.C[X]$, where X is taken from a set $\mathsf{N_V}$ of *concept variables* that is pair-wise disjoint with $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$, and $C[X]$ is a concept in which X is used like a concept name, but occurs only positively, that is, under an even number of negations. For a formal definition of the semantics, we refer to [23]. Intuitively, $\nu X.C[X]$ corresponds to the limit of the sequence

$$\top, \quad C[\top], \quad C[C[\top]], \quad C[C[C[\top]]], \ldots,$$

where in each case, $C[D]$ refers to the result of replacing X in $C[X]$ by D.

Option 3: Since fixpoint operators are harder to understand for end-users and also not supported by the OWL standard, which is the format used to store ontologies in practical applications, a third option is to simulate greatest fixpoint operators using auxiliary

concepts: In particular, we can replace any greatest fixpoint expression $\nu X.C[X]$, provided that it occurs only positively (e.g., under an even number of negations on the left-hand side of a CI, and under an odd number of negations on the right-hand side), by a fresh concept name D for which we add a new CI $D \sqsubseteq C[D]$. We then approximate the uniform Σ-interpolant *signature-wise*, resulting in an ontology that is Σ-inseparable from the original ontology and uses names outside of Σ in a syntactically restricted way.

Regardless of the option chosen, the general idea of practical uniform interpolation methods is to eliminate the names that should not occur in the interpolant using dedicated inferences. For this, we find two approaches: 1) using resolution [94] (as in Chapter 1 and Chapter 12)) or 2) using Ackermann's Lemma [1, 134] (also discussed in Chapter 12). The method in [80], which we present in the following, uses both in combination.

Because $\mathcal{ALC}$ is more complex than propositional logics, and syntactically more restricted than first-order logics, in order to pursue 1), we cannot use standard resolution methods, but need a specific resolution procedure that is appropriate for the logic and the aim of computing uniform interpolants. The first resolution-based approach for uniform interpolation in $\mathcal{ALC}$ was presented in [94] and is based on a complex inference system for modal logic introduced in [34] and extended for uniform interpolation in modal logics in [60]. This inference system intuitively specifies an unbounded number of inference rules that allow to perform resolution on symbols occurring in different nesting levels of role restrictions inside a CI. By increasing the bound on the nesting level on which inferences are performed, we can then approximate the uniform interpolant up to a specified role depth.

Ackermann's Lemma [1] has been extensively used in the context of Second-Order Quantifier Elimination. To formulate it for DLs, we use a stronger version of inseparability as introduced in Definition 4, in which instead of $\mathcal{L}$ concept inclusions arbitrary second-order logic (SOL) sentences are considered. We use $\mathcal{O}_1 \equiv_\Sigma^{\text{SOL}} \mathcal{O}_2$ to denote that $\mathcal{O}_1, \mathcal{O}_2$ entail the same SOL sentences over signature Σ. Let $\mathcal{O}[A \mapsto C]$ denote the ontology that is obtained from $\mathcal{O}$ by replacing every occurrence of A by C.

▶ **Lemma 11.** *Let $\mathcal{O}$ be an ontology, A a concept name, C a concept with $A \notin \mathsf{sig}(C)$, and $\Sigma = \mathsf{sig}(\mathcal{O}) \setminus \{A\}$. Assume A occurs only positively in $\mathcal{O}$. Then, $\mathcal{O} \cup \{A \sqsubseteq C\} \equiv_\Sigma^{SOL} \mathcal{O}[A \mapsto C]$.*

Intuitively, this means that, if we can bring the ontology into the right form, we can compute a very strong kind of uniform interpolants by simply applying the equivalence in Lemma 11 left-to-right. If A does occur on the right-hand side in $A \sqsubseteq C$, we can use a generalization of Ackermann's lemma from [110] that introduces greatest fixpoints. When restricted to DLs, it can be formulated as follows.[1]

[1] For the original FO formalization of both lemmas and their application to uniform interpolation, see also Chapter 12.

▶ **Lemma 12** (Generalized Ackermann's Lemma). *Let $\mathcal{O}$ be an ontology, A a concept name, $C[A]$ a concept in which A occurs, and $\Sigma = \mathsf{sig}(\mathcal{O}) \setminus \{A\}$. Assume A occurs only positively in $\mathcal{O}$ and $C[A]$. Then, the following holds:*

$$\mathcal{O} \cup \{A \sqsubseteq C[A]\} \equiv^{SOL}_{\Sigma} \mathcal{O}[A \mapsto \nu X.C[X]]$$

For both lemmas to be applicable, we have to bring the ontology into the required form. In particular, one has to isolate all negative occurrences of A into a single axiom of the form $A \sqsubseteq C$, which is not always possible. The approach in [134], implemented in the uniform interpolation tool FAME, essentially follows this strategy, and uses additional constructs such as inverse roles to make this possible, for instance, using the equivalence of axioms $C \sqsubseteq \forall r.D$ and $\exists r^{-}.C \sqsubseteq D$. The full method implemented in FAME is much more complex, and cannot only eliminate concept names, but also role names. The advantage of using (generalized) Ackermann's lemma is that the computed uniform interpolants are of a very strong kind, namely they are SOL(Σ)-inseparable. This is why the method implemented in FAME is also called *semantic forgetting*. Since SOL(Σ)-inseparability is undecidable (see Section 3.3), it is in general not possible to perform semantic forgetting to obtain an ontology expressed in a decidable logic, and indeed FAME cannot compute uniform interpolants for all possible inputs.

The technique in [80], implemented in the uniform interpolation tool LETHE, combines resolution with Ackermann's lemma and is based on a special normal form for $\mathcal{ALC}$ ontologies. In this normal form, axioms are represented as disjunctions $L_1 \sqcup \cdots \sqcup L_n$ of literals of the following forms, where $A \in \mathsf{N_C}$ and D belongs to special set $\mathsf{N_D} \subseteq \mathsf{N_C}$ of concept names called *definers*:

$$A \quad | \quad \neg A \quad | \quad \exists r.D \quad | \quad \forall r.D$$

Such disjunctions are called *DL clauses*, and are interpreted *globally*. For example, the concept inclusion $A \sqsubseteq B$ is represented as the DL clause $\neg A \sqcup B$. $\mathcal{ALC}$ ontologies $\mathcal{O}$ are normalized into $\mathsf{sig}(\mathcal{O})$-inseparable ontologies in this normal form through standard CNF transformation techniques and through the introduction of fresh concept names which we call *definers*. For instance, we would replace an axiom $A \sqsubseteq \exists r.(B \sqcup C)$ by two clauses $\neg A \sqcup \exists r.D$, $\neg D \sqcup B \sqcup C$, where D is an introduced definer. These definers play a central role in the method. We can normalize ontologies so that every clause contains at most one negative occurence of a definer. This invariant ensures that for every definer D, we can transform the clauses that contain $\neg D$ into a single GCI of the form $D \sqsubseteq C$, and then eliminate D again by using Ackermann's lemma or its generalization. The resolution procedure produces only clauses that also satisfy this invariant, which is why all definers can be eliminated in a final step.

The resolution method used by LETHE uses the following two inference rules:

$$\textbf{Resolution: } \frac{C_1 \sqcup A \quad C_2 \sqcup \neg A}{C_1 \sqcup C_2} \qquad \textbf{Role Propagation: } \frac{C_1 \sqcup \forall r.D_1 \qquad C_2 \sqcup \mathsf{Q}r.D_2}{C_1 \sqcup C_2 \sqcup \mathsf{Q}r.D_{12}}$$

where $\mathsf{Q} \in \{\exists, \forall\}$, and D_{12} is a possibly fresh definer that represents $D_1 \sqcap D_2$. We can introduce such a definer by adding the clauses $\neg D_{12} \sqcup D_1$, $\neg D_{12} \sqcup D_2$ and immediately

resolving on D_1 and D_2. In order to ensure termination, the method keeps track of introduced definers and reuses them where possible—for details we refer to [80].

The idea is now to perform all possible inferences on the concept names we want to eliminate, where we avoid inferences that would introduce a clause with two negative occurences of a definer. Because of this invariant, some resolution inferences become only possible through application of the role propagation rule, as illustrated in the following example. Once all inferences on a concept name to be eliminated have been applied, we can remove all occurences of that concept name.[2] Once we have eliminated all concept names outside of the signature Σ in this way, we eliminate all introduced definers using Ackermann's lemma and its generalization to obtain the uniform interpolant. To eliminate a role name r, we perform all possible inferences on r using role propagation. Afterwards, we can remove literals $\exists r.D$ for which D is unsatisfiable, and filter out the remaining occurrences of r.[3]

▶ **Example 13.** We want to compute the $\{A, B, D, E, r\}$-interpolant of the following ontology:

$$\mathcal{O} = \{ \quad A \sqsubseteq \exists r.(B \sqcap C) \qquad \exists r.(C \sqcap D) \sqsubseteq E \quad \}$$

The corresponding set of DL clauses is the following:

1. $\neg A \sqcup \exists r.D_1$ 2. $\neg D_1 \sqcup B$ 3. $\neg D_1 \sqcup C$ 4. $\forall r.D_2 \sqcup E$ 5. $\neg D_2 \sqcup \neg C \sqcup \neg D$

We need to eliminate C. We cannot resolve on Clauses 3 and 5, since it would produce a clause with two negative occurrences of a definer, which would break our invariant. To make resolution on C possible, we need to first apply role propagation on Clauses 1 and 4, followed by an immediate resolution on D_1 and D_2:

6. $\neg A \sqcup E \sqcup \exists r.D_{12}$ 7. $\neg D_{12} \sqcup D_1$ 8. $\neg D_{12} \sqcup D_2$
9. $\neg D_{12} \sqcup B$ 10. $\neg D_{12} \sqcup C$ 11. $\neg D_{12} \sqcup \neg C \sqcup \neg D$

Clauses 7 and 8 are only intermediate inferences and can be discarded. We can now resolve upon C, the name to be eliminated, namely on Clauses 10 and 11:

12. $\neg D_{12} \sqcup \neg D$

No further resolution steps on C are possible without breaking the invariant that every clause contains at most one negative occurence of a definer. The procedure would now remove all clauses containing C (Clause 3, 5, 10 and 11) and apply Ackermann's lemma to eliminate the

---

[2] This is essentially the same idea as in the resolution-based approach for propositional logic discussed in Chapter 1, or in the second-order quantifier elimination tool SCAN [111] described in Chapter 12.

[3] [83] discusses a more involved method for role elimination. Both methods are implemented in the current version of LETHE [81, 80].

introduced names D_1, D_2 and D_{12} again. The resulting set of DL clauses can be represented as the following set of CIs, which is the desired uniform interpolant:

$$\{ \quad A \sqsubseteq \exists r.B, \qquad A \sqcap \forall r.(\neg B \sqcup D) \sqsubseteq E \quad \} \qquad \lrcorner$$

Note that while this approach will always compute a uniform interpolant in $\mathcal{ALC}\mu$, it may not find a uniform interpolant without fixpoints even if one exists. As argued above, in practice, one can keep the problematic definers, or compute approximations of the fixpoint expressions. A practical method that is able to decide the existence of uniform interpolants without fixpoints on realistic ontologies has not been found so far.

3.3 Related Notions and Applications

We close this section by discussing four relevant notions closely related to uniform interpolation: inseparability and conservative extensions, logical difference, and modules.

Inseparability and Conservative Extensions Inseparability as in Definition 4 is a key property that is desired in many applications. For instance, when an ontology evolves by adding, removing, or changing concept inclusions, we might want to ensure that the change does not affect entailments in a certain signature of interest. Since inseparability is built-in into uniform interpolation, there are natural connections to these applications; we discuss some of them.

Very closely related to uniform interpolation is the notion of conservative extensions. Conservative extensions are a classical notion studied in logic which has been introduced to KR in the seminal work [50]. Let $\mathcal{L}$ be a DL and $\mathcal{O}_1$, $\mathcal{O}_2$ be two $\mathcal{L}$ ontologies. Recall that we write $\mathcal{O}_1 \equiv^{\mathcal{L}}_{\Sigma} \mathcal{O}_2$ to indicate that $\mathcal{O}_1$ and $\mathcal{O}_2$ entail precisely the same concept inclusions that can be expressed in $\mathcal{L}$ using signature Σ. We call $\mathcal{O}_2$ a *conservative extension* of $\mathcal{O}_1$ if $\mathcal{O}_1 \subseteq \mathcal{O}_2$ and $\mathcal{O}_1 \equiv^{\mathcal{L}}_{\mathsf{sig}(\mathcal{O}_1)} \mathcal{O}_2$. Conservative extensions are relevant when we want to extend an ontology with new concept inclusions about new symbols, while preserving the behaviour with respect to the original signature. Hence, the central problem is to decide given $\mathcal{O}_1, \mathcal{O}_2$ whether $\mathcal{O}_2$ is a conservative extension of $\mathcal{O}_1$. As pointed out in [50], it is immediate from the definitions that

$$\mathcal{O}_2 \text{ is a conservative extension of } \mathcal{O}_1 \text{ iff } \mathcal{O}_1 \text{ is a uniform } \mathcal{L}(\mathsf{sig}(\mathcal{O}_1))\text{-interpolant of } \mathcal{O}_2. \quad (13)$$

This can in some cases be exploited to decide conservative extensions. Indeed, if $\mathcal{O}_2$ has a uniform $\mathcal{L}(\mathsf{sig}(\mathcal{O}_1))$-interpolant, then we can compute it and check equivalence with $\mathcal{O}_1$. As discussed before, this is however not always possible since uniform interpolants may not exist. In general, even when not only adding concept inclusions, we may want to decide whether two ontologies are inseparable. As for conservative extensions, uniform interpolants could help also in this case. Indeed, if $\mathcal{O}_1^{\Sigma}$ and $\mathcal{O}_2^{\Sigma}$ are uniform $\mathcal{L}(\Sigma)$-interpolants of $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively, then $\mathcal{O}_1 \equiv^{\mathcal{L}}_{\Sigma} \mathcal{O}_2$ iff $\mathcal{O}_1^{\Sigma} \equiv \mathcal{O}_2^{\Sigma}$, that is, $\mathcal{O}_1^{\Sigma} \models \mathcal{O}_2^{\Sigma}$ and $\mathcal{O}_2^{\Sigma} \models \mathcal{O}_1^{\Sigma}$.

Table 3 The complexity of uniform interpolant recognition.

Ontology Language	Complexity
$\mathcal{EL}$	ExpTime [100]
$\mathcal{ALC}$, $\mathcal{ALCI}$	2ExpTime [50, 99]
$\mathcal{ALCO}$	3ExpTime-hard [69]
$\mathcal{ALCFIO}$	undecidable [99]

Both conservative extensions and inseparability have been heavily studied for description logics, see for example [50, 99, 20, 68, 100, 69]. The complexity results obtained there have implications also on the side of uniform interpolation, especially for the recognition problem. Here, *uniform interpolant recognition* is the problem of deciding whether an ontology $\mathcal{O}^\Sigma$ is a uniform $\mathcal{L}(\Sigma)$-interpolant of another ontology $\mathcal{O}$. The recognition problem has not been explicitly studied in the literature, but it might be relevant for cases when the ontology engineer has a candidate ontology in mind and wants to verify whether it is indeed a uniform interpolant. Table 3 summarizes some complexity results for the recognition problem. The lower bounds are inherited from deciding conservative extensions via the reduction (13). For the upper bounds, observe that uniform interpolant recognition reduces to entailment and inseparability as follows:

$$\mathcal{O}^\Sigma \text{ is a uniform } \mathcal{L}(\Sigma)\text{-interpolant of } \mathcal{O} \text{ iff } \mathcal{O} \models \mathcal{O}^\Sigma \text{ and } \mathcal{O} \equiv^{\mathcal{L}}_{\Sigma} \mathcal{O}^\Sigma.$$

It is worth noting that nominals are particularly challenging in this context, and we will see in Section 4 that this is also the case for interpolation at the level of concepts.

We point out that in applications related to query answering, we need alternative notions of inseparability (and thus uniform interpolation), namely inseparability *by queries*. As this is beyond the scope of this chapter, we refer the reader to [20] for a survey.

Logical Difference In case two ontologies are not $\mathcal{L}(\Sigma)$-inseparable, a natural question is in which $\mathcal{L}(\Sigma)$-entailments they actually differ, which is captured by the logical difference of the ontologies:

▶ **Definition 14** (Logical Difference). *Let $\mathcal{O}_1$, $\mathcal{O}_2$ be two ontologies and Σ be a signature. The logical difference from $\mathcal{O}_1$ to $\mathcal{O}_2$ in Σ is defined as*

$$\mathit{diff}(\mathcal{O}_1, \mathcal{O}_2, \Sigma) = \{\alpha \mid \mathit{sig}(\alpha) \subseteq \Sigma, \mathcal{O}_1 \models \alpha, \mathcal{O}_2 \not\models \alpha\}$$

Logical difference is particularly useful to track changes between different versions of an ontology. In contrast to a purely syntactic check (which CIs have been added/removed/changed), logical difference provides a semantic way to understand whether entailments in a signature of interest have changed. Note that if the logical difference between two ontologies is non-empty, then it contains infinitely many concept inclusions. Nonetheless, it is often possible to compute

finite representations [76]. For more expressive logics, the only existing tools compute finite representations of the logical difference by reduction to uniform interpolation [94, 80, 93].

Modules As discussed in the introduction, one key application of uniform interpolation is ontology reuse. Modules are an alternative solution for this. Modules work the other way around to conservative extensions: given an ontology $\mathcal{O}$ and a signature Σ, a $\mathcal{L}(\Sigma)$-*module of* $\mathcal{O}$ is an $\mathcal{L}(\Sigma)$-inseparable subset of $\mathcal{O}$. Modules are investigated in detail in [58, 74, 75], and are relevant for the application *Modularisation and Reuse* discussed in the introduction, but they have also been used for other purposes such as to improve reasoner performance [116], or as preprocessing step for uniform interpolant computation [94, 80, 134]. Differently to uniform interpolants, the size of a module is bounded by the size of the original ontology, and the syntactical structure of CIs is not changed, which is sometimes relevant. At the same time, uniform interpolants can often be more compact than modules, and give guarantees on the used signature. Practical methods for uniform interpolation have been used to compute subset-minimal $\mathcal{EL}(\Sigma)$ and $\mathcal{ALC}(\Sigma)$-modules [81, 133], but there are also very fast syntactical methods that compute SOL(Σ)-modules [58], which may however not be subset-minimal. A compromise between modules and uniform interpolants that can leverage the advantages of both are *generalised modules*. A generalised $\mathcal{L}(\Sigma)$-module of an ontology $\mathcal{O}$ can be just any ontology that is $\mathcal{L}(\Sigma)$-inseparable with $\mathcal{O}$, but is ideally smaller and simpler than $\mathcal{O}$. General modules can be significantly simpler than both uniform interpolants and modules [108, 132]. The method in [132] uses a technique similar to that of Lethe discussed in this chapter.

4 Craig Interpolation for Description Logic Concepts

In this section, we will define and discuss interpolation in description logics at the level of concepts rather than at the level of ontologies as in preceding section. We start in Section 4.1 with (Craig) interpolation and then move in Section 4.2 to the tightly related concept of Beth definability. In Section 4.3, we discuss the main applications of interpolants and explicit definitions which motivate the necessity of computing them. The computation problem is then covered in Section 4.4.

4.1 Craig Interpolation

We use the following standard definition of interpolants in the context of DL ontologies.

▶ **Definition 15** (Interpolants). *Let $\mathcal{L}$ be any DL. Let C_1, C_2 be $\mathcal{L}$ concepts, $\mathcal{O}$ an $\mathcal{L}$ ontology and Σ be a signature. Then, an* $\mathcal{L}(\Sigma)$-interpolant for C_1 and C_2 under $\mathcal{O}$ *is any* $\mathcal{L}(\Sigma)$ *concept I that satisfies $\mathcal{O} \models C_1 \sqsubseteq I$ and $\mathcal{O} \models I \sqsubseteq C_2$.*

We remark that this definition of interpolants is in line with the one used for modal logic in Chapter 4. More precisely, with an empty ontology $\mathcal{O}$, an interpolant for C_1 and C_2 is an interpolant for the validity $\models C_1 \rightarrow C_2$, when viewing C_1, C_2 as modal logic formulas.

The case with ontologies is also related to the setting of interpolation *relative to theories* in first-order logic, see Chapter 2.

A classical application of interpolants in DLs is in explaining subsumption relations. More precisely, the idea put forward in [122] is that an $\mathcal{L}(\Sigma)$-interpolant for C and D with $\models C \sqsubseteq D$ for a *minimal* Σ is a good explanation of the possibly complicated subsumption $C \sqsubseteq D$. The following example illustrates the idea.

▶ **Example 16.** Consider $C = \exists \mathsf{child}.\top \sqcap \forall \mathsf{child}.\mathsf{Doctor}$ and $D = \exists \mathsf{child}.(\mathsf{Doctor} \sqcup \mathsf{Rich})$. Clearly, $\models C \sqsubseteq D$ and $\exists \mathsf{child}.\mathsf{Doctor}$ is an $\mathcal{ALC}(\{\mathsf{child}, \mathsf{Doctor}\})$-interpolant of C and D. Moreover, one can show that there is no $\mathcal{ALC}(\Sigma)$-interpolant of C and D for $\Sigma \subsetneq \{\mathsf{child}, \mathsf{Doctor}\}$. Hence, $\exists \mathsf{child}.\mathsf{Doctor}$ is an arguably simple explanation of the subsumption. ⌟

This application motivates the study of the following *$\mathcal{L}$-interpolant existence* problem:
Input $\mathcal{L}$ ontology $\mathcal{O}$, $\mathcal{L}$ concepts C_1, C_2 and a signature Σ.
Question Does there exist an $\mathcal{L}(\Sigma)$-interpolant for C_1 and C_2 under $\mathcal{O}$?
We will see that this potentially difficult existence problem has a surprisingly simple solution for DLs that satisfy the *Craig interpolation property* (CIP), defined following [28, 8]. Recall that $\mathsf{sig}(X)$ denotes the set of concept and role names used in object X; we use $\mathsf{sig}(\mathcal{O}, C)$ to abbreviate $\mathsf{sig}(\mathcal{O}) \cup \mathsf{sig}(C)$.

▶ **Definition 17** (Craig Interpolation Property)**.** *Let $\mathcal{L}$ be any DL. We say that $\mathcal{L}$ enjoys the* Craig interpolation property (CIP) *if for every $\mathcal{L}$ concepts C_1, C_2 and $\mathcal{L}$ ontologies $\mathcal{O}_1, \mathcal{O}_2$ with $\mathcal{O}_1 \cup \mathcal{O}_2 \models C_1 \sqsubseteq C_2$, there exists a* Craig interpolant*, that is, an $\mathcal{L}(\Sigma)$-interpolant of C_1 and C_2 under $\mathcal{O}_1 \cup \mathcal{O}_2$ where $\Sigma = \mathit{sig}(\mathcal{O}_1, C_1) \cap \mathit{sig}(\mathcal{O}_2, C_2)$.*

In the same way as the definition of interpolants, the definition of the CIP under empty ontologies $\mathcal{O}_1 = \mathcal{O}_2 = \emptyset$ coincides with the standard definition of the CIP in modal logic. The split of the ontology into two parts is necessary to ensure the existence of Craig interpolants in the presence of ontologies. Indeed, consider the following example from [8]. Let $\mathcal{O} = \{A_1 \sqsubseteq A_2, A_2 \sqsubseteq A_3\}$, and consider concepts $C_1 = A_1$ and $C_2 = A_3$. Then for every $\mathcal{O}_1, \mathcal{O}_2$ with $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$, there is an $\mathcal{ALC}(\Sigma)$-interpolant for C_1 and C_2 under $\mathcal{O}$ with $\Sigma = \mathsf{sig}(\mathcal{O}_1, C_1) \cap \mathsf{sig}(\mathcal{O}_2, C_2)$. However, there is no $\mathcal{ALC}(\Sigma_0)$-interpolant for C_1 and C_2 under $\mathcal{O}$ with $\Sigma_0 = \mathsf{sig}(C_1) \cap \mathsf{sig}(C_2) = \emptyset$.

