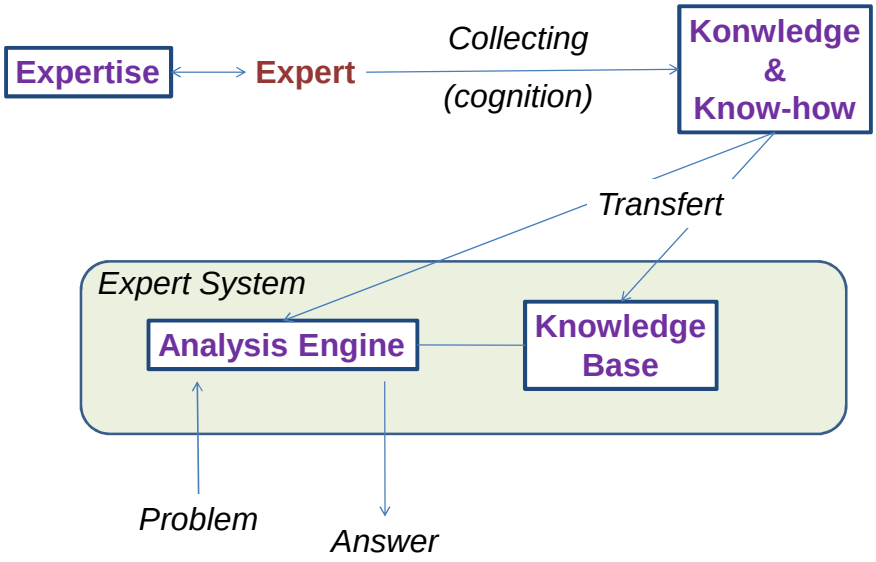
Artificial Intelligence

Logic and Expert Systems

Introduction



(C) Hervé Barbot, 2009 4

System with Knowledge Base ?

- SBC (Système à Base de Connaissance)
  - Knowledge Base is related to a specific domain, a type of problem, …
  - Explicit representation of the knowledge
- Transfert of the knowledge
  - Implies a change of :
    - Media / support
    - Form
  - Linguistic modeling
    - Power of expression / statements
      - Is it possible to transfert the complete knowledge that the system shall take into account?
      - Is it possible to transfert all the knowledge that the expert owns?
    - Enables reasoning
      - Is the form well adapted to the processing that are required for solving a specific problem?
    - Efficient!

(C) Hervé Barbot, 2009 5

Examples of knowledge base representation

- Triplets < object , attribute , value >
- Semantic networks
  - Nodes = concepts
  - Arcs = relations between concepts
- Logic – Predicates
  - Predicates represent facts, situations
  - Predicates can be defined based on conditions satisfied by other predicates
  - Equivalences are defined between logical expressions
  - A goal is defined by a predicate to satisfy
- Rules
  - Relations between:
    - Known information
    - Information to determine / verify / infer
  - Can include the execution of procedures