The Craig interpolation property is so powerful since it *guarantees* the existence of interpolants/explanations in terms of the common signature $\Sigma = \mathsf{sig}(\mathcal{O}_1, C_1) \cap \mathsf{sig}(\mathcal{O}_2, C_2)$. Fortunately, many DLs enjoy the CIP, also in the presence of ontologies [28].

▶ **Theorem 18.** *$\mathcal{ALC}$ and any extension with $\mathcal{S}$, $\mathcal{I}$, $\mathcal{F}$ enjoys the Craig interpolation property.*

The proof of Theorem 18 in [28] is based on a tableau algorithm for subsumption and is constructive in the sense that it also computes a Craig interpolant in case the subsumption holds. We will next provide a shorter (though not constructive) model-theoretic proof, which bears a lot of similarity with the model-theoretic proof for the CIP in modal logic in Chapter 4.

The main difference is that we have to accomodate the presence of an ontology. The first step of this proof is to provide a model-theoretic characterization for interpolant existence in terms of bisimulations, inspired by early works by Robinson [115]. This is a rather uniform step that can be adapted to virtually every (description) logic by "plugging in" the respective bisimulation notion capturing the expressive power of the logic. The second step is then an amalgamation lemma which shows that the model-theoretic condition is satisfied whenever the subsumption holds. This second step does not work for all logics, and we will see that failure of this step for some logic usually means that this logic does not enjoy the CIP.

We introduce the necessary notation. Let $\mathcal{L}$ be some description logic. Let C_1, C_2 be $\mathcal{L}$ concepts, $\mathcal{O}$ be an $\mathcal{L}$ ontology, and Σ be a signature. Then C_1, C_2 are called *jointly* $\sim_{\mathcal{L},\Sigma}$*-consistent under* $\mathcal{O}$ if there exist models $\mathcal{I}_1, \mathcal{I}_2$ of $\mathcal{O}$ and elements $d_i \in C_i^{\mathcal{I}_i}$ for $i = 1, 2$, satisfying $\mathcal{I}_1, d_1 \sim_{\mathcal{L},\Sigma} \mathcal{I}_2, d_2$.[4] Joint consistency can be thought of as a strong form of satisfiability of two concepts with an additional bisimilarity requirement. The following lemma characterizes the existence of interpolants in terms of joint consistency. The proof is rather standard and relies on the fact that $\mathcal{L}$-bisimulations capture the expressive power of $\mathcal{L}$ and crucially also on compactness, see Chapter 4 for a similar lemma, but also [56].

▶ **Lemma 19.** *Let C_1, C_2 be $\mathcal{L}$ concepts, $\mathcal{O}$ an $\mathcal{L}$ ontology, and Σ be a signature. Then the following conditions are equivalent:*

1. *there is no $\mathcal{L}(\Sigma)$-interpolant for C_1 and C_2 under $\mathcal{O}$;*
2. *C_1 and $\neg C_2$ are jointly $\sim_{\mathcal{L},\Sigma}$-consistent under $\mathcal{O}$.*

Intuitively, for an $\mathcal{L}(\Sigma)$-interpolant for C_1 and C_2 to exist, the inconsistency of C_1 and $\neg C_2$ must be detectable by an $\mathcal{L}(\Sigma)$-bisimulation, since otherwise we cannot express it using an $\mathcal{L}(\Sigma)$ concept. Hence, in order to show the CIP for a DL, it suffices to show that a witness for Point 2 can be turned into a witness for failure of the subsumption $\mathcal{O} \models C_1 \sqsubseteq C_2$. This is the content of the following amalgamation lemma, which is stated and proved for $\mathcal{ALC}$ only, for the sake of simplicity. Its proof is similar to the proof of an analogous amalgamation lemma for modal logic in Chapter 4. We refer the reader to Chapter 8 for more on amalgamation.

▶ **Lemma 20.** *Let $\mathcal{I}_1, \mathcal{I}_2$ be models of an $\mathcal{ALC}$ ontology $\mathcal{O}_1 \cup \mathcal{O}_2$ and suppose $\mathcal{I}_1, d_1 \sim_{\mathcal{ALC},\Sigma_1 \cap \Sigma_2} \mathcal{I}_2, d_2$ for $d_1 \in \Delta^{\mathcal{I}_1}, d_2 \in \Delta^{\mathcal{I}_2}$ and signatures $\Sigma_1 \supseteq \mathsf{sig}(\mathcal{O}_1), \Sigma_2 \supseteq \mathsf{sig}(\mathcal{O}_2)$. Then there is a model $\mathcal{J}$ of $\mathcal{O}_1 \cup \mathcal{O}_2$ and $e \in \Delta^{\mathcal{J}}$ such that $\mathcal{J}, e \sim_{\mathcal{ALC},\Sigma_1} \mathcal{I}_1, d_1$ and $\mathcal{J}, e \sim_{\mathcal{ALC},\Sigma_2} \mathcal{I}_2, d_2$.*

As announced, Lemmas 19 and 20 can be used to prove that $\mathcal{ALC}$ enjoys the CIP.

Proof of Theorem 18 for $\mathcal{ALC}$. Suppose $\mathcal{O}_1 \cup \mathcal{O}_2 \models C_1 \sqsubseteq C_2$ for $\mathcal{ALC}$ ontologies $\mathcal{O}_1, \mathcal{O}_2$ and $\mathcal{ALC}$ concepts C_1, C_2 with $\Sigma_1 = \mathsf{sig}(\mathcal{O}_1, C_1)$ and $\Sigma_2 = \mathsf{sig}(\mathcal{O}_2, C_2)$. If there is no Craig interpolant for C_1 and C_2 under $\mathcal{O}_1 \cup \mathcal{O}_2$, then by Lemma 19, there are models $\mathcal{I}_1, \mathcal{I}_2$ of

[4] It is worth noting that *joint $\sim_{\mathcal{L},\Sigma}$-consistency* is dual to the notion of *entailment along bisimulations* in Chapter 4. Indeed, C_1 and $\neg C_2$ are jointly $\sim_{\mathcal{L},\Sigma}$-consistent iff C_1 does not entail C_2 along $\mathcal{L}(\Sigma)$-bisimulations. We stick to joint bisimilarity since it has been the term recently used in the DL community.

$\mathcal{O}_1 \cup \mathcal{O}_2$ and $d_1 \in \Delta^{\mathcal{I}_1}, d_2 \in \Delta^{\mathcal{I}_2}$ such that $d_1 \in C_1^{\mathcal{I}_1}$, $d_2 \notin C_2^{\mathcal{I}_2}$, and $\mathcal{I}_1, d_1 \sim_{\mathcal{ALC},\Sigma_1\cap\Sigma_2} \mathcal{I}_2, d_2$. By Lemma 20, there is a model $\mathcal{J}$ of $\mathcal{O}_1 \cup \mathcal{O}_2$ and $e \in \Delta^{\mathcal{J}}$ such that $\mathcal{J}, e \sim_{\mathcal{ALC},\Sigma_1} \mathcal{I}_1, d_1$ and $\mathcal{J}, e \sim_{\mathcal{ALC},\Sigma_2} \mathcal{I}_2, d_2$. Now, since $d_1 \in C_1^{\mathcal{I}_1}$, $\mathsf{sig}(C_1) \subseteq \Sigma_1$, and $\mathcal{J}, e \sim_{\mathcal{ALC},\Sigma_1} \mathcal{I}_1, d_1$, we also have $e \in C_1^{\mathcal{J}}$. Analogously, one can show that $d_2 \notin C_2^{\mathcal{J}}$. This is in contradiction to the assumed subsumption $\mathcal{O}_1 \cup \mathcal{O}_2 \models C_1 \sqsubseteq C_2$. ◀

We come back to the problem of $\mathcal{L}$-interpolant existence. As mentioned above, the problem trivializes for DLs that enjoy the CIP if we ask for interpolants in the common signature. Moreover, it is not difficult to reduce the existence of $\mathcal{L}(\Sigma)$-interpolants for given signature Σ to the existence of Craig interpolants. Indeed, one can verify that, for any standard DL $\mathcal{L}$, the following are equivalent for all $\mathcal{L}$ concepts C_1, C_2, $\mathcal{L}$ ontologies $\mathcal{O}$, signatures Σ, and $C_{2\Sigma}$ and $\mathcal{O}_\Sigma$ obtained from C_2 and $\mathcal{O}$, respectively, by renaming all symbols not in Σ uniformly to fresh symbols:

- there is an $\mathcal{L}(\Sigma)$-interpolant for C_1 and C_2 under $\mathcal{O}$;
- there is a Craig interpolant for C_1 and $C_{2\Sigma}$ under $\mathcal{O} \cup \mathcal{O}_\Sigma$.

Since existence of Craig interpolants coincides with the respective subsumption relationship in DLs enjoying the CIP, we have reduced $\mathcal{L}$-interpolant existence to subsumption checking for such DLs. A reduction of subsumption to interpolant existence is also possible. Since subsumption in all DLs mentioned in Theorem 18 is ExpTime-complete [124], we obtain:

▶ **Theorem 21.** *For any DL $\mathcal{L}$ in Theorem 18, $\mathcal{L}$-interpolant existence is* ExpTime*-complete.*

It is worth noting that the proof of Theorem 18 (and thus of Theorem 21) is not constructive since the proof of the model-theoretic characterization in Lemma 19 relies on compactness and does not provide an interpolant if one exists. We will address the computation problem in Section 4.4.

Here, we will turn our attention to the fact that, unfortunately, not all DLs enjoy the CIP.

▶ **Theorem 22** ([28, 8])**.** *$\mathcal{ALCO}$ and $\mathcal{ALCH}$ and their extensions by $\mathcal{S}, \mathcal{I}, \mathcal{F}$ do not enjoy the Craig interpolation property.*

Proof. We show the argument only for $\mathcal{ALCO}$, for $\mathcal{ALCH}$ and their extensions see [28, 8]. Consider $\mathcal{ALCO}$ concepts $C_1 = \{a\} \sqcap \exists r.\{a\}$ and $C_2 = A \to \exists r.A$ with common signature $\{r\}$. Then $\models C_1 \sqsubseteq C_2$, but there is no $\mathcal{ALCO}(\{r\})$-interpolant for C_1 and C_2. Indeed, the interpretations in Figure 1 witness that C_1 and $\neg C_2$ are jointly $\sim_{\mathcal{ALCO},\{r\}}$-consistent, and by Lemma 19 there is no Craig interpolant. ◀

One consequence of the lack of the CIP is that the reduction of interpolant existence to subsumption checking used to prove Theorem 21 does not work anymore. Fortunately, it turns out that the problem is still decidable, although typically harder than subsumption. We exploit that Lemma 19 provides a reduction of interpolant existence to the problem of deciding joint consistency under ontologies, so it suffices to decide the latter. For didactic purposes and since we need it later, we provide first a relatively simple algorithm for deciding joint

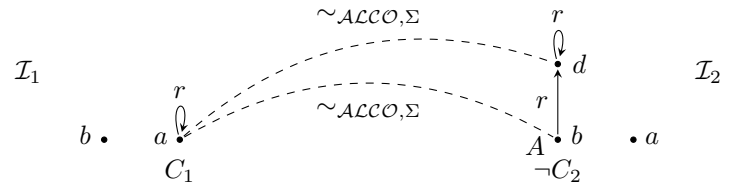


Figure 1 Interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ illustrating the proof of Theorem 22.

consistency in $\mathcal{ALC}$, which is inspired by standard type elimination algorithms for deciding satisfiability in DLs [11] and is closely related to the type elimination sequences in Chapter 4.

To describe the algorithm deciding joint consistency, let us fix an $\mathcal{ALC}$ ontology $\mathcal{O}$, $\mathcal{ALC}$ concepts C_1, C_2, and a signature Σ. Let Γ denote the set of all subconcepts that occur in $\mathcal{O}$, C_1, C_2, closed under single negation. A *type* is any set $t \subseteq \Gamma$ such that there is a model $\mathcal{I}$ of $\mathcal{O}$ and an element $d \in \Delta^{\mathcal{I}}$ such that $t = \mathsf{tp}_{\mathcal{I}}(d)$ where, as in Section 3.1,

$$\mathsf{tp}_{\mathcal{I}}(d) = \{C \in \Gamma \mid d \in C^{\mathcal{I}}\}.$$

A *mosaic* m is a pair $m = (t_1, t_2)$ of types. Intuitively, a mosaic (t_1, t_2) provides an abstract description of two elements d_1, d_2 in two models $\mathcal{I}_1, \mathcal{I}_2$ of $\mathcal{O}$ which have types t_1, t_2 and are $\mathcal{ALC}(\Sigma)$-bisimilar. Of course, not all mosaics are realizable in this sense and the goal of the elimination is to identify those that are.

We write $t \rightsquigarrow_r t'$ if an element of type t' is a viable r-successor of an element of type t, that is, $\{C \mid \forall r.C \in t\} \subseteq t'$. We use $(t_1, t_2) \rightsquigarrow_r (t_1', t_2')$ to abbreviate $t_1 \rightsquigarrow_r t_1'$, $t_2 \rightsquigarrow_r t_2'$. Let $\mathcal{M}$ be a set of mosaics. We call $(t_1, t_2) \in \mathcal{M}$ *bad* if it violates one of the following conditions:

(Atomic Consistency) for every $A \in \Sigma$, $A \in t_1$ iff $A \in t_2$;

(Existential Saturation) for every $r \in \Sigma$, every $i = 1, 2$, and every $\exists r.C \in t_i$, there is $(t_1', t_2') \in \mathcal{M}$ such that $C \in t_i'$ and $(t_1, t_2) \rightsquigarrow_r (t_1', t_2')$.

Clearly, a mosaic (t_1, t_2) violating atomic consistency cannot be realized as required, due to concept name $A \in \Sigma$. Similarly, a mosaic violating existential saturation cannot be realized since it lacks a viable r-successor for one of the t_i. We have the following characterization of joint consistency:

▶ **Lemma 23.** *C_1 and C_2 are jointly $\sim_{\mathcal{ALC},\Sigma}$-consistent under $\mathcal{O}$ iff there is a set $\mathcal{M}^*$ of mosaics that does not contain bad mosaics and some $(t_1, t_2) \in \mathcal{M}^*$ with $C_1 \in t_1$ and $C_2 \in t_2$.*

The existence of an $\mathcal{M}^*$ as in Lemma 23 can be decided by a simple mosaic elimination algorithm. The idea is to compute a sequence of sets of mosaics

$$\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \ldots$$

where $\mathcal{M}_0$ is the set of all mosaics, and for $i \geq 0$, $\mathcal{M}_{i+1}$ is obtained from $\mathcal{M}_i$ by eliminating all bad mosaics from $\mathcal{M}_i$. Since $\mathcal{M}_0$ is finite, this process reaches a fixpoint after finitely

many steps. Then, C_1 and C_2 are jointly $\sim_{\mathcal{ALC},\Sigma}$-consistent under $\mathcal{O}$ iff the fixpoint $\mathcal{M}^*$ of that process contains (t_1, t_2) as in Lemma 23. The algorithm runs in exponential time in the size of the input, since the number of mosaics is at most exponential in the size of the input, and each mosaic can be checked for badness in time polynomial in the number of mosaics. Thus, the approach to interpolant existence via deciding joint consistency is not worse than the sketched reduction to subsumption checking. We will see in Section 4.4 how to extend it to compute interpolants as well.

This idea can be generalized to decide $\mathcal{L}$-interpolant existence for DLs lacking the CIP.

▶ **Theorem 24** ([8]). *For $\mathcal{L} \in \{\mathcal{ALCO}, \mathcal{ALCH}, \mathcal{ALCOI}, \mathcal{ALCHI}\}$, $\mathcal{L}$-interpolant existence is* 2ExpTime*-complete.*

Due to Lemma 19, it suffices to show the results for joint consistency. We sketch it here for $\mathcal{ALCO}$, see [8] for a full proof and the other DLs covered in the theorem. To get an intuition for the lower bound proof, it is instructive to look again at the proof of Theorem 22 and particularly Figure 1, where the elements b, d in $\mathcal{I}_2$ are *forced* to be bisimilar, as they are both bisimilar to a in $\mathcal{I}_1$. This idea is extended to force *exponentially many* elements to be bisimilar in witnesses for joint $\sim_{\mathcal{L},\Sigma}$-consistency, which in turn is exploited to synchronize configurations of exponentially space bounded alternating Turing machines.

For the upper bound we extend the mosaic elimination algorithm that was used to decide joint consistency for $\mathcal{ALC}$ to $\mathcal{ALCO}$. Let $\mathcal{O}$ be an $\mathcal{ALCO}$ ontology, C_1, C_2 be $\mathcal{ALCO}$ concepts, and Σ be a signature. Let Γ be again the set of all subconcepts of $\mathcal{O}, C_1, C_2$. A *type* is defined as before as a subset of Γ that is realizable in a model of $\mathcal{O}$. To address that one can force elements to be bisimilar, we have to extend the notion of a mosaic m to be a pair (T_1, T_2) of *sets* of types. Intuitively, a mosaic (T_1, T_2) describes collections of elements in two interpretations $\mathcal{I}_1, \mathcal{I}_2$ which realize precisely the types in T_1, T_2 and are mutually $\mathcal{ALCO}(\Sigma)$-bisimilar. Naturally, not all such mosaics can be realized as described and the goal is to find the realizable ones. We write $(T_1, T_2) \rightsquigarrow_r (T_1', T_2')$ if for $i = 1, 2$ and every $t \in T_i$, there is $t' \in T_i'$ with $t \rightsquigarrow_r t'$, that is, every type in T_i has a viable r-successor in T_i'.

Let $\mathcal{M}$ be a set of mosaics. A mosaic $(T_1, T_2) \in \mathcal{M}$ is *bad* if it violates one of the conditions atomic consistency and existential saturation, suitably generalized to sets of types:

(Atomic Consistency) for every $A \in \Sigma$ and every $t, t' \in T_1 \cup T_2$, $A \in t$ iff $A \in t'$;

(Existential Saturation) for every $r \in \Sigma$, $i = 1, 2$, every $t \in T_i$, and every $\exists r.C \in t$, there is $(T_1', T_2') \in \mathcal{M}$ with $(T_1, T_2) \rightsquigarrow_r (T_1', T_2')$ and such that $C \in t'$ for some $t' \in T_i'$ with $t \rightsquigarrow_r t'$.

We need a further property of sets of mosaics to ensure that nominals are handled correctly: they are realized precisely once in every interpretation. A set $\mathcal{M}$ of mosaics is *good for nominals* if for every individual $a \in \Gamma$ and $i = 1, 2$, there is exactly one t_a^i with $\{a\} \in t_a^i \in \bigcup_{(T_1,T_2)\in\mathcal{M}} T_i$ and exactly one pair $(T_1, T_2) \in \mathcal{M}$ with $t_a^i \in T_i$. Moreover, if $a \in \Sigma$, then the pair takes one of the forms $(\{t_a^1\}, \{t_a^2\})$, $(\emptyset, \{t_a^2\})$, or $(\{t_a^1\}, \emptyset)$. One can then show:

▶ **Lemma 25** (Lemma 6.5 in [8]). *C_1 and C_2 are jointly $\sim_{\mathcal{ALCO},\Sigma}$-consistent iff there is a set $\mathcal{M}^*$ of mosaics that is good for nominals and contains no bad mosaic such that there is $(T_1, T_2) \in \mathcal{M}^*$ and $t_1 \in T_1, t_2 \in T_2$ with $C_1 \in t_1$ and $C_2 \in t_2$.*

The upper bound in Theorem 24 now follows from the fact that there are only double exponentially many mosaics and that an $\mathcal{M}^*$ as in Lemma 25 can be found (if it exists) by eliminating mosaics from all *maximal* sets of mosaics that are good for nominals. The upper bound for $\mathcal{ALCH}$ and the other logics in Theorem 24 is similar. In fact, mosaic elimination has been used to show decidability of interpolant existence also for other logics that related to DLs and relevant to KR [71, 85, 86].

$\mathcal{EL}$ and its Relatives Given the importance of the lightweight DL $\mathcal{EL}$ in applications, we report also on the (surprisingly different) situation in $\mathcal{EL}$ and its relatives. While $\mathcal{EL}$ itself and its extension with role hierarchies or transitive roles do enjoy the CIP [98, 36], most other extensions of $\mathcal{EL}$ such as with inverse roles, nominals, or the combination of transitive roles and role hierarchies do not [36]. Thus, the extension with $\mathcal{H}$ does not lead to the loss of the CIP in $\mathcal{EL}$, but it does in $\mathcal{ALC}$, and conversely, the extension with $\mathcal{I}$ does lead to the the loss of the CIP in $\mathcal{EL}$, but not in $\mathcal{ALC}$. As a rule of thumb, the complexity of the interpolant existence problem in an extension of $\mathcal{EL}$ without CIP coincides with the complexity of subsumption in that extension [36]. This is in contrast to $\mathcal{ALC}$ where interpolant existence is generally one exponent more difficult than subsumption for extensions without CIP.

Repairing the CIP We conclude this part with a discussion of *repairing* the lack of the CIP by allowing interpolants from a richer logic. A notable positive instance of this approach is to allow formulas in the guarded, two-variable fragment (GF^2) of FO as interpolants under $\mathcal{ALCH}$ ontologies: GF^2 extends $\mathcal{ALCH}$ and it is known that GF^2 does enjoy the CIP [63, 64]. Consequently, there is a GF^2-interpolant for every valid subsumption in $\mathcal{ALCH}$. The situation is more complicated in the case of $\mathcal{ALCO}$. While it would be interesting to investigate existence of interpolants in logics more expressive than $\mathcal{ALCO}$, say GF^2 with constants, it has been shown that (under mild conditions) there is no *decidable* extension of $\mathcal{ALCO}$ that enjoys CIP [26]. Hence, there is no decidable logic which guarantees the existence of interpolants for every valid subsumption in $\mathcal{ALCO}$.

4.2 Beth Definability

Since interpolation is intimately tied to definability, c.f. Chapter 2 and Chapter 14, and since definability is a central topic when working with DL ontologies, in this section we take a closer look at this connection in the context of DLs. We start with introducing the relevant notions. Recall that, for a signature Σ, the Σ-*reduct* $\mathcal{I}|_\Sigma$ of an interpretation $\mathcal{I}$ is the interpretation obtained from $\mathcal{I}$ by dropping the interpretation of all symbols not in Σ.

▶ **Definition 26** (Explicit/Implicit Definability). *Let $\mathcal{L}$ be any DL. Let $\mathcal{O}$ be an $\mathcal{L}$ ontology, C, C_0 be $\mathcal{L}$ concepts, and Σ be a signature. We say that C_0 is:*

- explicitly $\mathcal{L}(\Sigma)$-definable under $\mathcal{O}$ and C *if there is an explicit $\mathcal{L}(\Sigma)$-definition of C_0 under $\mathcal{O}$ and C, that is, an $\mathcal{L}(\Sigma)$ concept D satisfying $\mathcal{O} \models C \sqsubseteq (C_0 \leftrightarrow D)$, and*
- implicitly Σ-definable under $\mathcal{O}$ and C *if for all models $\mathcal{I}$ and $\mathcal{J}$ of $\mathcal{O}$ with $\mathcal{I}_{|\Sigma} = \mathcal{J}_{|\Sigma}$ and all $d \in C^{\mathcal{I}}$, we have $d \in C_0^{\mathcal{I}}$ iff $d \in C_0^{\mathcal{J}}$.*

Intuitively, C_0 is implicitly Σ-definable under $\mathcal{O}$ and C if for every model $\mathcal{I}$ of $\mathcal{O}$ and all $d \in C^{\mathcal{I}}$, the Σ-reduct $\mathcal{I}|_\Sigma$ "determines" whether $d \in C_0^{\mathcal{I}}$. Implicit definability can be equivalently defined in terms of subsumption: C_0 is implicitly Σ-definable under $\mathcal{O}$ and C iff

$$\mathcal{O} \cup \mathcal{O}_\Sigma \models C \sqcap C_0 \sqsubseteq C_\Sigma \to C_{0\Sigma} \tag{14}$$

where $\mathcal{O}_\Sigma$, C_Σ, and $C_{0\Sigma}$ are obtained from $\mathcal{O}$, C and C_0, respectively, by replacing every non-Σ symbol uniformly by a fresh symbol. It is worth noting that a common alternative definition of explicit/implicit definability does not use the context concept C; it is a special case of the above definition in which C is set to $\top$. We use the more general definition in order to establish a stronger relation to Craig interpolation later on. The following example illustrates implicit and explicit definability and makes do with the simpler definition.

▶ **Example 27.** Consider the $\mathcal{ALC}$ ontology consisting of the following CIs:

$$\mathsf{Parent} \equiv \exists \mathsf{child}.\top \qquad \mathsf{Parent} \equiv \mathsf{Father} \sqcup \mathsf{Mother}$$
$$\mathsf{Father} \sqsubseteq \mathsf{Man} \qquad \mathsf{Mother} \sqsubseteq \mathsf{Woman} \qquad \mathsf{Man} \sqsubseteq \neg \mathsf{Woman}$$

Then, Mother is implicitly Σ-definable under $\mathcal{O}$ and $C = \top$, for $\Sigma = \{\mathsf{hasChild}, \mathsf{Woman}\}$. Indeed, in any model $\mathcal{I}$ of $\mathcal{O}$, any element that satisfies Woman and has an hasChild-successor has to satisfy Mother, and other elements can not satisfy Mother. This is equivalent with saying that $\mathsf{Woman} \sqcap \exists \mathsf{hasChild}.\top$ is an explicit $\mathcal{ALC}(\Sigma)$-definition of Mother under $\mathcal{O}$. ⌟

It should be clear that, if a concept is explicitly $\mathcal{L}(\Sigma)$-definable under $\mathcal{O}$ and C, then it is implicitly Σ-definable under $\mathcal{O}$ and C, for any language $\mathcal{L}$. A logic enjoys the projective Beth definability property if the converse implication holds as well.

▶ **Definition 28.** *A DL $\mathcal{L}$* enjoys the projective Beth definability property *(*PBDP*) if for any $\mathcal{L}$ ontology $\mathcal{O}$, $\mathcal{L}$ concepts C and C_0, and signature $\Sigma \subseteq \mathsf{sig}(C, \mathcal{O})$ the following holds: if C_0 is implicitly Σ-definable under $\mathcal{O}$ and C, then C_0 is explicitly $\mathcal{L}(\Sigma)$-definable under $\mathcal{O}$ and C.*

Explicit definability is, similarly to interpolant existence, characterized via joint consistency.

▶ **Lemma 29.** *Let $\mathcal{L}$ be any DL. Let C_0, C be $\mathcal{L}$ concepts, $\mathcal{O}$ an $\mathcal{L}$ ontology, and Σ be a signature. Then the following conditions are equivalent:*

1. *there is no explicit $\mathcal{L}(\Sigma)$-definition for C_0 under $\mathcal{O}$ and C;*
2. *$C \sqcap C_0$ and $C \sqcap \neg C_0$ are jointly $\sim_{\mathcal{L},\Sigma}$-consistent under $\mathcal{O}$.*

Based on Lemma 29, one can prove the equivalence of PBDP and the CIP.

▶ **Lemma 30.** *For every extension $\mathcal{L}$ of $\mathcal{ALC}$ with any of $\mathcal{S}$,$\mathcal{I}$,$\mathcal{F}$,$\mathcal{O}$,$\mathcal{H}$, $\mathcal{L}$ enjoys the CIP iff $\mathcal{L}$ enjoys the PBDP. Moreover, there are polynomial time reductions between computing interpolants and computing explicit definitions.*

Proof. The proof of "⇒" is by a standard argument [40]: Assume that an $\mathcal{L}$ concept C_0 is implicitly Σ-definable under an $\mathcal{L}$ ontology $\mathcal{O}$ and $\mathcal{L}$ concept C, for some signature Σ. Then (14) holds. Take an $\mathcal{L}(\Sigma)$-interpolant I for $C \sqcap C_0$ and $C_\Sigma \to C_{0\Sigma}$ under $\mathcal{O} \cup \mathcal{O}_\Sigma$. Then I is an explicit $\mathcal{L}(\Sigma)$-definition of C_0 under $\mathcal{O}$ and C.

For "⇐", suppose $\mathcal{O}_1 \cup \mathcal{O}_2 \models C \sqsubseteq D$, for $\mathcal{L}$ ontologies $\mathcal{O}_1, \mathcal{O}_2$ and $\mathcal{L}$ concepts C, D, and let $\Sigma = \mathsf{sig}(\mathcal{O}_1, C_1) \cap \mathsf{sig}(\mathcal{O}_2, C_2)$. Based on Lemmas 19 and 29, one can show that there is a Craig interpolant for C and D under $\mathcal{O}_1 \cup \mathcal{O}_2$ iff there is an explicit $\mathcal{L}(\Sigma)$-definition of D under $D \to C$ and $\mathcal{O}_1 \cup \mathcal{O}_2$. Finally observe that D is implicitly Σ-definable under $\mathcal{O}_1 \cup \mathcal{O}_2$ and $D \to C$: The right-hand side of the concept inclusion in (14) is $(D_\Sigma \to C_\Sigma) \to D_\Sigma$, a tautology. ◀

Lemma 30 is remarkable since for many logics the CIP is strictly stronger than the PBDP, see Chapter 3 and for example [103]. On the one hand, the proof of "⇒" of Lemma 30 is rather robust and also applies to weaker logics, such as $\mathcal{EL}$, and weaker versions of explicit/implicit definability, such as the one with $C = \top$. On the other hand, the proof of "⇐" relies on both the presence of the context concept C and on the expressive power of $\mathcal{ALC}$. In fact, we conjecture that the "⇐"-direction does not hold for the weaker definition with $C = \top$.

Taking into account our knowledge about the CIP from Theorems 18 and 22, we obtain:

▶ **Corollary 31.** *$\mathcal{ALC}$ and its extensions by $\mathcal{S}$, $\mathcal{I}$, $\mathcal{F}$ enjoy the PBDP, but $\mathcal{ALCO}$ and $\mathcal{ALCH}$ and their extensions by $\mathcal{S}$, $\mathcal{I}$, $\mathcal{F}$ do not.*

We conclude the section with a brief discussion of two weaker forms of Beth definability that have also been considered in the DL literature.

- In the *non-projective Beth definability property (BDP)*, we are only interested in defining concept names $C_0 = A$ in terms of all other symbols in the signature, that is, $\Sigma = \mathsf{sig}(\mathcal{O}) \setminus \{A\}$. Clearly, the PBDP implies the BDP, but the other direction is not true in general. For example, $\mathcal{ALCH}$ enjoys the BDP, but it lacks the PBDP. $\mathcal{ALCO}$ does not enjoy already BDP. We refer the reader to [8] for a broader discussion.
- In the *concept-based Beth definability property (CBDP)* both implicit and explicit definability are defined in terms of signatures which always contain all relevant role names, that is, only concept names may be restricted. In the same way, one can define a corresponding *concept Craig interpolation property (CCIP)*. This notion has been studied extensively in [27]. Moreover, it has been shown recently using a sequent calculus that the highly expressive DL $\mathcal{RIQ}$ enjoys both CBDP and CCIP [102].

4.3 Applications

We discuss several applications of interpolants and explicit definitions, starting with a recently discovered relation of Craig interpolants to *description logic concept learning* in the presence of nominals. Informally, concept learning is the task of inducing a concept description from sets of positive and negative data examples, which has received a lot of interest in recent years, see e.g. [87, 70, 39]. The predominant application scenario is reverse-engineering of concepts to support ontology engineers in writing complex concepts. To make the connection to interpolation precise, we introduce some notation.

In our context, an *example* is a pair $(\mathcal{D}, a_0)$, where $\mathcal{D}$ is a database, that is, a finite set of facts of the form $A(a)$ and $r(a, b)$ with A a concept name, r a role name, and a, b individuals, and a_0 is an individual from database $\mathcal{D}$. A *labeled data set* is a pair (P, N) of sets of positive and negative examples. Let $\mathcal{L}$ be some DL, $\mathcal{O}$ be an $\mathcal{L}$ ontology, Σ a signature, and (P, N) a labeled dataset. An $\mathcal{L}(\Sigma)$-*separator* for $\mathcal{O}, P, N$ is an $\mathcal{L}(\Sigma)$ concept C such that

- $\mathcal{O} \cup \mathcal{D} \models C(a)$ for all $(\mathcal{D}, a) \in P$ and
- $\mathcal{O} \cup \mathcal{D} \models \neg C(a)$ for all $(\mathcal{D}, a) \in N$.

Intuitively, a separator is a potentially complex concept that distinguishes between the positive and negative examples in the sense of consistent learning. We refer to the induced problem of deciding the existence of an $\mathcal{L}(\Sigma)$-separator for given $\mathcal{O}, \Sigma, P, N$ with $\mathcal{L}$-*separator existence*.

It has been recently observed that there is a strong connection between interpolants and separators in many DLs extending $\mathcal{ALCO}$. More precisely, for such DLs $\mathcal{L}$ there are polynomial time reductions between $\mathcal{L}$-interpolant existence and $\mathcal{L}$-separator existence. Moreover, the reductions guarantee that interpolants can be used as separators and vice versa [8, Theorem 4.3]. The presence of nominals is crucial in these reductions as they are needed to encode the database. Theorem 24 implies that separator existence in $\mathcal{ALCO}$ and its extenions is a hard yet decidable problem.

A special case of separator existence is the existence of *referring expressions*. Referring expressions originate in linguistics to refer to a single object in the domain of discourse [24], but have recently been introduced in data management and KR [7, 84, 19, 9]. Existence of a referring expression for object a can be cast as a separator existence problem, in which the task is to find a separator between a and all other individuals in the domain of discourse. Indeed, in many cases, the concrete individual, say the internal id $p12345$, could be less meaningful to the user than a concise description in terms of a given signature, e.g., *head of finance*. Referring expressions are also a special case of explicit definitions.

Consider next the process of *ontology construction*. It offers essentially two ways to extend a present ontology by a new concept name, say A [27]. First, in the *explicit* manner, one adds a concept definition $A \equiv C$ for some concept description C over the signature of the present ontology. Second, in the *implicit* manner, one can add (several) concept inclusions with the property that in every model of the ontology, the interpretation of A is uniquely determined by the interpretation of the other symbols—in other words: A is implicitly defined. Of course, the explicit way is more transparent and thus to be preferred. However, if the

ontology language of choice enjoys the PBDP, there is actually an explicit definition for A, also in the second case. The notion of PBDP has been studied in this sense under the disguise of *definitorial completeness* in [12, 27].

Finally, we discuss ontology-mediated query answering (OMQ). Recall that in OMQ we access a knowledge base consisting of ontology $\mathcal{O}$ and database $\mathcal{D}$ using a query $\varphi(\vec{x})$ with the goal to determine the *certain answers*, that is, all tuples $\vec{a}$ such that $\mathcal{O} \cup \mathcal{D} \models \varphi(\vec{a})$. Depending on the ontology language, this is a potentially difficult entailment problem and the question is whether one can *rewrite* φ and $\mathcal{O}$ into $\varphi_{\mathcal{O}}$ such that $\mathcal{O} \cup \mathcal{D} \models \varphi(\vec{a})$ iff $\mathcal{D} \models \varphi_{\mathcal{O}}(\vec{a})$, for all $\vec{a}$. Note that in this case $\varphi_{\mathcal{O}}$ can be answered efficiently by a standard database system. The PBDP and the connection to interpolation from Lemma 30 have been used several times to study the existence and computation of such rewritings [123, 37, 38, 125]. Chapter 14 discusses further applications of the PBDP in the context of database query answering.

4.4 Computing Interpolants

In the previous sections, we have motivated the need for computing interpolants and explicit definitions in various applications. Unfortunately, in spite of that, we are not aware of any DL reasoner that supports the actual computation of interpolants. Hence, this section is restricted to the known theoretical results. By Lemma 30, we can focus on computing interpolants. We concentrate on extensions of $\mathcal{ALC}$ that enjoy the CIP and refer the reader to [36] for $\mathcal{EL}$ and its relatives. In the following theorem from [28], we refer to a DAG-representation in which each subconcept is represented only once as *succinct representation* of concepts.

▶ **Theorem 32** (Theorems 3.10, 3.26 in [28]). *Let $\mathcal{L}$ be $\mathcal{ALC}$ or any extension with $\mathcal{S}$, $\mathcal{I}$, $\mathcal{F}$. Then, one can compute, given $\mathcal{L}$ ontologies $\mathcal{O}_1, \mathcal{O}_2$ and $\mathcal{L}$ concepts C_1, C_2 with $\mathcal{O}_1 \cup \mathcal{O}_2 \models C_1 \sqsubseteq C_2$, in double exponential time a Craig interpolant of C_1 and C_2 under $\mathcal{O}_1 \cup \mathcal{O}_2$. If concepts are represented succinctly, then the interpolant can be computed in exponential time.*

In the same paper, it is shown that the statement in Theorem 32 is essentially optimal. We give the lower bound in terms of explicit definitions; it transfers to Craig interpolants via (the proof of) Lemma 30. It is also not difficult to see that it remains valid when we consider extensions of $\mathcal{ALC}$ with $\mathcal{S}$,$\mathcal{I}$,$\mathcal{F}$.

▶ **Theorem 33** (Theorem 4.11 in [28]). *Let $\Sigma = \{r, s\}$ consist of two role names. For every $n \geq 0$, there are an $\mathcal{ALC}$ concept C_n and an $\mathcal{ALC}$ ontology $\mathcal{O}_n$ of size polynomial in n such that $\Sigma \subseteq \mathsf{sig}(\mathcal{O}_n, C_n)$ and C_n is implicitly Σ-definable under $\mathcal{O}_n$, but the smallest explicit $\mathcal{ALC}(\Sigma)$-definition of C_n under $\mathcal{O}_n$ is double exponentially long in n.*

Proof sketch. The proof is via a well-known *path-set construction* [96]. The idea is that C_n together with $\mathcal{O}_n$ enforces a full binary r/s-tree of depth 2^n. One then shows that any $\mathcal{ALC}(\Sigma)$ concept D that explicitly defines such a tree is double exponentially long in n, since each path in the tree has to be reflected in a (different) path in the syntax tree of D. ◀

The original proof of Theorem 32 is by an extension of the tableau-based proof of the CIP for the respective DLs in [28]. Instead of giving it here, we provide two different perspectives on the computation problem, focusing on $\mathcal{ALC}$ for the sake of simplicity. The first one is a reduction of computing interpolants under ontologies to the case without ontologies. The second one is a refined analyis of the mosaic elimination algorithm that decides joint $\sim_{\mathcal{ALC},\Sigma}$-consistency given in Section 4.1. The benefit of the former is that this enables us to use all methods for computing interpolants in modal logic presented in Chapter 4. The benefit of the latter is that it provides the same upper bounds as claimed in Theorem 32.

Perspective 1: The Reduction

The main idea is to *materialize* the ontology to sufficient depth. To formalize this, let $\mathcal{ALC}_n(\Sigma)$ denote the set of all $\mathcal{ALC}(\Sigma)$ concepts of role depth at most n, and define for each $\mathcal{ALC}$ ontology $\mathcal{O}$ using role names Σ_R and each $n \geq 0$, a concept $\Phi^n_{\mathcal{O}} \in \mathcal{ALC}_n(\Sigma)$ by taking

$$\Phi^n_{\mathcal{O}} = \bigsqcap_{m \leq n} \bigsqcap_{r_1,\dots,r_m \in \Sigma_R} \forall r_1 \dots \forall r_m . \bigsqcap_{C \sqsubseteq D \in \mathcal{O}} (\neg C \sqcup D).$$

Intuitively, the concept $\Phi^n_{\mathcal{O}}$ expresses that the concept inclusions in $\mathcal{O}$ are satisfied along any Σ_R-path of length at most n. The following lemma, whose proof is similar to the proof of Lemma 9 in [27], provides the promised reduction.

▶ **Lemma 34.** *Let $\mathcal{O}$ be an $\mathcal{ALC}$ ontology, C_1, C_2 be $\mathcal{ALC}$ concepts, and $N > 2^{||\mathcal{O}||+||C_1||+||C_2||}$. Then, for all signatures Σ, all $n \geq 0$, and all $I \in \mathcal{ALC}_n(\Sigma)$ the following are equivalent:*

1. *I is an $\mathcal{ALC}(\Sigma)$-interpolant of C_1 and C_2 under $\mathcal{O}$;*
2. *I is an $\mathcal{ALC}(\Sigma)$-interpolant of $\Phi^{N+n}_{\mathcal{O}} \sqcap C_1$ and $\Phi^{N+n}_{\mathcal{O}} \to C_2$.*

Thus, if we were interested in (Craig) interpolants of a certain role depth n, we could use Lemma 34 to compute them using methods for modal logic from Chapter 4. Since Theorem 32 entails an exponential upper bound on the role depth of interpolants (an interpolant that is of exponential size in succinct representation has at most exponential role depth), this is also a complete approach for computing Craig interpolants. We refrain from analyzing the size of interpolants constructed via this method.

Perspective 2: Construction from Mosaic Elimination

We conduct a refined analysis of the mosaic elimination algorithm that decides joint consistency in $\mathcal{ALC}$ in Section 4.1. Let $\mathcal{O}, C_1, C_2, \Sigma$ be an input to mosaic elimination. Recall that the elimination algorithm computes a sequence $\mathcal{M}_0, \mathcal{M}_1, \dots$ of sets of mosaics, where $\mathcal{M}_0$ is the set of all mosaics, and $\mathcal{M}_{i+1}$ is obtained from $\mathcal{M}_i$ by eliminating bad mosaics. We say that a mosaic (t_1, t_2) is *eliminated in round ℓ* if $(t_1, t_2) \in \mathcal{M}_\ell$ but $(t_1, t_2) \notin \mathcal{M}_{\ell+1}$. In a slight abuse of notation, we treat a type t as the concept $\bigsqcap_{D \in t} D$.

▶ **Lemma 35.** *If a mosaic (t_1, t_2) is eliminated in round ℓ, then there is an $\mathcal{ALC}_\ell(\Sigma)$ concept I such that $\mathcal{O} \models t_1 \sqsubseteq I$ and $\mathcal{O} \models t_2 \sqsubseteq \neg I$.*

Proof. The proof is by induction on ℓ. For the base case, let $\ell = 0$. If (t_1, t_2) is eliminated in round 0, then this is due to the violation of atomic consistency. Then we can choose $I = A$ or $I = \neg A$ depending on whether the problematic concept name A is contained in t_1 or in t_2.

For the inductive case, let $\ell > 0$. Then (t_1, t_2) has been eliminated due to violation of existential saturation, that is, there is $r \in \Sigma$, $i \in \{1, 2\}$, and $\exists r.C \in t_i$ such that each $(t_1', t_2') \in \mathcal{M}_i$ with $C \in t_i$ satsfies $(t_1, t_2) \not\rightsquigarrow_r (t_1', t_2')$. Assume first that $i = 1$, and let T_1, T_2 be the following sets of types:

$$T_1 = \{t \mid C \in t \text{ and } t_1 \rightsquigarrow_r t\} \quad \text{and} \quad T_2 = \{t \mid t_2 \rightsquigarrow_r t\}.$$

By induction hypothesis, for each $(t_1', t_2') \in T_1 \times T_2$ there is an $\mathcal{ALC}_{\ell-1}(\Sigma)$ concept $I_{t_1', t_2'}$ such that $\mathcal{O} \models t_1' \sqsubseteq I_{t_1', t_2'}$ and $\mathcal{O} \models t_2' \sqsubseteq \neg I_{t_1', t_2'}$. We choose

$$I = \exists r. \bigsqcup_{t_1' \in T_1} \bigsqcap_{t_2' \in T_2} I_{t_1', t_2'}.$$

Clearly, $I \in \mathcal{ALC}_\ell(\Sigma)$. We verify $\mathcal{O} \models t_1 \sqsubseteq I$ and $\mathcal{O} \models t_2 \sqsubseteq \neg I$.

- For the former, let $\mathcal{I}$ be a model of $\mathcal{O}$ and $d \in t_1^{\mathcal{I}}$. Since $\exists r.C \in t_1$, there is some $e \in C^{\mathcal{I}}$ with $(d, e) \in r^{\mathcal{I}}$. Let $t_1' = \mathsf{tp}_{\mathcal{I}}(e)$ and note that $t_1' \in T_1$. By what was said above, we have $\mathcal{O} \models t_1' \sqsubseteq I_{t_1', t_2'}$ for all $t_2' \in T_2$. Thus, $d \in (\exists r. I_{t_1', t_2'})^{\mathcal{I}}$ for all $t_2' \in T_2$, and hence $d \in I^{\mathcal{I}}$.
- For the latter, let $\mathcal{I}$ be a model of $\mathcal{O}$ and $d \in t_2^{\mathcal{I}}$. We have to show that $d \notin I^{\mathcal{I}}$, that is, $d \in (\forall r. \bigsqcap_{t_1' \in T_1} \bigsqcup_{t_2' \in T_2} \neg I_{t_1', t_2'})^{\mathcal{I}}$. Suppose e is an r-successor of d and let $t_2' = \mathsf{tp}_{\mathcal{I}}(e)$. Note that $t_2' \in T_2$. By what was said above, we have $e \in (\neg I_{t_1', t_2'})^{\mathcal{I}}$ for all $t_1' \in T_1$. Thus $d \notin I^{\mathcal{I}}$.

The case $i = 2$ is dual; the interpolant has a leading universal quantifier then. ◀

Using the same arguments, one can finally show the following:

▶ **Lemma 36.** *For $i = 1, 2$, let T_i be the set of all types that contain C_i. If the result $\mathcal{M}^*$ of the mosaic elimination does not contain any mosaic from $T_1 \times T_2$, then*

$$I = \bigsqcup_{t_1 \in T_1} \bigsqcap_{t_2 \in T_2} I_{t_1, t_2}$$

is an $\mathcal{ALC}_N(\Sigma)$-interpolant of C_1 and $\neg C_2$ under $\mathcal{O}$, for $N \leq 2^{||\mathcal{O}||+||C_1||+||C_2||}$.