 6

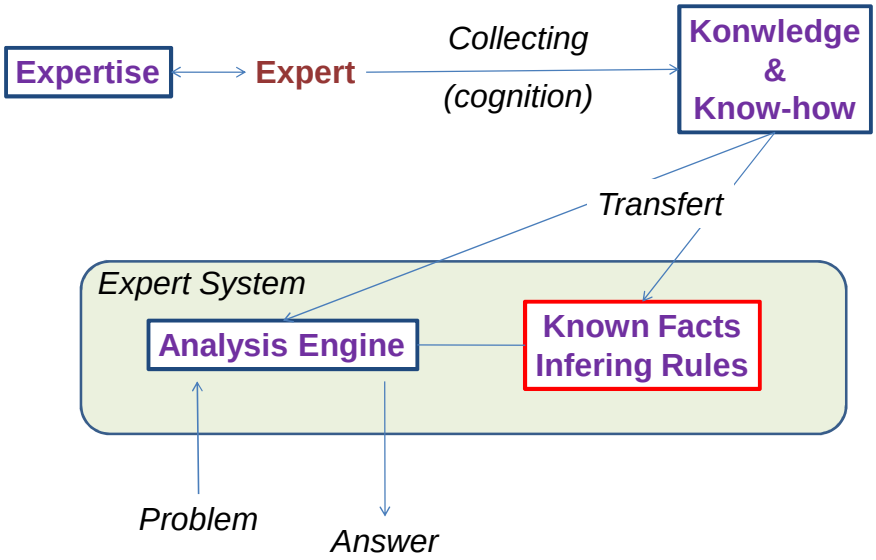
Rules-based reasoning

- Components :
  - Set of rules
  - Set of facts
  - Infering engine
- Principle
  - Find among the set of rules the ones that can be « fired », or « applied »:

    If conditions are verified, then conclusions are applicable
  - Select one of these rules

    This may need conflict resolution
  - « Fire » / execute the rule

 7



(C) Hervé Barbot, 2009 9

Logic

(C) Hervé Barbot, 2009 10

Introductory example

- Facts
  - on_top(C,A)
  - on_table(A) on_table(B)
  - free(B) free(C)

- Operations / Moving blocks:
  « remove » & « pile » actions
  - R1: free(x) ⇔ ¬ (∃ y, on_top(y,x))
  - R2: on_top(y,x) ∧ remove(y,x) ⇒ free(x) ∧ ¬ on_top(y,x)
  - R3: free(x) ∧ free(y) ∧ pile(x,y) ⇒ on_top(x,y)
- How to move 'A' on top of 'B'?
  i.e. how to obtain « on_top(A,B) »?

- Facts & Rules
  `on_top(C,A) on_table(A) on_table(B) free(B) free(C)`
  R1: `free(x) ⇔ ¬ ( ∃ y, on_top(y,x) )`
  R2: `on_top(y,x) ∧ remove(y,x) ⇒ free(x) ∧ ¬ on_top(y,x)`
  R3: `free(x) ∧ free(y) ∧ pile(x,y) ⇒ on_top(x,y)`
- « `on_top(A,B)` »?
  - R3 says: possible to obtain if we have
    « `free(A) ∧ free(B) ∧ pile(A,B)` »
  - free(B)=true ; pile(A,B) is an action to execute
  - « free(A) »=?
    R1 says: yes if `¬ ( ∃ y, on_top(y,A) )`
  - We have on_top(C,A)! … How to obtain ¬on_top(C,A)?
    Need to apply R2:
    `on_top(C,A) ∧ remove(C,A) ⇒ free(A) ∧ ¬ on_top(C,A)`
  - Thus: execute « remove(C,A) »:
    « `free(A) ∧ ¬ on_top(C,A)` » becomes true
  - Then execute « pile(A,B) »:
    the goal « `on_top(A,B)` » becomes true!

Logic

- « Language » / Formal way to represent the knowledge
  - Logics = formal languages used in order to represent/translate:
    - Information (known facts)
    - Reasoning mechanisms
      - i.e. the method used to make conclusions/ take decisions based on known facts

Syntax + semantic

- Syntax
  - Made of words and symbols
  - Words + symbols = sentences
- Semantic
  - Meaning of a particular sentence in a specific environment
    - Meaning = truth value

Types of logic

- Predicates - « ordre 0 »

  une suite de symboles séparés par des conjonctions (et), des disjonctions (ou) ou des négations (non)

  exemples:

  "Socrate est un homme" → HommeSocrate

  "Platon est un homme" → HommePlaton

  "Platon et Socrate sont des hommes"

  → HommeSocrate ∧ HommePlaton

- Propositions - « ordre 1 »

  une suite de symboles, de variables et de relations avec des quantificateurs universels et existentiels

  exemples:

  – Homme(Socrate) ∀x Homme(x) ⇒ Mortel(x)

(C) Hervé Barbot, 2009 31

Propositions

(C) Hervé Barbot, 2009 33

- Enable the representaton of:
  - Facts about the « world »: « Jean loves Marie »
  - Negations: « Marie does not love Jean »
  - Conjonctions & disjunctions
  - Sentences with logical consequenc… does not love Maris, she does not l… neither »

La logique des propositions permet
- des faits sur le monde: "Jean aime
- des négations: "Marie n'aime pas J
- des conjonctions et des disjonction
- des phrases avec "conséquence" l… l'aime pas non plus."