A straightforward analysis shows that, moreover, the size of interpolants I constructed in this way is at most double exponential in the size of the input $\mathcal{O}, C_1, C_2, \Sigma$, and at most exponential if we allow for succinct representation. Thus, this approach achieves the upper bounds claimed in Theorem 32.

We conjecture that it is rather straightforward to extend this way of constructing interpolants to extensions of $\mathcal{ALC}$ enjoying the CIP. It appears to be much more challenging for

the extension with $\mathcal{H}$ and/or $\mathcal{O}$, which lead to the loss of the CIP. Some progress has recently been made for the extension with $\mathcal{H}$ in [67], which also reports that an earlier algorithm for computing interpolants under $\mathcal{ALCH}$ ontologies from [8] is not correct (and not easy to fix).

5 Interpolation in Logic Programming

In this section, we briefly discuss interpolation in the context of logic programming (LP) [13], where applications of interpolation align with the ones discussed in Section 1. Here, we focus on logic programs under the stable model semantics [48], as this arguably is the most widely used semantics in LP. In particular, it is the semantics employed in answer set programming (ASP) [46, 90], a form of declarative programming aimed at solving large combinatorial (commonly NP-hard) search problems. In addition, a large part of the research in LP on interpolation and, in particular, on the problem of forgetting focusses on ASP, motivated by its applications.[5]

5.1 Logic Programs and Answer Sets

We start by recalling necessary notions and notation for logic programs [46, 90].

We consider first-order signatures Σ consisting of mutually disjoint and countably infinite sets of constants, functions, predicates, and variables. This allows us to define *terms* as usual as constants, variables, and complex terms $f(t_1, \ldots, t_n)$, where f is a function and $t_1, \ldots t_n$ are terms, as well as *atoms* $p(t_1, \ldots, t_n)$ where p is a predicate and $t_1, \ldots t_n$ are terms. We denote the set of all atoms over Σ with $A(\Sigma)$.

Given Σ, a *(logic) program* P is a finite set of *rules* r of the form[6]

$$a_1 \vee \ldots \vee a_k \leftarrow b_1, ..., b_l, not\, c_1, ..., not\, c_m, not\, not\, d_1, ..., not\, not\, d_n \tag{15}$$

where $\{a_1, \ldots, a_k, b_1, \ldots, b_l, c_1, \ldots, c_m, d_1, \ldots, d_n\} \subseteq A(\Sigma)$. We also represent such rules with

$$H \leftarrow B^+, not\, B^-, not\, not\, B^{--}\ , \tag{16}$$

where $H = \{a_1, \ldots, a_k\}$, $B^+ = \{b_1, \ldots, b_l\}$, $B^- = \{c_1, \ldots, c_m\}$, and $B^{--} = \{d_1, \ldots, d_n\}$. We distinguish the *head* of r, $head(r) = H$, and its *body*, $body(r) = B^+ \cup not\, B^- \cup not\, not\, B^{--}$, where $not\, B^-$ and $not\, not\, B^{--}$ represent $\{not\, p \mid p \in B^-\}$ and $\{not\, not\, p \mid p \in B^{--}\}$, respectively. We refer to the set of all (logic) programs as the *class of (logic) programs*. Different more specific kinds of rules exist: if $n = 0$, then r is *disjunctive*; if also $k \leq 1$, then r is *normal*; if in addition $m = 0$, then r is *Horn*, if, moreover, $l = 0$, then r is a *fact*. The

[5] Stable models and answer sets are commonly used synonymously. Though the former is more associated with semantics and the latter with problem solving, the latter has gained wider adoption [89, 22].

[6] We admit double negation in rules which is historically less common, but needed in our context.

class of *disjunctive programs* is defined as a set of finite sets of disjunctive rules, and other classes can be defined accordingly.

For the semantics, we focus on programs in which variables have been instantiated. Given a program P, this is captured by a ground program, denoted $ground(P)$, which is obtained from P by replacing the variables in P with ground terms over Σ in all possible ways. We also call variable-free terms, atoms, rules, and programs *ground*. Accordingly, we consider the set of all ground atoms over Σ, denoted $A_{gr}(\Sigma)$. Note that ground programs correspond to propositional programs, where the signature is restricted to a countably infinite set of propositional variables, which can be viewed as atoms built over predicates with arity 0, i.e., no terms, in which case Σ would simplify to such a set of nullary predicates.

We recall the definition of answer sets of logic programs [49], utilizing here a variant building on a characteriztion of answer sets [114] as a minimal model notion grounded in the monotonic intuitionistic logic of here-and-there (HT) going back to Heyting [62]. Note that HT logic is actually defined for a general propositional language, which covers propositional logic programs, and which has interpolation [104]. Here, we restrict our focus to logic programs.

First, we recall the *reduct* P^I of a ground program P with respect to a set I of ground atoms, adapted here to extended rules:

$$P^I = \{H \leftarrow B^+ \mid r \text{ of the form (16) in } P \text{ such that } B^- \cap I = \emptyset \text{ and } B^{--} \subseteq I\}.$$

The idea is, assuming the atoms in I as true, to transform (ground) P into a corresponding program P^I that does not contain negation. Then, HT-models are defined as a pair of sets of (ground) atoms, making use of a standard entailment relation $\models$ for programs, where (ground) programs are viewed as a conjunction of implications corresponding to the rules of the form (15) with $\neg$, $\wedge$, and $\vee$ denoting classical negation, conjunction, and disjunction, respectively. Intuitively, in an HT-interpretation $\langle X, Y\rangle$, Y is used to assess satisfaction of P as such, and X is employed to verify satisfaction of the reduct P^Y. An answer set is then a minimal model that satisfies the reduct with respect to itself.

Formally, let P be a ground program. An *HT-interpretation* is a pair $\langle X, Y\rangle$ such that $X \subseteq Y \subseteq A_{gr}(\Sigma)$, and $\langle X, Y\rangle$ is an *HT-model of* P if $Y \models P$ and $X \models P^Y$. The set of *all HT-models of* P is written $\mathcal{HT}(P)$, and P is *satisfiable* if $\mathcal{HT}(P) \neq \emptyset$. A set of (ground) atoms Y is an *answer set* of P if $\langle Y, Y\rangle \in \mathcal{HT}(P)$, and there is no $X \subsetneq Y$ such that $\langle X, Y\rangle \in \mathcal{HT}(P)$. We denote by $\mathcal{AS}(P)$ the set of all answer sets of P, and call P *coherent* if $\mathcal{AS}(P) \neq \emptyset$.

We use two entailment relations between logic programs P_1, P_2 for the two introduced model notions, allowing to specify that P_1 entails P_2: namely, $\models_{\mathsf{HT}}$ represents entailment for HT logic, and $\mid\!\sim_c$ represents cautious entailment for answer sets. Formally, given programs P_1 and P_2, $P_1 \models_{\mathsf{HT}} P_2$ holds if $\mathcal{HT}(P_1) \subseteq \mathcal{HT}(P_2)$, and $P_1 \mid\!\sim_c P_2$ holds if $\mathcal{AS}(P_1) \subseteq \mathcal{AS}(P_2)$. Note that this also allows us to express that program P_1 entails a fact, a set of facts, or a disjunction, etc, as P_2 allows to represent these.

▶ **Example 37.** Consider for simplicity the following (propositional) program P:

$$a \leftarrow not\, b \qquad b \leftarrow not\, c \qquad e \leftarrow d \qquad d \leftarrow a$$

Then $\langle b, bde\rangle$ is an HT-model of P because $\{b,d,e\} \models P$ and $\{b\} \models P^{\{b,d,e\}}$ where $P^{\{b,d,e\}} = \{b \leftarrow;\ e \leftarrow d;\ d \leftarrow a\}$. Hence, $\{b,d,e\}$ is not an answer set of P as it violates the minimality condition. Moreover, $\{b,d\}$ is not an answer set of P because $\{b,d\} \not\models P$. In fact, we can verify that $\{b\}$ is its only answer set. Hence, $b \leftarrow$ is entailed by P with respect to $\mid\!\sim_c$. ⌟

5.2 Craig Interpolants

In nonmonotonic formalisms, including logic programming, previously drawn conclusions may become invalid in the presence of new information. For example, if we were to add a rule $c \leftarrow$ to the program in Example 37, then $\{b\}$ would no longer be an answer set. Instead, the unique answer set would be $\{a,d,e\}$. This raises problems for (Craig) interpolation for logic programs, as simply replacing the entailment relation $\models$ in Definition 1 with a nonmonotonic entailment relation $\mid\!\sim$, which is not transitive in general, may not work.

Following ideas in the literature [40], Gabbay et al. [41] proposed to adjust the central condition of Craig interpolants to logic programs using two entailment relations. We recall this generalized definition of interpolants using two abstract entailment relations.

▶ **Definition 38.** *Let $\vdash_1$ and $\vdash_2$ be entailment relations and ϕ, ψ programs such that $\phi \mid\!\sim \psi$. A $(\vdash_1, \vdash_2)$* interpolant for (ϕ, ψ) *is a program χ such that $\phi \vdash_1 \chi$ and $\chi \vdash_2 \psi$, and χ uses only non-logical symbols occurring in both ϕ and ψ.*

This is motivated by the aim to combine a nonmonotonic entailment relation $\mid\!\sim$ with its *deductive base*, i.e., a logic L with monotonic entailment relation $\vdash_L$ such that (i) $\vdash_L \subseteq \mid\!\sim$; (ii) for programs ϕ_1, ϕ_2, if $\phi_1 \equiv_L \phi_2$, then $\phi_1 \approx \phi_2$ (i.e., ϕ_1 and ϕ_2 have the same nonmonotonic entailments); and (iii) if $\phi \mid\!\sim \chi$ and $\chi \vdash_L \psi$, then $\phi \mid\!\sim \psi$.

Two kinds of Craig interpolants are defined. The stronger one $(\mid\!\sim, \vdash_L)$ builds on this notion of deductive base, which guarantees among other things that the relation $(\mid\!\sim, \vdash_L)$ is transitive in the sense of (iii). The weaker one, $(\mid\!\sim, \mid\!\sim)$, is implied by the former, but does not guarantee transitivity per se.

Also, two variants of the entailment relation $\mid\!\sim_c$ are considered, one which is aligned with closed world assumption (CWA), i.e., atoms that cannot be proven true are considered false, and one which is aligned with open world assumption (OWA), where no such conclusions on false atoms can be drawn.

In the latter case, it is possible to show for propositional programs that $(\mid\!\sim_c, \models_{\mathsf{HT}})$ interpolants exist by constructing a representation of the minimal models of ϕ, taking advantage of the fact that HT logic serves as deductive base for answer set semantics and that Craig interpolants as per Definition 1 are guaranteed to exist for HT logic [104].

However, ASP employs CWA, and in this case, in general, only $(\mid\!\sim_c, \mid\!\sim_c)$ interpolants do exist. Here, the corresponding model representation can be obtained because, in addition to

the previously mentioned points, under CWA, the non-logical symbols occurring in ψ, but not in ϕ, can be considered false. This also ensures, among others, transitivity of ($\mid\sim_c$, $\mid\sim_c$) [41].

▶ **Theorem 39.** *([41]) For coherent propositional programs, ($\mid\sim_c$, $\mid\sim_c$) interpolants are guaranteed to exist.*

The stronger notion of ($\mid\sim_c, \models_{\mathsf{HT}}$) interpolants only holds for coherent propositional programs under CWA when the non-logical symbols of ψ are a subset of those of ϕ [41].

However, in the first-order case, such interpolants may not exist in general, as the formula representing the answer sets may not be first-order definable. Yet, first-order definability can be ensured for programs where function symbols are not allowed and where variables are safe. Here, *safe* refers to all variables in a rule having to appear in atoms in B^+ in (16), which ensures that any grounding is finite as well.[7] This restriction is not problematic in practice in the context of ASP with variables, as programs are required to be safe anyway for the grounding, employed in state-of-the-art ASP solvers, such as clingo [47] or DLV2 [4], where the usage of function symbols is also severely restricted to avoid grounding problems.

▶ **Theorem 40.** *([41]) For coherent, function-free, and safe first-order programs, ($\mid\sim_c$, $\mid\sim_c$) interpolants are guaranteed to exist.*

This result takes advantage of the fact that QHT [91], the first-order extension of HT logic, also has interpolation [112], and thus can serve as the monotonic base logic in this case.

5.3 Uniform Interpolation and Forgetting

As mentioned in Section 1, uniform interpolation and forgetting are strongly connected in their aim to reduce the language while preserving information over the remaining language. However, in the context of ASP, mainly forgetting has drawn considerable attention, varying from early purely syntactic approaches to a number of approaches using different means to capture the idea of preserving information over the remaining language [54]. In the following, we briefly discuss the main results and their implications on uniform interpolants.

In accordance with the literature, and to ease presentation, we focus on the propositional case. Hence, the signature Σ simplifies to propositional variables, i.e., only predicates of arity 0. We start by refining the definition of uniform interpolants in this context, as before admitting a generic entailment relation $\vdash$.

▶ **Definition 41.** *Let $V \subseteq \Sigma$ be a set of atoms, P a propositional program, and $\vdash$ an entailment relation. A* uniform $(V,\vdash)$-interpolant for P *is a program P_V such that*

- $P \vdash P_V$;
- P_V *uses only atoms from* V; *and*
- *for any program P' using only atoms from V, if $P \vdash P'$, then $P_V \vdash P'$.*

[7] In fact, a slightly more general notion of safe is used to capture a wider range of ASP constructs [41].

Note that for arbitrary nonmonotonic entailment relations, the first condition may in general not hold. But, building on results from forgetting, such interpolants can indeed be found.

For forgetting, the focus is on operators that return a program over the remaining language. Here, unlike uniform interpolation, the atoms to be forgotten are explicitly mentioned.

▶ **Definition 42.** *Given a class of logic programs $\mathcal{C}$ over Σ, a* forgetting operator (over $\mathcal{C}$) *is defined as a function $\mathsf{f} : \mathcal{C} \times 2^{\Sigma} \to \mathcal{C}$ where, for each $P \in \mathcal{C}$ and $V \subseteq \Sigma$, $\mathsf{f}(P, V)$, the* result of forgetting about V from P*, is a program over $\Sigma(P) \setminus V$.*

In the literature (see, e.g., [54]), operators then aim to preserve the semantic relations between atoms in $\Sigma(P)\setminus V$ from P, such as the dependencies discussed in the following example.

▶ **Example 43.** Consider again program P from Example 37. If we want to forget about d from P, then the first two rules should remain unchanged, while the latter two should not occur in the result. At the same time, implicit relations, such as e depending on a via d, should be preserved. Hence, we would expect the result $\{a \leftarrow \mathit{not}\, b;\ b \leftarrow \mathit{not}\, c;\ e \leftarrow a\}$. Alternatively, consider forgetting about b from P. Now, the final two rules remain, whereas the first two would be replaced by one linking a and c, i.e., $\{a \leftarrow \mathit{not}\, \mathit{not}\, c;\ e \leftarrow d;\ d \leftarrow a\}$. ⌟

Note that, in the second part, we cannot simply use the rule $a \leftarrow c$ instead, absorbing the double negation, as this is not equivalent in general (in terms of HT-models). Thus, results of forgetting may be required to be in a class more general than the one of the given program.

Example 43 illustrates the main idea of such semantic relations to be preserved, but leaves a formal specification open. Such characterization is usually semantic, i.e., a characterization of the desired models, which does not fix a specific program, and hence cannot align with a single function. This is captured by classes of forgetting operators that conjoin several operators based on a common characterization. Here, since we are interested in the existence of interpolants, we focus on individual operators instead (simplifying some technical material).

A number of different operators and classes of these have been defined in the literature, as well as certain properties with the aim of aiding the semantic characterization. We recall relevant properties in the context of interpolation and refer for a full account to [54].

We start with a property (for forgetting operators f) comparing the answer sets of a program and its result of forgetting. For that, given a set of atoms V, the *V-exclusion* of a set of answer sets $\mathcal{M}$, written $\mathcal{M}_{\|V}$, is defined as $\{X\setminus V \mid X \in \mathcal{M}\}$.

(CP) f satisfies *Consequence Persistence* if, for each program P and $V \subseteq \Sigma$, we have $\mathcal{AS}(\mathsf{f}(P, V)) = \mathcal{AS}(P)_{\|V}$.

Essentially, **(CP)** requires that answer sets of the forgetting result be answer sets of the original program projected to the remaining language and vice-versa [127]. There are operators that satisfy **(CP)**, e.g., those in class F_{SM} [127], whose forgetting results are characterized as the subset of HT-models (projected to the remaining language) such that the condition for **(CP)** holds. We can show that, if f satisfies (**CP**), then its forgetting result $\mathsf{f}(P, V)$ provides a uniform $(V, \mid\!\sim_c)$-interpolant.

▶ **Theorem 44.** *([127]) For propositional programs P and $V \subseteq \Sigma$, uniform $(V, \sim_c)$-interpolants are guaranteed to exist.*

The intuition here is that, since **(CP)** holds, we have $P \sim_c \mathsf{f}(P,V)$ and, likewise, the third condition of Def. 41 holds. Note that, for the two programs in Example 43, both would constitute uniform $(V, \sim_c)$-interpolants with $V = \{a, b, c, e\}$ in the first case and $V = \{a, c, d, e\}$ in the second.

Other properties rather focus on HT-models and entailments based on this notion [128].

(W) f satisfies *Weakening* if, for each program P and $V \subseteq \Sigma$, we have $P \models_{\mathsf{HT}} \mathsf{f}(P,V)$.

(PP) f satisfies *Positive Persistence* if, for each program P and $V \subseteq \Sigma$: if $P \models_{\mathsf{HT}} P'$, with $\Sigma(P') \subseteq \Sigma \backslash V$, then $\mathsf{f}(P,V) \models_{\mathsf{HT}} P'$.

These properties are aligned with Definition 41, and based on them and classes that satisfy these, e.g., F_{HT}, we can also capture interpolants with respect to $\models_{\mathsf{HT}}$.

▶ **Theorem 45.** *([129]) For propositional programs P and $V \subseteq \Sigma$, uniform $(V, \models_{\mathsf{HT}})$-interpolants are guaranteed to exist.*

Here, again both programs in Example 43 are uniform $(V, \models_{\mathsf{HT}})$-interpolants. For a simple example where these two kinds of interpolants differ, consider $P = \{a \leftarrow \mathit{not}\, b;\ b \leftarrow \mathit{not}\, a\}$. Then, $\{a \leftarrow \mathit{not}\, \mathit{not}\, a\}$ is uniform $(a, \sim_c)$-interpolant, but not a uniform $(a, \models_{\mathsf{HT}})$-interpolant. On the other hand $\{\}$ is a $(a, \models_{\mathsf{HT}})$-interpolant, but not a uniform $(a, \sim_c)$-interpolant.

Determining whether a given program is a result of forgetting, and hence whether it is a uniform interpolant, is computationally expensive. It has been shown that, for F_{HT} and F_{SM}, this problem is Π_2^P-complete with respect to data complexity (where the size of the input is measured in the number of facts) [129, 127]. It has also been argued that, e.g., for F_{SM} the algorithm to compute a result may result in an exponential blow-up in the size of the overall program [127].

Stronger notions of preserving semantic relations have been considered in Forgetting [72].

(SP) f satisfies *Strong Persistence* if, for each program P, $V \subseteq \Sigma$, and program R with $\Sigma(R) \subseteq \Sigma \backslash V$, we have $\mathcal{AS}(\mathsf{f}(P,V) \cup R) = \mathcal{AS}(P \cup R)_{\|V}$.

The idea is to preserve the answer sets no matter what other program is added to the given program and its result of forgetting. This turns out to be not possible in general [53], and concise conditions when such forgetting is possible and classes of operators approximating this have been investigated [55]. Complementarily, when the added programs R are restricted to sets of facts, forgetting is always possible with applications in modular ASP [52]. In both cases, it has been shown that determining whether a given program is a result of forgetting for these classes is more demanding, i.e., Π_3^P-complete (data complexity). This confirms that such notions of forgetting indeed go beyond uniform interpolation. Also, syntactic operators have been considered for these classes, but their construction may again, in the worst case, result in an exponential blowup in the size of the program [16, 15, 51]. Corresponding

algorithms of these, as well as those building on constructing a representative of the semantic characterization have been implemented in the Web tool ForgettingWeb [17].

One open problem for future research is how to tackle in general finding uniform interpolants/forgetting for first-order programs with variables in a wider sense (without requiring the grounding up-front). Possible avenues include extensions of the language of programs, in the line of the work by Aguado et al., which allows Strong Persistence to always hold [2]. Alternatively, connections to related notions on abstraction in ASP might be pursued, that rather aim at abstracting different elements of the language into a conjoined abstract one, either on the object level or in between different concepts [117, 118, 119, 120]. A further option arises from recent work where first-order Craig-interpolation is used to synthesize strongly equivalent programs given some background knowledge (in the form of a program) with an implementation using first-order reasoners [61].

6 Conclusion

We discussed interpolation in two relevant KR formalisms, DLs and logic programming. In the context of DLs, we focussed on uniform interpolation at the level of ontologies, and on Craig interpolation at the level of concepts. Uniform interpolation is strongly related to the topic of *forgetting*, which has been extensively studied in many settings. Interpolation has also been extensively studied for modal logics and classical logics, which are formalisms relevant for KR that are discussed in other chapters of this volume. Interestingly, there is comparably little research on interpolation in other formalisms relevant for KR.

With logic programs, we discussed one formalism with a nonmonotonic entailment relation. Nonmonotonic reasoning plays indeed a central role in many KR applications, and many more nonmonotonic KR formalisms have been proposed. Examples include *default logic* [18], *belief revision* [45], *argumentation frameworks* [14], as well as semantics based on *circumscription* [105] and on *repairs* [88]. Interpolation and the related notion of forgetting have been researched in some of these formalisms as well, but many problems remain open. The investigations in [42, Chapter 9], [121, Chapter 6] and [5] discuss interpolation in nonmonotonic logics on a more abstract level. Interpolation and Beth definability for default logics is discussed in [25], and forgetting for defeasible logic has been investigated in [6].

While there are implemented systems for computing uniform interpolants of DL ontologies, Craig interpolation and Beth definability have so far been investigated mostly on a theoretical level. Given the increased interest in explainability [107, 3] and supervised learning [113] for DLs in the recent years, which are relevant use cases of Craig interpolation, this situation might change in the future. Indeed, Craig interpolation at the level of ontologies, implemented via a reduction to uniform interpolation, is used in the ontology explanation tool Evee to create so-called *elimination proofs* [3].

Acknowledgments

The authors would like to thank Michael Benedikt and Balder ten Cate for valuable comments on earlier versions of this chapter. M. Knorr acknowledges partial support by project BIO-REVISE (2023.13327.PEX) and NOVA LINCS (UIDB/04516/2025) with the financial support of FCT.IP.

References

1 Wilhelm Ackermann. Zum Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 111(1):61–63, 1935. doi:10.1007/BF01448035.

2 Felicidad Aguado, Pedro Cabalar, Jorge Fandinno, David Pearce, Gilberto Pérez, and Concepción Vidal. Forgetting auxiliary atoms in forks. *Artif. Intell.*, 275:575–601, 2019. doi:10.1016/J.ARTINT.2019.07.005.

3 Christian Alrabbaa, Stefan Borgwardt, Tom Friese, Anke Hirsch, Nina Knieriemen, Patrick Koopmann, Alisa Kovtunova, Antonio Krüger, Alexej Popovic, and Ida S. R. Siahaan. Explaining reasoning results for OWL ontologies with Evee. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024*, 2024. doi:10.24963/KR.2024/67.

4 Mario Alviano, Francesco Calimeri, Carmine Dodaro, Davide Fuscà, Nicola Leone, Simona Perri, Francesco Ricca, Pierfrancesco Veltri, and Jessica Zangari. The ASP system DLV2. In *Logic Programming and Nonmonotonic Reasoning - 14th International Conference, LPNMR 2017*, volume 10377 of *Lecture Notes in Computer Science*, pages 215–221. Springer, 2017. doi:10.1007/978-3-319-61660-5_19.

5 Eyal Amir. Interpolation theorems for nonmonotonic reasoning systems. In *Logics in Artificial Intelligence, European Conference, JELIA 2002*, volume 2424 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 2002. doi:10.1007/3-540-45757-7_20.

6 Grigoris Antoniou, Thomas Eiter, and Kewen Wang. Forgetting for defeasible logic. In *Logic for Programming, Artificial Intelligence, and Reasoning - 18th International Conference, LPAR-18*, volume 7180 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2012. doi:10.1007/978-3-642-28717-6_9.

7 Carlos Areces, Alexander Koller, and Kristina Striegnitz. Referring expressions as formulas of description logic. In *Proceedings of the 5th International Natural Language Generation Conference, INLG 2008*. The Association for Computer Linguistics, 2008. URL: https://aclanthology.org/W08-1107/.

8 Alessandro Artale, Jean Christoph Jung, Andrea Mazzullo, Ana Ozaki, and Frank Wolter. Living without Beth and Craig: Definitions and interpolants in description and modal logics with nominals and role inclusions. *ACM Trans. Comput. Log.*, 24(4):34:1–34:51, 2023. doi:10.1145/3597301.

9 Alessandro Artale, Andrea Mazzullo, Ana Ozaki, and Frank Wolter. On free description logics with definite descriptions. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, pages 63–73, 2021. doi:10.24963/kr.2021/7.

10 Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 364–369. Professional Book Center, 2005. URL: http://ijcai.org/Proceedings/05/Papers/0372.pdf.

11 Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017. doi:10.1017/9781139025355.

12 Franz Baader and Werner Nutt. Basic description logics. In *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003. doi:10.1017/CBO9780511711787.

13 Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2010. doi:10.1017/CBO9780511543357.

14 Pietro Baroni, Francesca Toni, and Bart Verheij. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games: 25 years later. *Argument Comput.*, 11(1-2):1–14, 2020. doi:10.3233/AAC-200901.

15 Matti Berthold. On syntactic forgetting with strong persistence. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022*, 2022. URL: https://proceedings.kr.org/2022/5/.

16 Matti Berthold, Ricardo Gonçalves, Matthias Knorr, and João Leite. A syntactic operator for forgetting that satisfies strong persistence. *Theory Pract. Log. Program.*, 19(5-6):1038–1055, 2019. doi:10.1017/S1471068419000346.

17 Matti Berthold, Matthias Knorr, and Daphne Odekerken. Forgettingweb. In *Proceedings of the 39th International Conference on Logic Programming, ICLP 2023*, volume 385, pages 321–323. EPTCS, 2023. doi:10.4204/EPTCS.385.

18 Philippe Besnard. *An introduction to default logic*. Symbolic computation. Springer, 1989. doi:10.1007/978-3-662-05689-9.

19 Alexander Borgida, David Toman, and Grant E. Weddell. On referring expressions in query answering over first order knowledge bases. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning, KR 2016*, pages 319–328. AAAI Press, 2016. URL: http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12860.

20 Elena Botoeva, Boris Konev, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Inseparability and conservative extensions of description logic ontologies: A survey. In *Reasoning Web: Logical Foundation of Knowledge Graph Construction and Query Answering - 12th International Summer School 2016*, volume 9885 of *Lecture Notes in Computer Science*, pages 27–89. Springer, 2016. doi:10.1007/978-3-319-49493-7_2.

21 Ronald J. Brachman and Hector J. Levesque. *Knowledge Representation and Reasoning*. Elsevier, 2004. doi:10.1016/B978-1-55860-932-7.X5083-3.

22 Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011. doi:10.1145/2043174.2043195.

23 Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99*, pages 84–89. Morgan Kaufmann, 1999. URL: http://ijcai.org/Proceedings/99-1/Papers/013.pdf.

24 Ronnie Cann. *Formal Semantics: An Introduction*. Cambridge Textbooks in Linguistics. Cambridge University Press, 1993. doi:10.1017/CBO9781139166317.002.

25 Valentin Cassano, Raul Fervari, Carlos Areces, and Pablo F. Castro. Interpolation and Beth definability in default logics. In *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019*, volume 11468 of *Lecture Notes in Computer Science*, pages 675–691. Springer, 2019. doi:10.1007/978-3-030-19570-0_44.

26 Balder ten Cate. Interpolation for extended modal languages. *J. Symb. Log.*, 70(1):223–234, 2005. doi:10.2178/jsl/1107298517.

27 Balder ten Cate, Willem Conradie, Maarten Marx, and Yde Venema. Definitorially complete description logics. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning, KR 2006*, pages 79–89. AAAI Press, 2006. URL: http://www.aaai.org/Library/KR/2006/kr06-011.php.

28 Balder ten Cate, Enrico Franconi, and Inanç Seylan. Beth definability in expressive description logics. *J. Artif. Intell. Res.*, 48:347–414, 2013. doi:10.1613/JAIR.4057.

29 Giovanna D'Agostino and Marco Hollenberg. Logical questions concerning the mu-calculus: Interpolation, Lyndon and Los-Tarski. *J. Symb. Log.*, 65(1):310–332, 2000. doi:10.2307/2586539.

30 Manuel de Sousa Ribeiro and João Leite. Aligning artificial neural networks and ontologies towards explainable AI. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 4932–4940. AAAI Press, 2021. doi:10.1609/AAAI.V35I6.16626.

31 Warren Del-Pinto and Renate A. Schmidt. Abox abduction via forgetting in ALC. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 2768–2775. AAAI Press, 2019. doi:10.1609/AAAI.V33I01.33012768.

32 Kevin Donnelly et al. SNOMED-CT: The advanced terminology and coding system for eHealth. *Studies in health technology and informatics*, 121:279, 2006. URL: https://ebooks.iospress.nl/volumearticle/9130.

33 Thomas Eiter and Gabriele Kern-Isberner. A brief survey on forgetting from a knowledge representation and reasoning perspective. *Künstliche Intell.*, 33(1):9–33, 2019. doi:10.1007/S13218-018-0564-6.

34 Patrice Enjalbert and Luis Fariñas del Cerro. Modal resolution in clausal form. *Theor. Comput. Sci.*, 65(1):1–33, 1989. doi:10.1016/0304-3975(89)90137-0.

35 Jorge Fandinno and Claudia Schulz. Answering the "why" in answer set programming — A survey of explanation approaches. *Theory Pract. Log. Program.*, 19(2):114–203, 2019. doi:10.1017/S1471068418000534.

36 Marie Fortin, Boris Konev, and Frank Wolter. Interpolants and Explicit Definitions in Extensions of the Description Logic EL. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022*, pages 152–162, 8 2022. doi:10.24963/kr.2022/16.

37 Enrico Franconi and Volha Kerhet. Effective query answering with ontologies and DBoxes. In *Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*, volume 11560 of *Lecture Notes in Computer Science*, pages 301–328. Springer, 2019. doi:10.1007/978-3-030-22102-7_14.

38 Enrico Franconi, Volha Kerhet, and Nhung Ngo. Exact query reformulation over databases with first-order and description logics ontologies. *J. Artif. Intell. Res.*, 48:885–922, 2013. doi:10.1613/JAIR.4058.

39 Maurice Funk, Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. Learning description logic concepts: When can positive and negative examples be separated? In *Proceedings of the 28h International Joint Conference on Artificial Intelligence, IJCAI 2019*, pages 1682–1688, 2019. doi:10.24963/ijcai.2019/233.

40 Dov M. Gabbay and Larisa Maksimova. *Interpolation and Definability: Modal and Intuitionistic Logics*. Oxford University Press, 05 2005. doi:10.1093/acprof:oso/9780198511748.001.0001.

41 Dov M. Gabbay, David Pearce, and Agustín Valverde. Interpolable formulas in Equilibrium Logic and Answer Set Programming. *J. Artif. Intell. Res.*, 42:917–943, 2011. URL: https://www.jair.org/index.php/jair/article/view/10743/25661.

42 Dov M. Gabbay and Karl Schlechta. *A New Perspective on Nonmonotonic Logics*. Springer, 2016. doi:10.1007/978-3-319-46817-4.

43 Dov M. Gabbay, Renate A. Schmidt, and Andrzej Szalas. *Second-Order Quantifier Elimination - Foundations, Computational Aspects and Applications*, volume 12 of *Studies in logic : Mathematical logic and foundations*. College Publications, 2008. URL: http://collegepublications.co.uk/logic/mlf/?00009.

44 Fabien Gandon. *Distributed Artificial Intelligence And Knowledge Management: Ontologies And Multi-Agent Systems For A Corporate Semantic Web. (Intelligence Artificielle Distribuée Et Gestion Des Connaissances : Ontologies Et Systèmes Multi-Agents Pour Un Web Sémantique Organisationnel)*. PhD thesis, University of Nice Sophia Antipolis, France, 2002. URL: https://tel.archives-ouvertes.fr/tel-00378201.

45 Peter Gärdenfors. *Knowledge in flux: Modeling the dynamics of epistemic states*. The MIT press, 1988. doi:10.2307/2275379.

46 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012. doi:10.2200/S00457ED1V01Y201211AIM019.

47 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*, 19(1):27–82, 2019. doi:10.1017/S1471068418000054.

48 Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, pages 1070–1080. MIT Press, 1988.

49 Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3-4):365–385, 1991. doi:10.1007/BF03037169.

50 Silvio Ghilardi, Carsten Lutz, and Frank Wolter. Did I damage my ontology? A case for conservative extensions in description logics. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning, KR 2006*, pages 187–197. AAAI Press, 2006. URL: http://www.aaai.org/Library/KR/2006/kr06-021.php.

51 Ricardo Gonçalves, Tomi Janhunen, Matthias Knorr, and João Leite. On syntactic forgetting under uniform equivalence. In *Logics in Artificial Intelligence - 17th European Conference, JELIA 2021, Proceedings*, volume 12678 of *Lecture Notes in Computer Science*, pages 297–312. Springer, 2021. doi:10.1007/978-3-030-75775-5_20.

52 Ricardo Gonçalves, Tomi Janhunen, Matthias Knorr, João Leite, and Stefan Woltran. Forgetting in Modular Answer Set Programming. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 2843–2850. AAAI Press, 2019. doi:10.1609/AAAI.V33I01.33012843.

53 Ricardo Gonçalves, Matthias Knorr, and João Leite. You can't always forget what you want: On the limits of forgetting in Answer Set Programming. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 957–965. IOS Press, 2016. doi:10.3233/978-1-61499-672-9-957.

54 Ricardo Gonçalves, Matthias Knorr, and João Leite. Forgetting in answer set programming - A survey. *Theory Pract. Log. Program.*, 23(1):111–156, 2023. doi:10.1017/S1471068421000570.

55 Ricardo Gonçalves, Matthias Knorr, João Leite, and Stefan Woltran. On the limits of forgetting in Answer Set Programming. *Artif. Intell.*, 286:103–307, 2020. doi:10.1016/J.ARTINT.2020.103307.

56 Valentin Goranko and Martin Otto. Model theory of modal logic. In *Handbook of Modal Logic*, pages 249–329. Elsevier, 2007. doi:10.1016/S1570-2464(07)80008-5.

57 Bernardo Cuenca Grau. Privacy in ontology-based information systems: A pending matter. *Semantic Web*, 1(1-2):137–141, 2010. doi:10.3233/SW-2010-0009.

58 Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular reuse of ontologies: Theory and practice. *J. Artif. Intell. Res.*, 31:273–318, 2008. doi:10.1613/JAIR.2375.

59 Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of Knowledge Representation*. Elsevier Science, San Diego, CA, USA, 2007.

60 Andreas Herzig and Jérôme Mengin. Uniform interpolation by resolution in modal logic. In *Logics in Artificial Intelligence, 11th European Conference, JELIA 2008*, volume 5293 of *Lecture Notes in Computer Science*, pages 219–231. Springer, 2008. doi:10.1007/978-3-540-87803-2_19.

61 Jan Heuer and Christoph Wernhard. Synthesizing strongly equivalent logic programs: Beth definability for answer set programs via Craig interpolation in first-order logic. In *Automated Reasoning - 12th International Joint Conference, IJCAR 2024*, volume 14739 of *Lecture Notes in Computer Science*, pages 172–193. Springer, 2024. doi:10.1007/978-3-031-63498-7_11.

62 Arend Heyting. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften. Physikalisch-mathematische Klasse*, pages 42–56, 1930.

63 Eva Hoogland and Maarten Marx. Interpolation and definability in guarded fragments. *Studia Logica*, 70(3):373–409, 2002. doi:10.1023/A:1015154431342.

64 Eva Hoogland, Maarten Marx, and Martin Otto. Beth definability for the guarded fragment. In *Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning, LPAR 1999*, pages 273–285. Springer, 1999. doi:10.1007/3-540-48242-3_17.

65 Ian Horrocks. Ontologies and the semantic web. *Commun. ACM*, 51(12):58–67, 2008. doi:10.1145/1409360.1409377.

66 Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: the making of a web ontology language. *J. Web Semant.*, 1(1):7–26, 2003. doi:10.1016/J.WEBSEM.2003.07.001.

67 Jean Christoph Jung, Jędrzej Kołodziejski, and Frank Wolter. Computation of interpolants for description logic concepts in hard cases. In *Proceedings of the 38th International Workshop on Description Logics (DL 2025)*. CEUR-WS.org, 2025.

68 Jean Christoph Jung, Carsten Lutz, Mauricio Martel, and Thomas Schneider. Conservative extensions in horn description logics with inverse roles. *J. Artif. Intell. Res.*, 68:365–411, 2020. doi:10.1613/JAIR.1.12182.

69 Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. Separating data examples by description logic concepts with restricted signatures. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, pages 390–399, 2021. doi:10.24963/kr.2021/37.

70 Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. Logical separability of labeled data examples under ontologies. *Artif. Intell.*, 313:103785, 2022. doi:10.1016/j.artint.2022.103785.

71 Jean Christoph Jung and Frank Wolter. Living without Beth and Craig: Definitions and interpolants in the guarded and two-variable fragments. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021*, pages 1–14. IEEE, 2021. doi:10.1109/LICS52264.2021.9470585.

72 Matthias Knorr and José Júlio Alferes. Preserving strong equivalence while forgetting. In *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014. Proceedings*, volume 8761 of *Lecture Notes in Computer Science*, pages 412–425. Springer, 2014. doi:10.1007/978-3-319-11558-0_29.

73 Daphne Koller and Nir Friedman. *Probabilistic Graphical Models — Principles and Techniques*. MIT Press, 2009. URL: https://mitpress.mit.edu/9780262013192/probabilistic-graphical-models/.

74 Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Formal properties of modularisation. In *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*, pages 25–66. Springer, 2009. doi:10.1007/978-3-642-01907-4_3.

75 Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artif. Intell.*, 203:66–103, 2013. doi:10.1016/J.ARTINT.2013.07.004.

76 Boris Konev, Dirk Walther, and Frank Wolter. The logical difference problem for description logic terminologies. In *Automated Reasoning, 4th International Joint Conference, IJCAR 2008*, volume 5195 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2008. doi:10.1007/978-3-540-71070-7_21.

77 Boris Konev, Dirk Walther, and Frank Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009*, pages 830–835, 2009. URL: http://ijcai.org/Proceedings/09/Papers/142.pdf.

78 Bogumil M. Konopka. Biomedical ontologies—a review. *Biocybernetics and Biomedical Engineering*, 35(2):75–86, 2015. doi:10.1016/j.bbe.2014.06.002.

79 Patrick Koopmann. *Practical uniform interpolation for expressive description logics*. PhD thesis, University of Manchester, UK, 2015. URL: https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.674705.

80 Patrick Koopmann. LETHE: forgetting and uniform interpolation for expressive description logics. *Künstliche Intell.*, 34(3):381–387, 2020. doi:10.1007/S13218-020-00655-W.

81 Patrick Koopmann and Jieying Chen. Deductive module extraction for expressive description logics. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1636–1643. ijcai.org, 2020. doi:10.24963/IJCAI.2020/227.

82 Patrick Koopmann, Warren Del-Pinto, Sophie Tourret, and Renate A. Schmidt. Signature-based abduction for expressive description logics. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020*, pages 592–602, 2020. doi:10.24963/KR.2020/59.

83 Patrick Koopmann and Renate A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In *Proceedings of 19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2019*, volume 8312 of *Lecture Notes in Computer Science*, pages 552–567. Springer, 2013. doi:10.1007/978-3-642-45221-5_37.

84 Emiel Krahmer and Kees van Deemter. Computational generation of referring expressions: A survey. *Comput. Linguistics*, 38(1):173–218, 2012. doi:10.1162/COLI_A_00088.

85 Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. Definitions and (uniform) interpolants in first-order modal logic. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023*, pages 417–428, 2023. doi:10.24963/KR.2023/41.

86 Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. The interpolant existence problem for weak K4 and difference logic. In *Advances in Modal Logic, AiML 2024*, pages 465–484. College Publications, 2024.

87 Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78:203–250, 2010. doi:10.1007/S10994-009-5146-2.

88 Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Inconsistency-tolerant semantics for description logics. In *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010*, volume 6333 of *Lecture Notes in Computer Science*, pages 103–117. Springer, 2010. doi:10.1007/978-3-642-15918-3_9.

89 Vladimir Lifschitz. What is answer set programming? In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1594–1597. AAAI Press, 2008. URL: http://www.aaai.org/Library/AAAI/2008/aaai08-270.php.

90 Vladimir Lifschitz. *Answer Set Programming*. Springer, 2019. doi:10.1007/978-3-030-24658-7.

91 Vladimir Lifschitz, David Pearce, and Agustín Valverde. A characterization of strong equivalence for logic programs with variables. In *Proceedings of the 9th Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2022*, volume 4483 of *Lecture Notes in Computer Science*, pages 188–200. Springer, 2007. doi:10.1007/978-3-540-72200-7_17.

92 Fangzhen Lin and Raymond Reiter. Forget it! In *Proceedings of Twelfth National Conference on Artificial Intelligence, AAAI 1994*, 1994. URL: https://cdn.aaai.org/Symposia/Fall/1994/FS-94-02/FS94-02-037.pdf.

93 Zhao Liu, Chang Lu, Ghadah Alghamdi, Renate A. Schmidt, and Yizheng Zhao. Tracking semantic evolutionary changes in large-scale ontological knowledge bases. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management*, pages 1130–1139. ACM, 2021. doi:10.1145/3459637.3482307.

94 Michel Ludwig and Boris Konev. Practical uniform interpolation and forgetting for ALC tboxes with applications to logical difference. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning, KR 2014*. AAAI Press, 2014. URL: http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7985.

95 Michel Ludwig and Dirk Walther. Towards a practical decision procedure for uniform interpolants of el-tboxes - a proof-theoretic approach. In *Proceedings of 2nd Global Conference on Artificial Intelligence, GCAI 2016*, volume 41 of *EPiC Series in Computing*, pages 147–160. EasyChair, 2016. doi:10.29007/BSQM.

96 Carsten Lutz. Complexity and succinctness of public announcement logic. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pages 137–143. ACM, 2006. doi:10.1145/1160633.1160657.

97 Carsten Lutz, Inanç Seylan, and Frank Wolter. An automata-theoretic approach to uniform interpolation and approximation in the description logic EL. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning, KR 2012*. AAAI Press, 2012. URL: http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4511.

98 Carsten Lutz, Inanç Seylan, and Frank Wolter. The data complexity of ontology-mediated queries with closed predicates. *Logical Methods in Computer Science*, 15(3), 2019. doi:10.23638/LMCS-15(3:23)2019.

99 Carsten Lutz, Dirk Walther, and Frank Wolter. Conservative extensions in expressive description logics. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*, pages 453–458, 2007. URL: http://ijcai.org/Proceedings/07/Papers/071.pdf.

100 Carsten Lutz and Frank Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *J. Symb. Comput.*, 45(2):194–228, 2010. doi:10.1016/J.JSC.2008.10.007.

101 Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pages 989–995. IJCAI/AAAI, 2011. doi:10.5591/978-1-57735-516-8/IJCAI11-170.

102 Timothy S. Lyon and Jonas Karge. Constructive interpolation and concept-based Beth definability for description logics via sequents. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 3484–3492. International Joint Conferences on Artificial Intelligence Organization, 8 2024. Main Track. doi:10.24963/ijcai.2024/386.

103 Larisa Maksimova. Intuitionistic logic and implicit definability. *Ann. Pure Appl. Log.*, 105(1-3):83–102, 2000. doi:10.1016/S0168-0072(99)00049-4.

104 Larissa Maksimova. Craig's interpolation theorem and amalgamable varieties. *Doklady Akademii Nauk SSSR*, 237(6):1281–1284, 1977. Translated to Soviet Math. Doklady.

105 John McCarthy. Circumscription — A form of non-monotonic reasoning. *Artif. Intell.*, 13(1-2):27–39, 1980. doi:10.1016/0004-3702(80)90011-9.

106 Deborah L. McGuinness and Alexander Borgida. Explaining subsumption in description logics. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95*, pages 816–821. Morgan Kaufmann, 1995. URL: http://ijcai.org/Proceedings/95-1/Papers/105.pdf.

107 Julián Méndez, Christian Alrabbaa, Patrick Koopmann, Ricardo Langner, Franz Baader, and Raimund Dachselt. Evonne: A visual tool for explaining reasoning with OWL ontologies and supporting interactive debugging. *Comput. Graph. Forum*, 42(6), 2023. doi:10.1111/CGF.14730.

108 Nadeschda Nikitina and Birte Glimm. Hitting the sweetspot: Economic rewriting of knowledge bases. In *Proceedings of the 11th International Semantic Web Conference, ISWC 2012*, volume 7649 of *Lecture Notes in Computer Science*, pages 394–409. Springer, 2012. doi:10.1007/978-3-642-35176-1_25.

109 Nadeschda Nikitina and Sebastian Rudolph. (Non-)succinctness of uniform interpolants of general terminologies in the description logic EL. *Artif. Intell.*, 215:120–140, 2014. doi:10.1016/J.ARTINT.2014.06.005.

110 Andreas Nonnengart, Hans Jürgen Ohlbach, and Andrzej Szałas. Elimination of predicate quantifiers. *Logic, Language and Reasoning: Essays in honour of Dov Gabbay*, pages 149–171, 1999. doi:10.1007/978-94-011-4574-9_9.

111 Hans Jürgen Ohlbach. SCAN - elimination of predicate quantifiers. In *Automated Deduction - CADE-13, 13th International Conference on Automated Deduction*, volume 1104 of *Lecture Notes in Computer Science*, pages 161–165. Springer, 1996. doi:10.1007/3-540-61511-3_77.

112 Hiroakira Ono. Model extension theorem and craig's interpolation theorem for intermediate predicate logics. *Rep. Math. Log.*, 15:41–58, 1983.

113 Ana Ozaki. Learning description logic ontologies: Five approaches. where do they stand? *Künstliche Intell.*, 34(3):317–327, 2020. doi:10.1007/S13218-020-00656-9.