Une proposition est une expressior vraie soit fausse.

Éléments de base:
- symboles de propositions: P, Q, …
- phrases spéciales: Vrai, Faux

Syntaxe

- Propositions:
  « P » « Q » « R »
- Operators (symbols):
  ¬ ∧ ∨ not or and
  ⇒ IF … THEN …
  ⇔ … IF AND ONLY IF …
- Examples:
  ¬(P∧Q)∨(P⇒(Q⇒R))
  P∧Q ∧(Q⇒R)

 35

Truth table

P	¬P	Q	P ∧ Q	P ∨ Q	P ⇒ Q	P ⇔ Q
V	F	V	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	V	F	F	F	V	V

Inference

- To infer = to establish the truth value of a sentence by applying syntactical modifications to the sentence
  - The sentence is the fact to prove
  - The modifications are made based on the « infering rules »
    - Example

      Knowledge

      A=true B=true (A∧B)⇒C

      Infering 'C'

      yes because A and B are both true, and (A∧B)⇒C

Predicates

Syntax

- Au lieu des symboles propositionnels P_1 , P_2 , Q …, utiliser des prédicats et des termes:
- exemples

1) Personne (Marie) Aime(Roméo, Juliette)

constantes "prédicats" (ou simplement “prédicats”) sans signification prédéterminée

constantes individuelles (ou simplement “constantes”) sans signification prédéterminée

2) “Jean donne Fifi à Marie”
Donne(Jean, Marie, Fifi)

3) Égal(Plus(2,2), 4)

Constantes	RoiJean, 2, UniGE, ...
Prédicats	Frère, >, ...
Fonctions	Sqrt(.), JambeGaucheDe(.), ...
Variables	x, y, a, b, ...
Connecteurs	$\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
Égalité	=
Quantificateurs	$\forall \ \exists$

Formula

Phrases atomiques = *prédicat(terme$_1$, ... , terme$_n$)*
ou *terme$_1$ = terme$_2$*

Terme = *fonction(terme$_1$, ... , terme$_n$)*
ou *constante* ou *variable*

Exemples
1) Frère(RoiJean, RichardCoeurDeLion)
2) > (Longueur(JambeGaucheDe(Richard)),15)

Phrases complexes = phrases atomiques réunies par des connecteurs

$\neg S$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$

Exemples
1) Frère(RoiJean, Richard) $\Rightarrow$ Frère(Richard, RoiJean)
2) $>(1, 2) \vee \leq (1, 2)$
3) $>(1, 2) \wedge \neg > (1, 2)$

Truth table

my_predicate(A,B,C)

is true if and only if

A, B and C belong alltogether to the relationship defined by my_predicate

Quantification

- Used to define properties on sets of objects

  ∀x … whatever the value of x, … is true

  ∃x … there exist a value of x such that … is true

- ∃ <*variables*> <*sentence*>

  « There exist someone who is intelligent outside EPITECH »

  ∃ x human_being(x)∧intelligent(x)∧¬epitech(x)

∃x Chien(x) "il y a un chien"

variable liée par ∃

les variables désignent des emplacements possibles pour des constantes:

- x est une variable libre dans Chien(x)
- x est une variable liée dans ∃x Chien(x)

- ∀ *<variables> <sentence>*

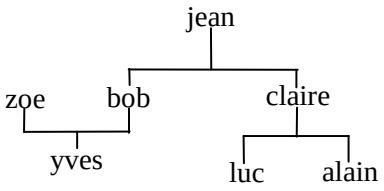
  Everyone at Epitech is intelligent

  ∀ x epitech(x) ⇒ intelligent(x)

Morgan law

…	*is the same as*	…
¬ ∃ x P		∀ x ¬ P
¬ ∀ x P		∃ x ¬ P
¬ (P ∨ Q)		¬ P ∧ ¬ Q
¬ (P ∧ Q)		¬ P ∨ ¬ Q

Example: genealogical tree



- Predicates:

  Father(x,y)

  father(yves,bob), father(claire,jean), …

  Mother(x,y)

  Allows a complete representation of the known facts

- Other relationships are derived from « father » and « mother » predicates by means of infering rules:

  $$father(x,y) \vee mother(x,y) \Rightarrow parent(x,y)$$

  Or

  $$father(x,y) \Rightarrow parent(x,y)$$
  $$mother(x,y) \Rightarrow parent(x,y)$$

(C) Hervé Barbot, 2009 50

Summary

(C) Hervé Barbot, 2009 53

Logic

- « Language » / Formal way to represent the knowledge
  - Logics = formal languages used in order to represent/translate:
    - Information: known facts
    - Reasoning mechanisms: know-how
      - i.e. the method used to make conclusions/ take decisions based on known facts

(C) Hervé Barbot, 2009 54

Types of logic

- Predicates - « ordre 0 »
  - Sentences combined from symbols and operators 'and', 'or', 'not'
    - ManSocrate — *Socrate is a man*
    - ManSocrate $\wedge$ ManPlaton — *Socrate and Planton are men*

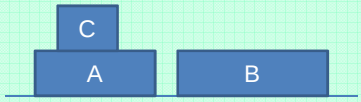
    with logical operators $\Rightarrow$, $\Leftrightarrow$
    - ManSocrate $\Rightarrow$ MortalSocrate

      If Socrate is a man, then Socrate is mortal
- Propositions - « ordre 1 »
  - Symboles, variables, quantifiers ($\exists$, $\forall$)
    - Man(Socrate)
    - $\forall$x Man(x) $\Rightarrow$Mortal(x) — *All men are mortal*

(C) Hervé Barbot, 2009 55

Example with predicates

- Facts
  - on_top(C,A)
  - on_table(A) on_table(B)
  - free(B) free(C)



- Operations / Moving blocks:
  « remove » & « pile » actions
  - R1: free(x) ⇔ ¬ (∃ y, on_top(y,x))
  - R2: on_top(y,x) ∧ remove(y,x) ⇒ free(x) ∧ ¬ on_top(y,x)
  - R3: free(x) ∧ free(y) ∧ pile(x,y) ⇒ on_top(x,y)
- How to move 'A' on top of 'B'?
  i.e. how to obtain « on_top(A,B) »?

(C) Hervé Barbot, 2009 56

- Facts & Rules
  `on_top(C,A) on_table(A) on_table(B) free(B) free(C)`
  R1: `free(x) ⇔ ¬ ( ∃ y, on_top(y,x) )`
  R2: `on_top(y,x) ∧ remove(y,x) ⇒ free(x) ∧ ¬ on_top(y,x)`
  R3: `free(x) ∧ free(y) ∧ pile(x,y) ⇒ on_top(x,y)`
- « `on_top(A,B)` »?
  - R3 says: possible to obtain if we have
    « `free(A) ∧ free(B) ∧ pile(A,B)` »
  - free(B)=true ; pile(A,B) is an action to execute
  - « free(A) »=?
    R1 says: yes if `¬ ( ∃ y, on_top(y,A) )`
  - We have on_top(C,A)! … How to obtain ¬on_top(C,A)?
    Need to apply R2:
    `on_top(C,A) ∧ remove(C,A) ⇒ free(A) ∧ ¬ on_top(C,A)`
  - Thus: execute « remove(C,A) »:
    « `free(A) ∧ ¬ on_top(C,A)` » becomes true
  - Then execute « pile(A,B) »:
    the goal « `on_top(A,B)` » becomes true!