114 David Pearce. A new logical characterisation of stable models and answer sets. In *Proceedings of the Second Workshop on Non-Monotonic Extensions of Logic Programming NMELP 96*, volume 1216 of *Lecture Notes in Computer Science*, pages 57–70. Springer, 1996. doi:10.1007/BFB0023801.

115 Abraham Robinson. A result on consistency and its application to the theory of definition. *Journal of Symbolic Logic*, 25(2):174–174, 1960. doi:10.2307/2964240.

116 Ana Armas Romero, Bernardo Cuenca Grau, and Ian Horrocks. More: Modular combination of OWL reasoners for ontology classification. In *Proceedings of the 11th International Semantic Web Conference, ISWC 2012*, volume 7649 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012. doi:10.1007/978-3-642-35176-1_1.

117 Zeynep Saribatur and Thomas Eiter. Omission-based abstraction for answer set programs. *Theory Pract. Log. Program.*, 21(2):145–195, 2021. doi:10.1017/S1471068420000095.

118 Zeynep G. Saribatur, Matthias Knorr, Ricardo Gonçalves, and João Leite. On abstracting over the irrelevant in answer set programming. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024*, 2024. doi:10.24963/KR.2024/61.

119 Zeynep G. Saribatur and Stefan Woltran. Foundations for projecting away the irrelevant in ASP programs. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023*, pages 614–624, 2023. doi:10.24963/KR.2023/60.

120 Zeynep Gozen Saribatur, Thomas Eiter, and Peter Schüller. Abstraction for non-ground answer set programs. *Artif. Intell.*, 300:103563, 2021. doi:10.24963/IJCAI.2022/807.

121 Karl Schlechta. *Formal Methods for Nonmonotonic and Related Logics — Vol II: Theory Revision, Inheritance, and Various Abstract Properties*. Springer, 2018. doi:10.1007/978-3-319-89650-2.

122 Stefan Schlobach. Explaining subsumption by optimal interpolation. In *Proceedings of 9th European Conference on Logics in Artificial Intelligence, JELIA 2004*, volume 3229 of *Lecture Notes in Computer Science*, pages 413–425. Springer, 2004. doi:10.1007/978-3-540-30227-8_35.

123 Inanç Seylan, Enrico Franconi, and Jos de Bruijn. Effective query rewriting with ontologies over dboxes. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009*, pages 923–925, 2009. URL: http://ijcai.org/Proceedings/09/Papers/157.pdf.

124 Stephan Tobies. *Complexity results and practical algorithms for logics in knowledge representation*. PhD thesis, RWTH Aachen University, Germany, 2001. URL: https://publications.rwth-aachen.de/record/52234.

125 David Toman and Grant E. Weddell. First-order rewritability in ontology-mediated querying in horn description logics. In *Proceedings of Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 5897–5905. AAAI Press, 2022. doi:10.1609/AAAI.V36I5.20534.

126 Ly Ly T. Trieu, Tran Cao Son, and Marcello Balduccini. xASP: An explanation generation system for answer set programming. In *Proceedings of the 16th Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2022*, volume 13416 of *Lecture Notes in Computer Science*, pages 363–369. Springer, 2022. doi:10.1007/978-3-031-15707-3_28.

127 Yisong Wang, Kewen Wang, and Mingyi Zhang. Forgetting for answer set programs revisited. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*, pages 1162–1168. IJCAI/AAAI, 2013. URL: https://www.ijcai.org/Proceedings/13/Papers/175.pdf.

128 Yisong Wang, Yan Zhang, Yi Zhou, and Mingyi Zhang. Forgetting in logic programs under strong equivalence. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012*. AAAI Press, 2012. URL: http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4389.

129 Yisong Wang, Yan Zhang, Yi Zhou, and Mingyi Zhang. Knowledge forgetting in Answer Set Programming. *J. Artif. Intell. Res.*, 50:31–70, 2014. doi:10.1613/JAIR.4297.

130 Zhe Wang, Kewen Wang, Rodney W. Topor, and Jeff Z. Pan. Forgetting for knowledge bases in DL-Lite. *Ann. Math. Artif. Intell.*, 58(1-2):117–151, 2010. doi:10.1007/S10472-010-9187-9.

131 Zhe Wang, Kewen Wang, Rodney W. Topor, and Xiaowang Zhang. Tableau-based forgetting in $\mathcal{ALC}$ ontologies. In *Proceedings of 19th European Conference on Artificial Intelligence, ECAI 2010*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 47–52. IOS Press, 2010. URL: https://ebooks.iospress.nl/volumearticle/5737.

132 Hui Yang, Patrick Koopmann, Yue Ma, and Nicole Bidoit. Efficient computation of general modules for ALC ontologies. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 3356–3364. ijcai.org, 2023. doi:10.24963/IJCAI.2023/374.

133 Hui Yang, Yue Ma, and Nicole Bidoit. Efficient extraction of el-ontology deductive modules. In *Proceedings of 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, pages 6575–6582. AAAI Press, 2023. doi:10.1609/AAAI.V37I5.25808.

134 Yizheng Zhao and Renate A. Schmidt. FAME(Q): an automated tool for forgetting in description logics with qualified number restrictions. In *Proceedings of the 27th International Conference on Automated Deduction, CADE 2019*, volume 11716 of *Lecture Notes in Computer Science*, pages 568–579. Springer, 2019. doi:10.1007/978-3-030-29436-6_34.

Chapter 16

Definability and Interpolation in Philosophy

Johan van Benthem
University of Amsterdam, The Netherlands
Stanford University, USA
Tsinghua University, China

Abstract

This paper offers a light tour of occurrences of Craig's interpolation theorem and Beth's definability theorem in philosophy. There is no coherent research program with a central agenda around this theme in the literature, but extending the discussion in [9], we trace some lines and identify some issues in establishing contacts at this interface between the realms of logic and the philosophy of logic, of science, of language and information, epistemology, and metaphysics. Our presentation can also be viewed as a case study for more general aspects of the logic–philosophy interface.

Contents

1 The Broad Ideas 590
2 What is a Logical System? 592
3 Supervenience 596
4 Determinism 598
5 Questions 600
6 Dependence 601
7 Information and Generalized Consequence 603
8 Theoretical Terms in Scientific Theories 606
9 Modularity in Knowledge and Belief 607
10 A Historical Precursor: Bolzano 609
11 Conclusion 610
Acknowledgments 612
References 612



© Johan van Benthem;
licensed under Creative Commons License CC BY 4.0
In Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter 16, pages 589–614. Ubiquity Press, 2026.
https://www.doi.org/10.5334/bdg.p

1 The Broad Ideas

Beth's Definability Theorem says that, in first-order theories, implicit definability of vocabulary is equivalent with explicit definability. More precisely, given a first-order theory T in a language with vocabulary $\boldsymbol{L}$ plus some predicate letter P, implicit definability says that, given two models M, M' for T which are identical qua domain of objects and interpretation of the vocabulary of $\boldsymbol{L}$, the predicate interpretations P^M and $P^{M'}$ must also be identical. Put more loosely, fixing the interpretation of $\boldsymbol{L}$ in a model for T automatically also fixes that of P. In contrast, explicit definability says that there exists a first-order formula $\varphi(\boldsymbol{x})$ using only the vocabulary $\boldsymbol{L}$ with a tuple of variables $\boldsymbol{x}$ of the arity of the predicate P such that the theory T implies the formula $\forall \boldsymbol{x}\,(P\boldsymbol{x} \leftrightarrow \varphi(\boldsymbol{x}))$.

▶ **Theorem 1.** *For first-order theories, implicit and explicit definability are equivalent.*

Clearly, explicit definability implies implicit definability, but Beth's achievement was to show the converse, which is not at all trivial for first-order logic [18].

The background to Beth's Theorem was his interest in logic as a joint study of definability and proof, a tandem view of the history of the discipline since Antiquity on which he elaborated in various publications, in particular his magnum opus [19] which also contains his more general philosophical views based on logical foundational research. In particular, Beth arrived at the Definability Theorem by reflecting on Padoa's Method for disproving explicit definability in the foundations of geometry. This combination of logic and philosophy of science was characteristic for the 1950s. Beth founded the "Instituut voor Logica en Grondslagen van de Exacte Wetenschappen" (Institute for Logic and Philosophy of the Exact Sciences), the precursor of the current Institute for Logic, Language and Computation in Amsterdam, and equally significantly, around the same time Beth's correspondent and ally Alfred Tarski founded the still existing Group in Logic and Methodology of the Exact Sciences at the University of California at Berkeley.

For our later discussion, it is convenient to rephrase the earlier formulation of Beth's notion to the following equivalent version of implicit definability:

> If two models M, N both make $T(\boldsymbol{L}, P)$ true and f is an $\boldsymbol{L}$-isomorphism between M and N, then f is also a P-isomorphism.

This isomorphism-lifting version of Beth's Theorem generalizes to weaker structure invariances such as potential isomorphism,[1] a notion that will return in Section 7 below.

Various notions of implicit and explicit determination in the above vein can be found in many areas of philosophy.

---

[1] A *potential isomorphism* between two models M, N is a family $\boldsymbol{F}$ of finite partial isomorphisms between these two models satisfying the following Back-and-Forth conditions. For every $f \in \boldsymbol{F}$ and $a \in M$ there is a $b \in N$ such that $f \cup \{(a, b)\} \in \boldsymbol{F}$, and the same condition holds in the opposite direction.

Next let us consider the Craig Interpolation Theorem for first-order logic, [25, 26], using the following simple formulation displaying just a few predicates for convenience.

▶ **Theorem 2.** *If formula $\varphi(P,Q)$ in predicate vocabulary $\{P,Q\}$ semantically implies formula ψ with vocabulary $\{Q,R\}$, then there exists a first-order formula α using only the shared vocabulary $\{Q\}$ such that $\varphi(P,Q) \models \alpha(Q) \models \psi(Q,R)$.*

More elaborate versions of the Interpolation Theorem, of which there exist quite a few by now, cf. [41] for a sample, give more detailed information about how the shared predicate Q occurs syntactically in α in terms of how it occurs in the formulas φ and ψ. See Chapter 2 in this book for more details and in-depth results.

Note. In various chapters of the present volume, a distinction is made between plain and deductive interpolation as versions of the preceding result which are not always equivalent for logical systems. While important, this distinction will not be highlighted in what follows.

Craig Interpolation implies Beth Definability for first-order logic by a well-known simple argument. The converse does not hold. For instance, the natural Guarded Fragment of the first-order language satisfies the latter property, but not the former, cf. [36, 35].[2] For more information on these matters, see also Chapter 10 in this book.

By an easy argument, Interpolation also implies the stronger Projective Beth Theorem which says the following, in a simplified notation. If the theory $T(A,B,C) \cup T(A,B',C')$ implies $\forall x\,(Cx \leftrightarrow C'x)$, then $T(A,B,C)$ implies an explicit definition for C by a first-order formula using the vocabulary A only. We will meet this stronger version occasionally.

We will also encounter applications of Interpolation in philosophy, though at first glance it may look like a more technical system property. For greater unity in philosophical applications, it would be nice if one could capture the surplus of interpolation over Beth in a natural and appealing manner. We will not resolve this matter here.[3]

We are going to show how the above two circles of ideas return in philosophy. Section 2 is about the role of the two theorems in defining the notion of a logical system, a live topic in the philosophy of logic. After that, the focus is on Beth's Theorem in Sections 3–6 on supervenience, determinism, and reasoning about questions, culminating in the suggestion that the common logical thread in philosophical applications of Beth's Theorem is the role of implicit and explicit *dependence*.

Then the focus of our narrative shifts to Craig's Theorem and its role in highlighting the importance of *vocabulary* in reasoning and theory construction, in Sections 7–9 on generalized notions of logical inference, the architecture of scientific theories, and the modular structure

---

[2] The first of these papers makes a case that the Guarded Fragment does satisfy the right version of interpolation from a 'modal' perspective – an issue that we do not go into here.

[3] One candidate for a mathematical notion that might throw light on this comparative surplus is the *amalgamation property* in varieties of algebras, perhaps in a generalized category-theoretic setting.

of beliefs. Finally, Section 10 flags some connections with the work of Bernard Bolzano, a creative and somewhat neglected 19th century pioneer in the emergence of modern logic.

2 What is a Logical System?

One immediate road into probing the importance of our two theorems is their relevance to a recurrent issue in the philosophy of logic, and just as well in logic itself.

What is a logical system?, or in a more refined version:
What is a good, or well-designed, logical system?

This is not an abstract quibble, at stake here is the proper view of what modern logic in its prolific system-building guise is about. What do our two theorems have to say here?

Our discussion will focus on the Interpolation Theorem, but the Definability Theorem plays a similar role. Satisfying a Craig-style interpolation theorem is sometimes seen as a key desideratum in the folklore, as it expresses a form of 'expressive completeness' of a logic for its language, as being able to formulate definitions and state crucial 'transitions' from premises to conclusions. Even so, this intuition is not easy to make precise beyond just restating what the theorems say, and among formal approaches that go further, I only know of [31] and an earlier proof-theoretic analysis in [50]. A general semantic way of seeing the centrality of Beth and Interpolation might go via their role in Abstract Model Theory, where Beth's Definability Theorem seems to follow from Lindström's Theorem for first-order logic under quite general conditions, though the case of Craig is less clear – cf. Chapter 7 in this volume.[4]

But here is another way into the above question, restated as follows with a flavor of the crucial notion of invariance that we already noted in connection with Beth's Theorem:

When are two logical systems the same?

One answer, found in both the mathematical and the philosophical literature, is this. The two logics $\mathfrak{L}_1$, $\mathfrak{L}_2$ should have mutual faithful syntactic *translations* or embeddings, in the sense that the following equivalence holds for all formulas under some syntactic translation τ for the languages of these logics:

$$\varphi \models \psi \text{ in } \mathfrak{L}_1 \quad \text{iff} \quad \tau(\varphi) \models \tau(\psi) \text{ in } \mathfrak{L}_2$$

and vice versa in the opposite direction from $\mathfrak{L}_2$ to $\mathfrak{L}_1$ for some syntactic translation σ.

This is just a start, however, and there is no philosophical consensus on what more might be required for a good notion of system identity. Here is one possible richer view.

[4] One might question this criterion given the ubiquity of failures of interpolation in important systems found in philosophical logic, cf. Chapter 3 in this book. This issue will return in Section 3 below.

In Abstract Model Theory, logical systems come with both abstract sentences and abstract semantic structures, connected by a truth relation. Accordingly, the natural notion of translation then also involves a semantic transformation $\mathbb{T}$ of models running in the opposite direction to the above syntactic map τ, leading to the following contravariant equivalence:

$$M \models \tau(\varphi) \quad \text{iff} \quad \mathbb{T}(M) \models \varphi \qquad (\#)$$

This equivalence views translation as having two components: connecting languages and connecting models for two logics. This dual perspective is the basic mechanism of information flow in Situation Theory [5] – see Section 7 below for more on this framework.

The abstract mathematical theory of this setting involves adjoint maps in Chu Spaces [8], an elegant category-theoretic perspective that we will not pursue here.[5]

The broader issue at stake here is the following. Looking under the surface of the myriads of logical systems found in the literature, what are the underlying invariances and connections?[6] In other words, what is the coherence of the field of the logic? And again, do Craig and Beth have a bearing on this?

Without entering into a general philosophical discussion of these issues, here is one connection with our main topic in this paper. If Interpolation is made an integral part of the definition of an abstract logic, *what constraint is induced on translations* between logical systems? Here, to formulate the interpolation property in its proper generality, we need abstract notions of 'sentences', 'consequence', 'vocabulary', and 'occurrence'. Proper translations should now respect this basic structure, and make sure that interpolation is transferred from one logical system to any properly translation-related logical system, a concrete constraint which now becomes a more precise issue.[7]

Still, making Interpolation key to the identity of logical systems and translations between them does not establish just *why* one should consider this property so important. [12] conjectured that Interpolation is the key axiom in a *complete axiomatization of the meta-theory* of first-order logic, viewed as a two-sorted structure consisting of sentences and finite vocabulary sets with relations of logical consequence and 'occurrence'. The thinking behind this conjecture was that Craig interpolation seems the last major property of first-order logic discovered in modern history.[8] However, [38] showed that even the meta-theory of classical

---

[5] Faithfulness of the translation requires extra conditions, such as surjectivity of the transformation T.

[6] One famous connection is the faithful *Gödel translation* from intuitionistic propositional logic into the classical modal logic S4 which has led to a wide variety of fruitful contacts between the two systems.

[7] As a concrete illustration of this transfer, [11] show how a pair consisting of suitable homomorphic translation from the Guarded Fragment into the Modal Dependence Logic LFD mentioned in Section 9 below, plus a converse satisfiability reduction yields the Interpolation Theorem for LFD from the earlier-mentioned Beth Theorem for the Guarded Fragment. More such case studies are needed to see more clearly which constraints are imposed by transfer of Interpolation.

[8] Of course, this historical observation is no longer true when we consider the remarkable probabilistic properties of first-order logic discovered later, such as the Zero-One Law, [29].

propositional logic construed in this way is *non-axiomatizable*, and in fact admits of a faithful embedding of True Arithmetic. Thus, the central role of Interpolation does not show in this particular construal of the structure of logical systems – but perhaps all this shows is that we have made the notion of 'meta-theory' too concrete, importing peculiarities of the natural numbers and the complexity of their theory.

We conclude this section with another link to our main theme that may be of interest. The topic of translations also raises a nice connection with the Beth Definability Theorem, and in fact, it suggests a more general take on what this theorem said about definability.

Translation as Generalized Definability. One can think of an explicit *syntactic translation* τ satisfying the above-mentioned equivalence (#) as providing a *generalized explicit definition* for the basic notions of the logic $\mathfrak{L}_1$, not inside of its own vocabulary (the special case of definability at issue in Beth's Theorem) but in terms of formulas in another language: namely, that of the logic $\mathfrak{L}_2$. Accordingly, this time, the corresponding notion of *implicit definability* is not in just one model of the same similarity type, but it runs across possibly different models related by the semantic model-transforming component of the translation.

Here is a first intuitive formulation of this idea:

> Fixing the $\mathfrak{L}_2$-structure of a model M suffices for fixing the denotation of the basic predicates of the logic $\mathfrak{L}_1$ in all $\mathbb{T}$-matching models $\mathbb{T}(M)$ for transformations $\mathbb{T}$ satisfying some suitable conditions.

More precisely, let the transformation $\mathbb{T}$ in the above definability schema (#) be defined as a surjective function from $\mathfrak{L}_2$-models to $\mathfrak{L}_1$-models which also maps the domain of a $\mathfrak{L}_2$-model onto the domain of the matching $\mathfrak{L}_1$-model, i.e., $\mathbb{T}$ also maps objects to objects. *Please note:* this is surjectivity for objects in domains of $\mathbb{T}$-correlated models, not for $\mathbb{T}$ viewed as a map from models to models themselves.

It is easy to see that the schema (#) then implies the following cross-model condition, which we state here for unary predicates only, purely for convenience:

> If $\mathbb{T} : M \to N$ with $\mathbb{T}(b) = a$, M is $\mathfrak{L}_2$-isomorphic to M' with b matching b', and $M' \to N'$ with $\mathbb{T}(b') = a'$, then for any $\mathfrak{L}_1$-predicate P, $P^M(a)$ implies $P^{M'}(a')$.

Technical Elaboration. The setting of generalized defintitions invites a use of the Projective Beth Theorem, or rather, the Interpolation Theorem behind it. It applies to yield an explicit translation from an implicit one, provided the above transformation $\mathbb{T}$ is defined in first-order terms. Let this definition take the form of a first-order theory Σ with vocabulary $\{\boldsymbol{L}_1, \boldsymbol{L}_2, T, M, N\}$.[9] Here M, N stand for disjoint unary predicates, the $\boldsymbol{L}_1$-predicates (the

[9] For convenience, we use the earlier symbols M, N that denoted whole models as unary predicate letters for domains of models now, and the binary predicate letter T as a name for the transformation $\mathbb{T}$. The latter local notation will not clash with our use of T for theories in other passages in this paper.

vocabulary of the logic $\mathfrak{L}_1$) can only hold inside the subset defined by N, the $\boldsymbol{L}_2$-predicates (from the logic $\mathfrak{L}_2$) can only hold inside the subset defined by M, and where Σ only makes assertions about objects in $M \cup N$, even if a model itself is larger.[10]

Now we state the extended implicit definability introduced here as a first-order consequence in an enriched language, where the symbol P is a predicate from $\boldsymbol{L}_1$ (taken to be unary again, just for simplicity), a, a', b, b' are individual constants, Z stands for a binary relation:

$\Sigma(\boldsymbol{L}_1, \boldsymbol{L}_2, T, M, N)$, $Tb = a$, Pa (i), and
"Z is an $\boldsymbol{L}_2$-isomorphism between M and M'", $b\,Z\,b'$ (ii), and
$\Sigma(\boldsymbol{L}_1', \boldsymbol{L}_2', T', M', N')$, $T'b' = a'$ (iii)
$\models\ P'a'$.

This says that, for two pairs of $\mathbb{T}$-related models, with one source model $\boldsymbol{L}_2$-isomorphic to the other, the target models will agree on the interpretation of the $\boldsymbol{L}_1$-predicate P.

By a standard appeal to the Compactness Theorem for first-order logic, the total set of formulas occurring in the clauses (i), (ii) and (iii) here can be taken to be finite, and with some abuse of notation we can rewrite this consequence to one that holds between formulas:

$\Sigma(\boldsymbol{L}_1, \boldsymbol{L}_2, T, M, N) \wedge Tb = a \wedge Pa\ \models$ ("Z is an $\boldsymbol{L}_2$-isomorphism between M and M'"
$\wedge\, b\,Z\,b' \wedge \Sigma(\boldsymbol{L}_1', \boldsymbol{L}_2', T', M', N') \wedge T'b' = a') \rightarrow P'a'$.

Here the only overlap in vocabulary between the antecedent and consequent formulas is the set $\{M, \boldsymbol{L}_2, b\}$. Therefore, there is an interpolant $\delta(M, \boldsymbol{L}_2, b)$ in this vocabulary which just describes a property of the object b in M-type models.[11]

By some standard manipulation, we then see that the following two assertions hold:

(a) $\Sigma(\boldsymbol{L}_1, \boldsymbol{L}_2, T, M, N)\ \models\ Tb = a \rightarrow (Pa \rightarrow \delta)$.
(b) "Z is an $\boldsymbol{L}_2$-isomorphism between M and M'" $\wedge\, b\,Z\,b' \wedge \Sigma(\boldsymbol{L}_1', \boldsymbol{L}_2', T, M', N')\ \models$
$Tb' = a' \rightarrow (\delta \rightarrow P'a')$.

Now clearly, the identity relation on M satisfies all conditions for an $\boldsymbol{L}_2$-isomorphism between M and $M' = M$. Substituting this for the predicate Z in clause (b), while renaming some predicate variables, implies

$\Sigma(\boldsymbol{L}_1, \boldsymbol{L}_2', T, M, N)\ \models\ Tb = a \rightarrow (\delta \rightarrow Pa)$.

Thus, in all, the theory Σ implies the explicit definition

$Ta = b \rightarrow (Pa \leftrightarrow \delta(a, b))$.

---

[10] This effect can be achieved by assuming that all quantifiers in Σ are bounded by $M \vee N$. The 'book-keeping conditions' stated here do not seem overly restrictive for defining a function between models.

[11] Reducing to this syntactic form may take some first-order manipulation, noting that quantification over N in the formula δ has no essential effect since no $\boldsymbol{L}_2$-predicates can be true there.