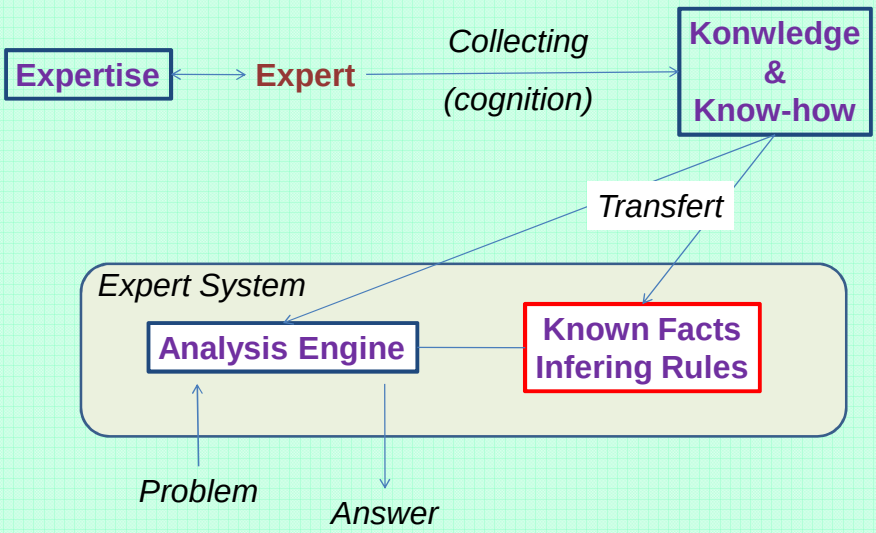
(C) Hervé Barbot, 2009 57

Rules-based reasoning

- Components :
  - Set of rules — *if …conditions… then …conclusion/action…*
  - Set of facts — *what is kown*
  - Infering engine
- Principle
  - Find among the set of rules the ones that can be « fired », or « applied »:
    If conditions are verified, then conclusions are applicable
  - Select one of these rules
    This may need conflict resolution
  - « Fire » / execute the rule

(C) Hervé Barbot, 2009 58

Rules-based expert system



(C) Hervé Barbot, 2009 59

Hypothesis : rules such as

IF A and B and … THEN X

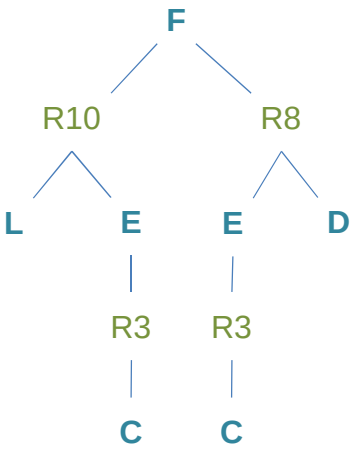
AND/OR Tree or Graph

Example

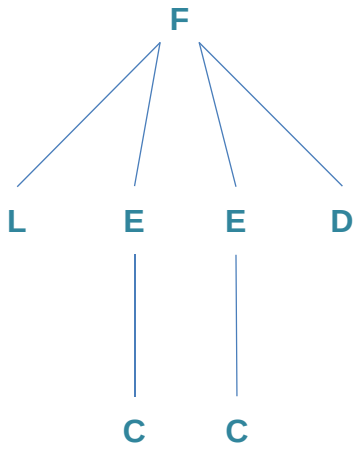
R1 : A -> B	R6 : M+L -> A
R2 : A -> C	R7 : I+B -> D
R3 : C -> E	R8 : E+D -> F
R4 : M -> C	R9 : K+F -> H
R5 : I+K -> A	R10 : L+E -> F