Finally, given $\boldsymbol{L}_2$-definitions for the predicates in $\boldsymbol{L}_1$, a standard inductive proof lifts this to an explicit *compositional translation* for all formulas of $\mathfrak{L}_1$. *Note:* Here the quantifier step of the formula induction uses the above assumption that the transformation $\mathbb{T}$ comes with a surjective function from the domain of M onto that of N.[12]

Of course, the preceding construal does not yet analyze the notion of 'translatability' as a relation between logical systems, which does not start from a given semantic transformation. For an abstract analysis of the latter sort, cf. [16].

Note that our discussion in this section has focused on isomorphism as the central semantic invariance. This is just one choice, and for instance, in Section 7, we will also use the weaker notion of 'potential isomorphism' introduced earlier. But in principle, our discussion would also apply to a whole spectrum of invariance notions, including still weaker ones such as *bisimulation*, in which case the results we cited would be talking about definability in *modal languages* rather than in full-fledged first-order languages.

In summary, this section has established some connections between our key theorems and issues in the philosophy of logic that may be worth reflecting on, but no definite answers were provided that could not be challenged. This sort of delicate contact is typical for the logic-philosophy interface, and we will encounter further instances as we proceed.

3 Supervenience

Our next topic starts from semantic invariance relations and their liftings. In philosophy, the notion of *supervenience* of a predicate Q on one or more predicates P is ubiquitous. Examples with broad sweep are the supervenience of our mental states on our physical/biological states, of the moral aspects on the factual aspects of a scenario, and so on.

For an excellent survey, we refer to the Encyclopedia article [39], from which we take just a few notions to make our main points. For a start, let us say that Q *supervenes on* P if

Whenever two situations are P-equivalent, they are also Q-equivalent.

We state this for situations, but we can also think of supervenience as comparing objects, or whatever relevant entity we want to consider. One reason why supervenience is attractive is that it expresses a 'harmony' qua properties between two different realms, but not necessarily brute-force *reducibility* of Q to P via some explicit definition 'explaining Q away'.

Of course, formulations like this require further precision to get more traction: which realm of situations or objects is considered, which notion of semantic equivalence is meant?

[12] Another construal of implicit definability would fix the semantic transformation $\mathbb{T}$ in the assertions about M, N and M', N'. Interpolation still applies in the above style, but we now also get undesirable definitions for the predicate P that do not just refer to $\boldsymbol{L}_2$, but also to the function $\mathbb{T}$, as in "Pa holds iff exactly 2 objects in M get $\mathbb{T}$-mapped onto a".

But even without specifying these parameters further, it should be obvious that supervenience has the same flavor as implicit definability in the sense of Beth's Theorem.[13]

In fact, more can be said, since several varieties of supervenience, weaker and stronger, have been proposed in the philosophical literature. Here are just two samples.

In a *local version*, the formulation that we started with compares equivalent or similar items inside one given world. Transposing this to a model-theoretic formulation, one way of expressing this more formally is that in a given model, P-automorphisms are Q-automorphisms – though other similarity relations could also be used, such as varieties of bisimulation.

But in a more *global version* across different worlds, the requirement might be as follows:

If object x in world w is P-similar to object y in world v, then x, y are also Q-similar.

In model-theoretic terms, this has precisely the flavor of the notion of implicit definability as introduced in Section 1, with similarity cashed out as isomorphism or potential isomorphism.

One recurrent issue in philosophy has been this: When does supervenience imply *reduction* in the sense of explicit definability? This is indeed what Beth's Theorem would say, but to apply this result to supervenience, one must first introduce a formal first-order language plus a background theory into the philosophical discussion. It might be debatable whether this methodological constraint fits the spirit of the philosophical literature, or at least, the felicity of this formalization move needs argumentation. As we said, failure of explicit definition is in fact one reason why supervenience is philosophically attractive as a non-reductionist position positing merely harmony between two realms without brute force reduction.

Two technical points can be made here. First, some logics with the Beth property might still be harmless from a reductionist perspective. For instance, in second-order logic an explicit definition can be extracted routinely from an implicit one by means of an impredicative existential quantification over the irrelevant vocabulary, trivializing Beth's Theorem. Such a 'reduction' restating the implicit definition seems harmless.

But more interestingly, one might try to take a fresh look at the relevant philosophical literature using non-higher-order non-classical logics that *lack the Beth property* as a way of avoiding reductionism. One example are the relevant logics lacking the Beth property discussed in [47], cf. also Chapter 3 in this book. But of course, this weakening of the base logic should also have some independent motivation: mere avoidance of the Beth property is hardly a positive virtue. After all, the function is there, but the vocabulary in logics of this kind does not allow us to define it explicitly. One immediate test for this way-out would be if the known technical counterexamples (sometimes produced by computer-assisted search) have any independent conceptual interest.[14]

---

[13] As we shall see in Section 9, supervenience is a form of abstract *dependence*, and in fact some authors have analyzed its general logic in these terms, cf. [30].

[14] Even so, somewhat perversely, could it be that undefinability (and once on this track, unprovability), i.e., keeping some things implicit only, might sometimes be regarded as positive features of a logical system?

Finally, there is also an alternative way of thinking that calls for different distinctions. By the above definition, supervenience as implicit semantic definability already says that there is an *abstract functional dependence* of the supervening predicate Q as the value of some function on the equivalence classes for its underlying predicate P. Beth's theorem then adds that, under special circumstances, this abstract functional dependence can be expressed generically across models, by some explicit first-order definition stated in the P-language. What does this further feature really imply?

Returning to the philosophical concern of brute-force 'reducibility', even such an explicit definition need not be a burdensome form of reduction if that definition is so unwieldy as to be inaccessible to bounded human agents. In this sense, one might say that explicit reduction only hurts and constrains us if we (or others) can *grasp and manipulate* the dependence.

What such finer distinctions would call for are more computational analyses of the powers of bounded human agents, and in terms of Beth's Theorem, stricter bounds on the *complexity of the definitions* produced. In this way, some of the work in computational logic on Beth-type theorems presented in Chapter 6 and Chapter 12 of this book might become relevant to philosophical discussions.

Finally, interestingly, the earlier local version of supervenience is not directly the content of Beth's Definability Theorem, and rather falls under this well-known theorem from [46]:

> If $\boldsymbol{L}$-automorphisms are also P-automorphisms in all models for a first-order theory T, then T implies a disjunction of explicit definitions for P in terms of $\boldsymbol{L}$, not necessarily one single definition.[15]

Supervenience is just one instance where a more detailed study of a rich philosophical literature might match up with a greater variety of notions and results in existing logical theory. We now turn to other examples of such interfaces.

4 Determinism

The notion of supervenience also relates to another pervasive philosophical theme: that of *determinism*. This notion occurs in classical metaphysics, where Aristotle's famous Sea Battle argument investigated a sense in which the logical law of Excluded Middle for future truths seems to imply determinedness of our present behavior placing it outside of our control. Similar themes have kept returning, all the way to causal determinism in the empirical sciences, though the determination there usually runs in the opposite forward temporal direction, from what is true now to what will become true later.

---

[15] To see the difference, think of a theory T consisting of just the statement $\forall x\,(Px \leftrightarrow Lx) \vee \forall x\,(Px \leftrightarrow \neg Lx)$. This does not satisfy the implicit definability for Beth's Theorem. Two models for T with the same domain and denotation of the predicate L can still disagree on the denotation of P. But what is true is that in each of these models, the automorphisms that leave L in place will also leave P in place.

A lot is at stake in debates around this notion: ranging from the (non-)existence of free will to the role of causality in Artificial Intelligence or the nature of what physics can provide.

As with supervenience, many proposals have been made for making the notion of determinism more precise including some by prominent philosophers such as Bertrand Russell.

As it happens, in this area, there is a landmark attempt by a well-known logician to sort out the issues and apply Beth's and Craig's results in the philosophy of science. [40] formalizes physical theories such as Newtonian mechanics in first-order terms as having a mathematical part referring to the natural numbers $\mathbb{N}$ and real numbers $\mathbb{R}$ plus tuples of functions on reals that encode relevant physical system properties such as position and momentum of particles, and other notions – including abstract, non-observable ones postulated by the theory.

In Montague's analysis, a physical theory T describes a set of possible histories, where we think of models as having the standard natural and real numbers (here the latter also encode the passage of time), while functions defined on the reals give the current state of, say, a mechanical system as a function of the time. In this setting, different definitions can be given of what is a 'deterministic theory', reflecting various proposals in the philosophical literature which we will not reproduce here.[16]

Here is one typical 'implicit' version which occurs widely in the literature, going back to a well-known historical statement of determinism in physics formulated by Laplace:

> *If two histories allowed by T have equal states for the physical system at time t_0, and $t_0 < t$, then the two histories also have the same total system state at time t.*

Modulo quite a few syntactic details in Montague's treatment which are irrelevant to the intended illustration here, this semantic condition on models for physical systems can be formulated as a form of implicit definability in a first-order theory. Craig Interpolation and Beth's Theorem then apply to conclude that the theory also provides strong determinism in the form of an explicit definition in the language of the theory for the system state at time t in terms of the state-at-time-t_0, t_0 and t, a definition which can be used for physical reasoning and empirical calculation.

At the same time, linking to our earlier discussion of various perspectives on supervenience, Montague's formal apparatus also allows us to see more precisely what is required for the Beth property, and thus, indirectly also, how one might try to avoid it.

It has to be said that there is a lot of fine-print to this analysis, and Montague's paper may not have had the impact it deserves due to an overload of notions, results and conclusions pointing in various directions. However, this may just illustrate an earlier general point:

---

[16] Montague notes that theories can easily be deterministic in terms of arbitrary functions for the dependence of one state on others, while the more practical issue is generic definability of the functional dependence, fit for computational uses. This distinction between model-dependent definitions and generic ones for all models echoes an earlier point made about supervenience and reductionism in Section 3.

bringing logic to bear on philosophical notions and discussions need not lead to unique or definitive outcomes that can lay matters to rest forever. Also, our earlier point applies that the logical results can only be made to work if one first formalizes the general philosophical discussion of determinism in terms of formal languages and theories.

There are many further types of formalization of empirical theories by logical means, on which there exists a vast literature, cf. [45, 6]. Of the many classical themes treated in this literature, one will return in Section 8 below, viz. the different vocabularies that go into the making of an empirical theory, where the relevant connection is now with Interpolation.

For now, we move to another broad topic in philosophy in the spirit of our themes so far.

5 Questions

New information from experiments in scientific theories or through more mundane channels of communication, often arises from asking *questions*. Accordingly, questions are ubiquitous in the philosophy of language, or in general epistemology – where knowledge can be seen as knowing the answer to some relevant question which controls the direction of inquiry.

Ever since the early pioneering work documented in [1] it has been proposed by philosophers to let logic deal with, not just consequence relations among propositions or statements that give information, but also between questions that ask for information.

In a very common setting for implementing this, cf. [33], logical semantics for questions models them in terms of their complete answers as forming a partition of a relevant state space. One often thinks of the partition as a current 'issue' to be resolved by identifying one of the partition cells as the actual one.

Valid consequence from a set of questions then becomes the following notion, which has been around for a long time in the semantic literature:

> *If we have all the answers to the 'premise' questions* $?P_1, \ldots, ?P_k$*, we automatically also have an answer to the 'conclusion' question* $?C$*.*

For instance, in this sense, (i) $?A$ implies $?\neg A$, (ii) $?A, ?B$ imply $?(A\&B)$, and so on, suggesting a connection between consequence for questions and definability from premises.

A semantic definition for the notion underlying consequence between questions runs as follows on modal state spaces where formulas can be true or false:

> *If two states* s, t *agree on all* P_i*-truth values* $(1 \leq i \leq k)$*, then they also agree on the* C*-truth value.*

Stated in other terms, the joint refinement of the P_i-partitions is included in the C-partition.

Of course, there is much more to the actual behavior of questions in natural language, as they raise or modify issues and steer the course of conversation and inquiry. In addition, questions interact with other informational events, and for instance, the above notion of

question-consequence can be made conditional on events $!Q$ of receiving the indubitable information that Q by restricting the total state space only to states where Q holds.

The preceding notion is clearly in the same spirit as the invariance and similarity notions discussed earlier in connection with Beth-style implicit definability.[17] Using this analogy, a complete logic for consequence between questions suitably defined in a predicate-logical setting was given in [23], who use the slightly strengthened projective version of Beth's Theorem to prove the following key equivalence for conditional implication between questions:

▶ **Theorem 3.** $!R, ?P \models ?Q$ *iff there is a first-order formula* $\alpha(P)$ *with* $R \models Q \leftrightarrow \alpha(P)$.

This result can be seen as a sort of expressive completeness of the usual logical constants for the purpose of giving explicit answers, tying logic of questions closely to logic of propositions.

Also, interestingly, the cited paper connects Uniform Interpolation (see Chapter 9 in this book) with the existence of a 'best answer' to a question given some background information.

Thus, the philosophically important notion of questions can be brought within the scope of logic, and the Beth and Craig theorems are key to obtaining more definite results.[18]

Again, this is just a start, and many issues remain for discussion, and here is just one. The preceding account will work in a first-order setting, but it is of course a moot point whether natural language is first-order. In particular, do the above results go through for extended languages with non-first-order *generalized quantifiers* such as "most" or "many"? In this area of Generalized Quantifier Theory, as elsewhere, formal semantics of natural language meets with issues in mathematical logic concerning the properties of logical systems, [43].

6 Dependence

It is time to step up the abstraction level of our analysis. Consequence between questions is a special case of semantic *functional dependence* between variables whose values can be truth values for Yes/No-questions, or other objects in the case of more general *wh*-questions, cf. [24]. But functional dependence also lies behind many notions discussed earlier in this paper, with our analysis of determinism involving variables taking values in state space, or supervenience as a case where the values of variables are even just equivalence classes of some relevant invariance relation.

In modern dependence logics, [48], functional dependence runs between sets of variables X and variables y in *state spaces* M where not all simultaneous assignments of values to these

[17] In fact, analogies run much wider, and similar 'equal answer' equivalence relations have been used to study the widespread phenomenon of 'ceteris paribus' reasoning [13].

[18] For much more on the still growing area of logics of questions, we refer to [27].

variables may be available at states. These 'gaps' are precisely what induce correlations between the values of variables, of which functional dependence is one important species.[19]

The dependence setting also makes sense in other areas of philosophy, such as the Situation Theory to be discussed in Section 7, where information flows and knowledge arises because of various types of informational correlation between situations in the world, which need not come from their structural similarities, but be enforced by global constraints on the behavior of the overall system.

The above semantic definition of functional dependence is one of *implicit semantic determination*, where we say that $s =_X t$ if s, t give the same values to all variables in the set X. Here is the general notion of functional dependence of a variable y on a set of variables X:

For any two states s, t in state space model M, $s =_X t$ implies $s =_y t$.

This abstract framework applies to questions with truth values for the relevant propositions, but also to knowing objects as in constructions like "knowing who killed Mr. T", and even to the earlier notion of supervenience. It also applies to implicit definability in Beth's sense when values of variables can be whole equivalence classes of some relevant similarity relation.

The earlier-quoted completeness theorem for the basic logic of propositional questions can then be seen as special case of a general minimal propositional modal logic of functional dependence proposed in [2] with questions as variables and truth values as their values.

In this setting, an earlier point returns. The stated semantic equivalence suffices for the existence of an abstract function F on value domains for variables so that for each state s,

$$s(y) = F(s[X]).$$

While such a function can always be found ad-hoc, locally in a given state space model M, Beth's Theorem then gives a special definable function F, under the right conditions, which works generically across models. This suggests a more general perspective on Beth-type results as an interesting companion to dependence logics, as providing conditions under which general abstract dependencies can be made explicit and thus fit for computation and deduction.

Given these connections, it is a natural question whether current dependence logics satisfy Beth Definability or Craig Interpolation theorems with respect to variables, and not just shared predicate vocabulary. For second-order dependence logic in the style of [48], the answer is that even Uniform Interpolation (see Chapter 9 for a definition and results) holds in the variable-sharing sense, as is explained in Chapter 7 in this book. For the decidable modal dependence logic of [2], only the predicate version of interpolation is known, [11], but it seems likely that a shared-variable version holds as well.

---

[19] Note that, in this semantic setting, variables are no longer the anonymous binding devices used in standard first-order logic, but named functions from states to values, displaying an individual behavior reflecting the uses of the notion of 'variable' in empirical science or probability theory.

Dependence seems to be the common thread behind most of our philosophical topics in this paper. It clearly applies to our discussions of supervenience and determinism. But dependence logic also applies to our earlier discussion of translations between logical systems, since as we noted, the existence of an explicit syntactic translation matches an implicit semantic dependence in the model classes for the two logics being connected. The reason why this works is, of course, that the variables in dependence logics can stand for anything, and likewise, their values can be anything from numbers or objects to equivalence classes of models or even information states of agents.

Given this ubiquity, basic dependence reasoning is an interesting candidate for extending the core calculus of logic beyond the usual repertoire of 'logical constants' in standard logics.

Going further, functional dependence of y on X is one end on a spectrum, where fixing the values of the variables in X leaves only one value for y. At the opposite end lies *independence*, where the current X-values place no constraint on the values that y can take at the current state. Independence, too, might play a role as a philosophical notion, say, as a sort of 'anti-definability' or 'unlearnability'. And in between on the spectrum lie notions of *correlation* between X and y which can 'soften' the force of functional dependence. This might make sense in various areas considered in this paper.

This correlation perspective again suggests interesting mathematical issues as to whether Beth's Theorem and Craig's Theorem admit of 'softer' versions producing definitions or interpolants up to exceptions, in either qualitative or even probabilistic senses.

We will now leave the invariance/dependence thread running through the first part of this paper, and look at how definability and interpolation highlight the crucial role of *vocabulary* in shaping both everyday beliefs and scientific knowledge.

7 Information and Generalized Consequence

Let us start with a core notion studied in logic, viz. *valid consequence*. A common philosophical view holds that drawing logical consequences has to do with *information*, in particular, with elucidating information that is implicitly present in given premises or data. The handbook chapter [15] discusses this view of consequence, while also identifying other broad notions of information at work in logic.

A broader focus on information is found in Situation Theory, [4], a logic-friendly formal paradigm inspired by the philosophical view of the notion of knowledge as based on information flow proposed in [28]. In this philosophical system the world consists of partial situations (at least if one takes the theory as one of metaphysics rather than epistemology), and information flows because of the existence of *correlations* between the behavior of these situations, mediated by abstract or concrete physical channels.

Logical consequence may now be thought of accordingly as acting typically *across different situations* or, more formally, different models. Say, when looking at a radar screen on the

ground, we observe light dots changing position, and conclude to movement of planes far away, and this is correct because of a correlation between their actual flight paths and what is visible locally on the screen that we are looking at.[20]

What the relevant connections are between the two situations in such inferences can of course vary in practice, since many kinds of correlation can carry information.

This situational view suggests a new notion of entailment, not just in one model as with ordinary logical consequence, but across models. Here are some notions and results from [3] developing this theme. We say that φ *entails* ψ *along a relation* R if,

for any two models M, N, if $M \models \varphi$ and $M\,R\,N$, then $N \models \psi$.

To make this a bit more concrete, in this perspective, a model M does not just support truths about itself which can be elucidated by inference from what we already know about M, the model is also a beacon 'broadcasting' further information to other models along various relations. These relations range from very strong analogies, like isomorphism, to weaker ones, including homomorphism or modal bisimulation. The further away one gets from M qua structural similarity, the weaker the information transfer.

This setting offers a fresh look at the Interpolation Theorem. We already have a valid consequence $\varphi \models \psi$. What is the further benefit of having an interpolant α in the shared vocabulary $\boldsymbol{L}$? What this really gives us is the following form of cross-model transfer:

if $M \models \varphi$ and N is $\boldsymbol{L}$-potentially isomorphic with M, then $N \models \psi$.[21]

That is, thanks to the interpolant, φ entails ψ along $\boldsymbol{L}$-potential isomorphism.[22]

This analysis of what interpolants do inspires a new version of the Interpolation Theorem.

▶ **Theorem 4.** *The following are equivalent for formulas as described just now:*
(i) Existence of an $\boldsymbol{L}$*-interpolant, (ii) Entailment along potential* $\boldsymbol{L}$*-isomorphism.*

For first-order logic, this new version is equivalent to the usual Craig Interpolation Theorem, using the downward Löwenheim–Skolem theorem plus the fact that, on countable models, potential isomorphism and isomorphism amount to the same thing. But in general, the new formulation of Interpolation is weaker, and for instance, for the fundamental infinitary

[20] This cross-situational function of inference was already present in ancient Indian logic with its standard example of 'smoke means fire', drawing conclusions about fire on an inaccessible mountain top from observations made here on the ground, [20].

[21] There are also other interpretations of the surplus. Say, one could view interpolation as stating that a logical consequence can always be related to a *relevant* implication in the sense of shared topics.

[22] Formally speaking, this notion has a modal form $\varphi \to [p.i._{\boldsymbol{L}}]\psi$. This also suggests looking at other modal notions such as $\langle p.i._{\boldsymbol{L}}\rangle\varphi$ which defines all pointed models that are $\boldsymbol{L}$-potentially-isomorphic to the current model. The latter would define a uniform interpolant in the present sense for the formula φ.

version of first-order logic $\mathfrak{L}_{\infty\omega}$ (see Chapter 7), often touted as a typical example of failure of the standard interpolation theorem, this new version turns out to hold.[23]

These findings bring us back to a general issue raised in Section 2. When saying that a good logical system might have Interpolation as a characteristic feature, we took the standard formulation of Craig's Theorem for granted. But with our two options on the table, this becomes more delicate.

What is the natural formulation of the Interpolation property? And more generally, what are natural stable versions of known meta-properties of logical systems that we would like to transfer, discounting peculiarities of their original formulation when first discovered?

Similar generalized-interpolation versions exist for the classical model-theoretic *preservation theorems* tying important types of semantic behavior to syntactic form, whose content already involves crossing between models. For instance, instead of the usual Los–Tarski theorem characterizing preservation under submodels in terms of definability in special universal syntactic form, we also have the following version:

▶ **Theorem 5.** *φ entails ψ along $\boldsymbol{L}$-submodels iff there exists a syntactically universal $\boldsymbol{L}$-formula α which interpolates as follows: $\varphi \models \alpha \models \psi$.*

The original model-theoretic preservation theorem is retrieved by taking the special case that $\varphi = \psi$. A similar case is the following characterization theorem for modal logic:

A first-order formula φ entails a first-order formula ψ along $\boldsymbol{L}$-bisimulation iff there exists a modally $\boldsymbol{L}$-definable interpolant.

The proofs of such results often proceed by inspecting standard proofs of the preservation result, and finding that they actually also establish the more informative interpolation version.

There is more to the logical correlation view of information flow between situations, since it need not arise from structural similarity, but can also result from global dynamical system behavior. This phenomenon was already identified in our discussion of dependence logics in Section 6 above, but we will not pursue it here.

Finally, what is the philosophical import of the preceding discussion? Inspired by early work on Situation Theory, this section has addressed some fundamental issues in the philosophy of logic but also in the philosophy of information and one might say, even in epistemology.

We now turn to other aspects of the role of vocabulary in philosophy: in particular, the structuring of theories consisting of organized data and regularities that we reason with.

---

[23] The proof of this result uses a Lindström-type argument with instead of the usual reliance on the Compactness Theorem, the still very powerful model existence property encoded by the well-known *projective undefinability of the notion of well-ordering* in the infinitary logic $\mathfrak{L}_{\infty\omega}$.

8 Theoretical Terms in Scientific Theories

A key theme in the philosophy of science is the difference between (a priori) mathematical theories and theories in the empirical sciences. The latter contain *observable terms* whose values can be observed or measured directly, but also crucial are *theoretical terms* whose values cannot be observed directly, but which organize and streamline the theory. Where one draws this boundary in any given theory has been a matter of much debate, but the observational/theoretical distinction makes intuitive sense, not just in scientific settings.

Worth noting historically here is the seminal paper [17], again by E. W. Beth, which for the first time introduced the idea that empirical theories in science could be studied model-theoretically as combining syntactic formulations of laws with defining collections of structures representing actual physical systems that the theory is about.

In these logical terms, consider a syntactic theory $T(\boldsymbol{L}_O, \boldsymbol{L}_t)$ with an observational and a theoretical vocabulary. If Beth's Theorem applies, the observable facts stated in $\boldsymbol{L}_O$ will uniquely fix the denotations of the theoretical terms in $\boldsymbol{L}_t$, and the latter will be explicitly definable in T. However, this is seldom the case, except for stipulative definitions for mathematical convenience without deeper physical significance.

So, let us ask a more general question with no explicit definability assumed:

What is the *empirical content* of a theory T, i.e., what does it say
about the observable world without theoretical superstructure?

A common syntactic answer here has run as follows: that content is the theory $T|\boldsymbol{L}_O$ of all consequences in the sublanguage $\boldsymbol{L}_O$ that follow, or are derivable, from T.

However, another, and highly influential view, due to [44], is semantic: the empirical content of T (its concrete range of application where the theory 'works') consists of all $\boldsymbol{L}_O$-structures that can be *expanded* to a model for T in its full language, or alternatively, the model class $\mathsf{MOD}(T)|\boldsymbol{L}_O$.[24] At least for finitely axiomatized theories T, the empirical content is then defined by the existential second-order *Ramsey sentence* $\exists L_t \cdot T(\boldsymbol{L}_O, \boldsymbol{L}_t)$.

We always have that the above expandable structures satisfy all principles in the syntactic theory $T|\boldsymbol{L}_O$. When the model class $\mathsf{MOD}(T)|\boldsymbol{L}_O$ in fact equals $\mathsf{MOD}(T|\boldsymbol{L}_O)$, the theory T is called 'Ramsey-eliminable'. This is not always the case.

For a counterexample, take the first-order theory of the natural numbers $\mathbb{N}$ with $\boldsymbol{L}_O = \{<\}$, and add a unary theoretical predicate A for which it is stipulated that it is non-empty without a lower bound. The $\boldsymbol{L}_O$-part of this theory is just the first-order theory of $<$ on $\mathbb{N}$, but the latter standard model cannot be expanded to a model for the whole theory.

What is always true, however, is that every structure in MOD(T)$|\boldsymbol{L}_O$ has an *elementary extension* to a model for the whole theory T. However, allowing this move in an empirical

[24] This fits the earlier view of abstract logics as having sentences and models on a par as components.

theory would amount to not just introducing new theoretical predicates in an empirical situation, but also additional new 'theoretical objects'.[25]

A natural specialization of this setting are finitely axiomatized empirical theories T. In that case, the first notion of empirical content asks for the *strongest consequence* of T in its empirical subvocabulary, which would require *Uniform Interpolation* as defined in Chapter 9.[26]

On a more refined view, there are more layers of vocabulary in a scientific theory: Philosophers sometimes distinguish 'observational' (fact-describing), 'dispositional' (low-level empirical generalizations) and 'theoretical'. But a similar analysis to the one above applies.[27]

There is much more to the interface of logic and philosophy of science than the few issues sketched here. For instance, notions of invariance and definability are equally fundamental in both fields, cf. [22] on the role of symmetries in physics.

Another crucial aspect of logic that we have left out completely in our brief discussion is the crucial computational role of perspicuous syntax in allowing us to express regularities and laws in scientific theories and compute practical predictions.

9 Modularity in Knowledge and Belief

Moving from scientific theories to bodies of knowledge and opinion, we now turn to epistemology as the philosophical study of what it means for humans to know and believe. In this broader field, the crucial role of language returns, if only because what we know or believe is usually expressed in natural language. Various new issues now emerge, and Craig Interpolation applies in new ways.

For a start, our knowledge and belief are about certain facts and notions, and the notion of the *topic* of a proposition, what it is 'about', has been a long-standing philosophical theme.

Now one natural logical approach to the topic of a formula defines it as the vocabulary of the smallest sublanguage in which the formula has a logical equivalent. The following result, due to [42], says that this is well-defined.

▶ **Theorem 6.** *For any formula φ stated in a first-order language $\boldsymbol{L}$, there is a unique smallest sublanguage of $\boldsymbol{L}$ under inclusion where φ has an equivalent.*

Here Craig Interpolation is at work again. Clearly there are inclusion-minimal sublanguages of the finite vocabulary $\boldsymbol{L}$ where an equivalent can be found. But suppose that there were two different ones, say, $\boldsymbol{L}_1, \boldsymbol{L}_2$, with equivalents φ_1, φ_2. Then $\varphi_1 \models \varphi_2$, and by interpolation,

[25] These notions have also been refined. The empirical content of T might also just be the observational facts and universal laws about observables that follow from T, a universal subtheory of $T|\boldsymbol{L}_O$.

[26] Interestingly, Craig, p.c., started his seminal work trying to solve the 'strongest consequence problem' for all of first-order logic, and then proving Craig Interpolation as the next best thing.

[27] The distinction also makes sense in natural language, where observable predicates such as hair color or temperature co-exist with more theoretical dispositional predicates such as being "friendly" or "breakable", or even highly theoretical predicates such as a person's "destiny".

there exists a formula α in $\boldsymbol{L}_1 \cap \boldsymbol{L}_2$ with $\varphi_1 \models \alpha \models \varphi_2$. But then α would be an equivalent for φ in a still smaller sublanguage: yielding a contradiction.

Here is another basic language-related issue concerning the modular architecture of first-order theories. It is related to the following equivalent version of the Craig Interpolation Theorem (cf. also the well-known Robinson Joint Consistency Theorem):

Suppose that a theory $T = T_1 \cup T_2$ merging two subtheories is inconsistent, then there is a formula α in the shared language $\boldsymbol{L}_1 \cap \boldsymbol{L}_2$ with $T_1 \models \alpha$ and $T_2 \models \neg\alpha$.

Now it is a commonplace that our beliefs are compartmentalized into different topics that can be disjoint with formulations that do not interact. However, most current logics of knowledge and belief just work with one cover-all language without subdivisions. Therefore, it is of interest to *split* the total language into disjoint sublanguages, and ask for the effects of this on the structure of theories. In particular, questions arise about information flow between the various components of a theory T in the total language.

Here Craig's Interpolation Theorem has been used in [37, 42] to obtain the following result on the fine-structure of bodies of beliefs.[28]

The relevant notion of modularity is the following. We say that a theory T is *split* by a partition $\boldsymbol{L}_1, \boldsymbol{L}_2$ of its total language $\boldsymbol{L}$ if there are formulas φ in $\boldsymbol{L}_1$, ψ in $\boldsymbol{L}_2$ such that T is $\{\alpha \mid \{\varphi, \psi\} \models \alpha\}$. There is a straightforward generalization to larger splittings.

Next, it is easy to see by two successive applications of Interpolation that the following statement holds whenever T is split into $\{A, B\}$ by a partition $\boldsymbol{L}_1, \boldsymbol{L}_2$:

If $T \models \varphi$, then there is an $\boldsymbol{L}_1$-formula α and an $\boldsymbol{L}_2$-formula β such that
(i) A $\models \alpha$, (ii) B $\models \beta$, and (iii) $\alpha, \beta \models \varphi$.

A Parallel Interpolation Theorem in this spirit can also be proved for larger finite splittings by suitably iterating Craig Interpolation. Using this, one obtains the following general result, where 'largest' refers to the number of partition cells in a splitting:

▶ **Theorem 7.** *Every theory with a finite vocabulary has a unique largest splitting.*

As a third example, we just mention a further major theme in formal epistemology, viz. *information dynamics*. Theories as bodies of knowledge and belief typically *change* when update takes place with new information produced by observation, reasoning, or communication – three major processes that govern science as well as ordinary life. With the above results in place, the usual logical accounts of knowledge update and belief revision for rational epistemic agents, [32, 10], can be considerably refined to introduce more topic-sensitivity and modular structure into the representation of what agents know and believe at any stage of the process.

[28] For convenience, we talk about finitely axiomatized theories though the cited results hold more generally by a simple appeal to the Compactness Theorem for first-order logic.

This concludes our survey of uses of Interpolation in various areas of philosophy. Compared to our first theme of semantic invariance and dependence, we do not offer one crisp denominator covering all of these, though the pervasive influence of particular vocabularies being used in reasoning and theory construction is an obvious red thread.[29] Whether these two themes unify further than this is not resolved in this paper, and we leave finding a more integrated perspective as a desideratum.

10 A Historical Precursor: Bolzano

Beth's Theorem and the Interpolation Theorem highlight the importance of language as crucial ingredient in the central notion of logical consequence. But actually, this theme in logic may be less new than we have made it out to be.

A historical precursor to this linguistic focus is Bernard Bolzano whose "Wissenschaftslehre", [21], has the first systematic semantic notion of consequence, and the first explicit study of different notions of consequence for different purposes (an early form of 'logical pluralism'), with varying properties depending on the standards that one adopts for strictness of styles of reasoning. The following discussion is taken from [7].

For Bolzano, whether we call a consequence valid depends on which parts of the premises and conclusion we consider as having a *fixed meaning*, and which parts are considered *variable*, and hence amenable to substitutions of other expressions in the same category.

For instance, a step from premises {John is taller than Mary, Mary is taller than Paul} to "John is taller than Paul" is valid in this setting when we consider "John, Paul, Mary" as variable terms, but fix the meaning of "taller". But this consequence is not valid if we also consider "taller" as a variable term for which we allow substitutions of other binary predicates that need not be transitive.

The inference from "x is P-er than y, y is P-er than z" to "x is P-er than z" will even remain valid in Bolzano's sophisticated sense if we just fix the meaning of the general comparative construction "-er" in natural language which turns adjectives into comparatives.

Thus, we can view Bolzano's notion of consequence as being *ternary* rather than binary:

$$P_1, \ldots, P_k \models_{\boldsymbol{L}} C,$$

where the third argument, the vocabulary set $\boldsymbol{L}$, explicitly records which subexpressions of the formulas involved we see as variable.

This means that a logical calculus must now keep explicit track of the relevant vocabulary, leading to questions, e.g., about how to combine consequences with different vocabularies.

---

[29] Our focus here has been the *non-logical vocabulary* in formulas. It should be noted, however, that *logical vocabulary* also plays a key role in constructing interpolants, and this feature may go into the intuition that an interpolation theorem requires some sort of expressive completeness for a logical system.

For instance, let $P_1, \ldots, P_k \models_{\boldsymbol{L}_1} C$ and $Q_1, \ldots, Q_k \models_{\boldsymbol{L}_2} D$ be given valid consequences. Should we now intersect the two vocabulary sets $\boldsymbol{L}_1, \boldsymbol{L}_2$, or rather take their union when drawing the natural conjunctive conclusion $C \wedge D$ from the union of the two sets of premises? We leave an answer to the reader.

All this enriches the usual modern way of thinking about the basic 'structural rules' governing different styles of inference, where the language aspect is not an explicit parameter: this time, it has become crucial.

This explicit manipulation of vocabulary seems in the spirit of the Interpolation Theorem, and ternary sequent calculi are in fact a standard tool in proof-theoretic arguments establishing its truth. But beyond this auxiliary use, we might also come to see ternary sequents that highlight vocabulary as the primary vehicles for inference.[30]

11 Conclusion

This short and light survey has revolved around one circle of general ideas, namely, dependence, lifting, vocabulary, transfer, and definability as these occur in various areas of philosophy. What we have tried to show is that both the *notions* and the *results* found in logic show strong analogies with themes in philosophy, and can, under certain circumstances, be applied to yield broader insights.

It should be noted here that our methodology at the logic-philosophy interface can be broadened. The discussion in this chapter has mainly concentrated on the semantic side of logic and on model-theoretic analysis. It would also be of interest to revisit all topics of the preceding sections in a proof-theoretic perspective. For instance, the standard proof-theoretic analysis of Craig Interpolation delivers a recursive procedure which, given a proof of an implication $\varphi \to \psi$ outputs an interpolant and separate proofs for $\varphi \to \alpha$ and $\alpha \to \psi$. This changes the discussion from pure truth or validity to 'reasons' for these semantic facts, and this, too, can be connected up with various themes in philosophy.[31]

But even as it is, our case study also raises issues concerning the fit of logical analysis to philosophy. Logic needs a strictly defined formal *language*, both for its model theory and proof theory, and while this language dependence helps for precision and proving definite results, it may not always fit the spirit of philosophical discourse and thinking, except for those areas which have undergone the twentieth-century 'linguistic turn'. And even when

[30] Not just our theme of language awareness has roots going back to pioneering work in the 19th century. The same is true for our analysis of invariance and definability, which can be traced back as far as the seminal studies in [34] on the emergence of language as a medium for describing notions that are invariant under relevant transformations. See also [49] for further history.

[31] In arithmetical terms, switching from validity to provability and coding up formulas and proofs as natural numbers, the complexity of the usual formulation of Craig Interpolation is Π^0_2. The proof-theoretic analysis provides a stronger Π^0_1 version, so there are also interesting technical issues here.

language matters, it is not clear whether the restriction to a first-order language (the original habitat of the Beth and Craig theorems) is always meaningful in philosophy, just as this restriction does not always fit practice in mathematics and the sciences.

In fact, except for technical subfields of formal epistemology, philosophy of science, and philosophy of language, professional philosophy largely makes do with *natural language*, perhaps enriched with some notations and regimented with technical terminology. What the meta-theorems of logic say in such a setting is clearly a legitimate matter for debate. And we have also suggested occasionally that the properties expressed by these meta-theorems, usually considered a virtue or even a requirement on logic design, might not always be desirable.

Having said this, one can also view the use of logic as lying at a greater distance, offering a model for a philosophical practice (with 'model' now in its broader colloquial sense), in which case application just means providing attractive new perspectives and suggestions. Also, the formal similarities noted in this paper could be a force for seeing unifying themes and perspectives across philosophical subdisciplines.

Continuing with the presuppositions of logical analysis, the reliance of Beth's Theorem on a specified *theory* can be questioned, since many philosophical discussions (and perhaps even working practices in sciences such as physics) are about One World where everything of interest takes place.[32] On the other hand, one might also argue that theory reliance is in line with the axiomatic approach in abstract mathematics, and this is definitely the broader context in which Beth saw his own work.

Despite these caveats, we believe that our survey has offered a wide variety of natural points of contact between logic and philosophy along our main themes, and indeed, several of the authors cited exemplify the union of the two fields, being logicians who were at the same time serious philosophers. More specifically, it might be a rewarding project to put the relevant literature in the areas of philosophy mentioned in this survey paper side by side with current logical studies of definability and interpolation in a much more detailed and systematic manner. Here, our selection of topics provided just one sample of course, while even the topics selected could profit from broadening the logic interface to include themes from computational logic such as complexity, from interfaces with quantitative approaches, [14], or from the vast area of non-classical logics.

Finally, as to what is at stake in a chapter like this, the real benefit of the juxtaposition that we have pursued may not be so much logical solutions to philosophical problems as *mutual inspiration* – since, conversely, logic can also learn from the philosophical imagination and wide-ranging theorizing in the areas highlighted here.

---

[32] Say, in Padoa's method, implicit definability might just be determination in the single model of Euclidean Space that we focus on, the way we think, say, about how a triangle determines its centre of gravity, and to match this, we would need localized versions of Beth's Theorem in suitable structures.

Acknowledgments

I thank Alexandru Baltag, Balder ten Cate, Sören Brinck Knudstorp, Martin Otto, Ian Pratt-Hartmann, Pablo Rivas Robledo, Martin Stokhof, Albert Visser, Frank Wolter, and Urszula Wybraniec-Skardowska, as well as audiences for presentations of this material in Amsterdam, Beijing, Berkeley, and Dortmund for their comments on topics in this paper. I also thank Christoph Wernhard for his generous help in optimizing the look of this paper.

References

1 K. Ajdukiewicz. Interrogative sentences. In J. Giedymin, editor, *The Scientific World Perspective and Other Essays; 1931–1963*, volume 108 of *Synthese Library*, pages 155–164. Springer, 1978. Originally published as: Zdania pytajne, in Logiczne podstawy nauczania, Nasza Księgarnia, Warszawa (1934).

2 A. Baltag and J. van Benthem. A simple logic of functional dependence. *Journal of Philosophical Logic*, 50:939–1005, 2021.

3 J. Barwise and J. van Benthem. Interpolation, preservation, and pebble games. *Journal of Symbolic Logic*, 64(2):881–903, 1999.

4 J. Barwise and J. Perry. *Situations and Attitudes*. Bradford Books, The MIT Press, 1983.

5 J. Barwise and J. Seligman. *Information Flow. The Logic of Distributed Systems*. Cambridge University Press, 1997.

6 J. van Benthem. The logical structure of science. *Synthese*, 51:431–472, 1982.

7 J. van Benthem. The variety of consequence, according to Bolzano. *Studia Logica*, 44(4):389–403, 1985.

8 J. van Benthem. Information transfer across Chu spaces. *Logic Journal of the Interest Group for Propositional Logics*, 8(6):719–731, 2000.

9 J. van Benthem. The many faces of interpolation. *Synthese*, 164(3):451–460, 2008.

10 J. van Benthem. *Logical Dynamics of Information and Interaction*. Cambridge University Press, Cambridge UK, 2011.

11 J. van Benthem, B. ten Cate, and R. Koudijs. Local dependence and guarding. In *Proceedings Advances in Modal Logic*, pages 135–154, 2022.

12 J. van Benthem and K. Doets. Higher-order logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume I, pages 275–329. Reidel, Dordrecht, 1983.

13 J. van Benthem, P. Girard, and O. Roy. Everything else being equal: a modal logic for ceteris paribus preferences. *Journal of Philosophical Logic*, 38(1):83–125, 2009.

14 J. van Benthem and T. Icard. Interleaving logic and counting. *Bulletin of Symbolic Logic*, 29(4):503–587, 2023.

15 J. van Benthem and M.-C. Martinez. The logical stories of information. In P. Adriaans and J. van Benthem, editors, *Handbook of the Philosophy of Information*, pages 217–280. Elsevier, Amsterdam, 2008.

16 J. van Benthem and D. Pearce. A mathematical characterization of interpretation between theories. *Studia Logica*, 43(3):295–303, 1984.

17 E. W. Beth. Towards an up-to-date philosophy of the natural sciences. *Methodos*, 1:178–185, 1949.

18 E. W. Beth. On Padoa's method in the theory of definition. *Koninklijke Nederlandse Akademie van Wetenschappen, Proceedings*, 56:330–339, 1953. Also in Indagationes Mathematicae, 15, 330–339, 1953.

19 E. W. Beth. *The Foundations of Mathematics*. North-Holland, Amsterdam, 1959.

20 J. Bochenski. *A History of Formal Logic*. University of Notre Dame Press, Notre Dame Indiana, 1961.

21 B. Bolzano. *Wissenschaftslehre*. Seidelsche Buchhandlung, Sulzbach, 1837.

22 J. Butterfield and S. de Haro. *The Philosophy and Physics of Duality*. Oxford University Press, 2025.

23 B. ten Cate and C. Shan. Axiomatizing Groenendijk's logic of interrogation. In M. Aloni, A. Butler, and P. Dekker, editors, *Questions in Dynamic Semantics*, pages 63–82. Brill, Leiden, 2007.

24 I. Ciardelli. Dependency as question entailment. In S. Abramsky, J. Kontinen, J. Väänänen, and H. Vollmer, editors, *Dependence Logic: Theory and Applications*, pages 129–181. Birkhäuser/Springer, Cham, 2016.

25 W. Craig. Linear reasoning. a new form of the Herbrand-Gentzen theorem. *Journal of Symbolic Logic*, 22:250–268, 1957.

26 W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.

27 C. Cross and F. Roelofsen. Questions. Stanford Encyclopedia of Philosophy, 2022.

28 F. Dretske. *Knowledge and the Flow of Information*. The MIT Press, 1983.

29 H. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, New York, 1995.

30 J. Fan. A modal logic of supervenience. *Notre Dame Journal of Formal Logic*, 60(2):283–309, 2019.

31 S. Feferman. Harmonious logic: Craig's interpolation theorem and its descendants. *Synthese*, 164:341–357, 2008.

32 P. Gärdenfors. *Knowledge in Flux. On the Dynamics of Epistemic States*. The MIT Press, 1987.

33 J. Groenendijk and M. Stokhof. Questions. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 1059–1131. Elsevier, Amsterdam, 1997.

34 H. von Helmholtz. Über die Thatsachen, welche der Geometrie zu Grunde liegen. In *Wissenschaftliche Abhandlungen*, volume II, pages 618–639. Johann Ambrosius Barth, Leipzig, 1883.

35 E. Hoogland and M. Marx. Interpolation and definability in guarded fragments. *Studia Logica*, 70:373–409, 2002.

36 E. Hoogland, M. Marx, and M. Otto. Beth definability for the guarded fragment. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Logic for Programming and Automated Reasoning*, volume 1705 of *Lecture Notes in Artificial Intelligence*, pages 273–285. Springer, 1999.

37 G. Kourousias and D. Makinson. Parallel interpolation, splitting, and relevance in belief change. *Journal of Symbolic Logic*, 72(3):994–1002, 2007.

38 I. Mason. The metatheory of the classical propositional logic is not axiomatizable. *Journal of Symbolic Logic*, 50(2):451–457, 1985.

39 B. McLaughlin. Supervenience. Stanford On-Line Encyclopedia of Philosophy, 2023.

40 R. Montague. Deterministic theories. In R. Thomason, editor, *Formal Philosophy*, pages 303–359. Yale University Press, New Haven, 1974.

41 M. Otto. An interpolation theorem. *Bulletin of Symbolic Logic*, 6(4):447–462, 2000.

42 R. Parikh. Beth definability, interpolation and language splitting. *Synthese*, 179:2011–2021, 2011.

43 S. Peters and D. Westerståhl. *Quantifiers in Language and Logic*. Oxford University Press, Oxford, 2006.

44 F. Ramsey. Theories. In R. B. Braithwaite, editor, *The Foundations of Mathematics and Other Logical Essays*. Kegan Paul, Trench, Trubner and Co., London, 1931.

45 F. Suppe, editor. *The Structure of Scientific Theories*. The University of Illinois Press, Chicago, 1974.

46 L. Svenonius. A theorem on permutations in models. *Theoria*, 3:173–178, 1959.

47 A. Urquhart. Failures of Beth's theorem in relevance logics. *Review of Symbolic Logic*, 18(3):949–962, 2025.

48 J. Väänänen. *Dependence Logic. A New Approach to Independence-Friendly Logic*. Cambridge University Press, 2009.

49 H. Weyl. *Philosophy of Mathematics and Natural Science*. Atheneum, New York, 1963. Reprint of German version from the 1930s.

50 J. Zucker. The adequacy problem for classical logic. *Journal of Philosophical Logic*, 7:517–535, 1978.