Graphical variations

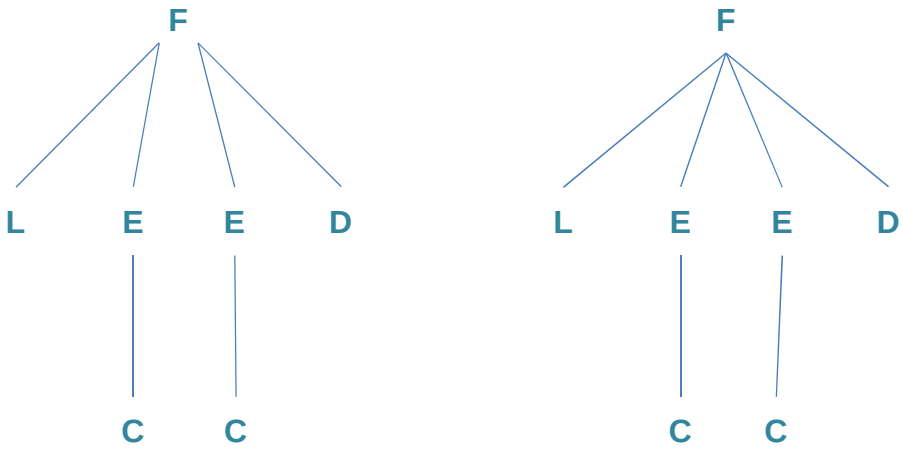
R3: C ⇒ E
R8: E∧D ⇒ F
R10: L ∧ E ⇒ F



Rules are explicitly shown

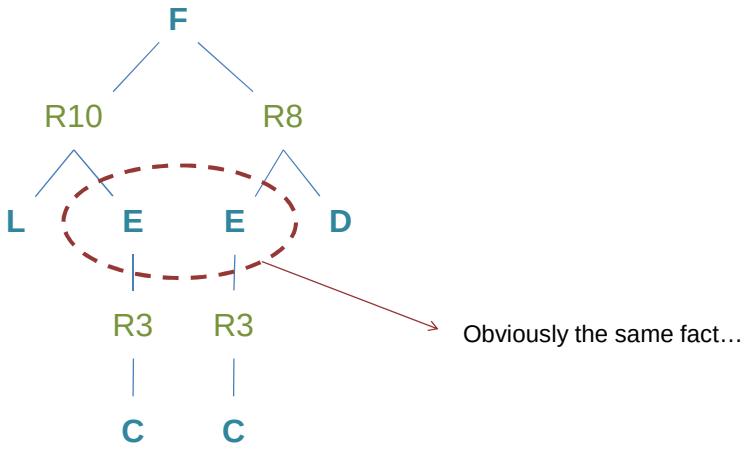


Rules are not shown

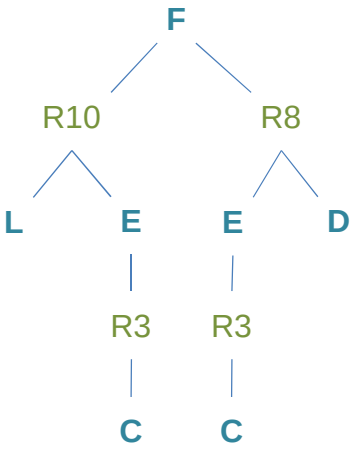


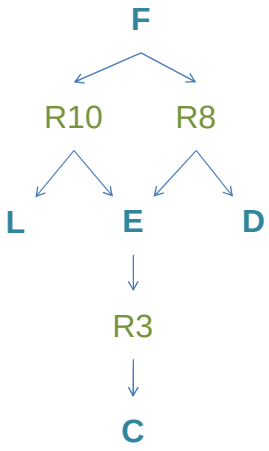
Is it the same?

67

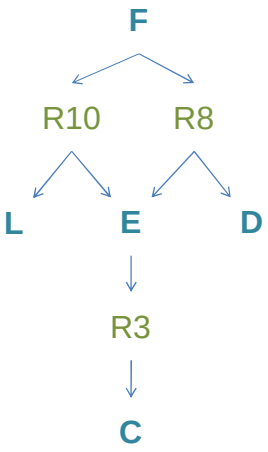


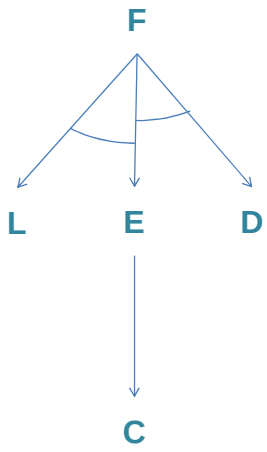
(C) Hervé Barbot, 2009 68





(C) Hervé Barbot, 2009 69





(C) Hervé Barbot, 2009 70

Chaînage avant

Forward chaining

Infering rules:
R1: A→B
R2: A→C
R3: C→E
R4: M→C
R5: I&K→A
R6: M&L→A
R7: I&B→D
R8: E&D→F
R9: K&F→H
R10: L&E→F

Known facts:
I
L
M

Problem statement :

According to:

Facts that we know are true
representing the initial state of the search,
Representing the actual description of the world

Infering rules (they are true as well!...)
representing the know-how of the problem

What other facts can we determine as true?

Lorsqu'un nouveau fait *p* est ajouté à la base de connaissances:

pour chaque règle telle que *p* s'unifie avec un prémisse
et si les autres prémisses sont connus
alors ajouter la conclusion à la base de connaissances et
continuer le chaînage

- le chaînage avant est piloté par les données (*"data-driven"*)

càd, on infère des propriétés et des catégories à partir des séquences perceptives

- Iterate on:
  - Select a rule that is applicable
    - That is for which the « conditions » part is true
  - Apply the rule
    - That is: add the conclusion in the « known facts » set
- Stop when there is no more applicable rule

- Applying a rule adds known facts
  - Thus, a rule not applicable at one time may be aplicable later…

Infering rules:
R1: A→B
R2: A→C
R3: C→E
R4: M→C
R5: I&K→A
R6: M&L→A
R7: I&B→D
R8: E&D→F
R9: K&F→H
R10: L&E→F

Initial facts = {I,L,M}

Rule applied	Knowledge	
	ILM	
R4	ILMC	
R6	ILMCA	
R1	ILMCAB	
R2	ILMCAB	*no new fact added!*
R3	ILMCABE	
R7	ILMCABED	
R8	ILMCABEDF	
R10	ILMCABEDF	*no new fact added!*

Order may differ depending on the strategy to select a rule at each step!

But the result is the same…


```
facts_set forward_chaining(
              rules_set rules
              facts_set initial_facts)

rules_set to_fire ← rules
facts_set facts ← initial_facts

WHILE there_is_applicable_rule(to_fire,facts) DO

   rule r ← select_rule(to_fire,facts)
   to_fire ← to_fire – r
   facts ← facts + conclusion(r)

ENDWHILE

RETURN facts
```


- Variants
  - Each time a rule is selected, add its conclusions in the knowledge base and fire (new) rules accordingly

    See previous exemple and pseudo-code
  - Fire rules only based on the initial facts
    - Think about rules with actions as conclusions: allow to determine what could be done, but without taking the (final) decision)

Chaînage arrière

Backward chaining

Infering rules:
R1: A→B
R2: A→C
R3: C→E
R4: M→C
R5: I&K→A
R6: M&L→A
R7: I&B→D
R8: E&D→F
R9: K&F→H
R10: L&E→F

Known facts:
I
L
M

Problem statement :

According to:

Facts that we know are true
representing the initial state of the search,
Representing the actual description of the world

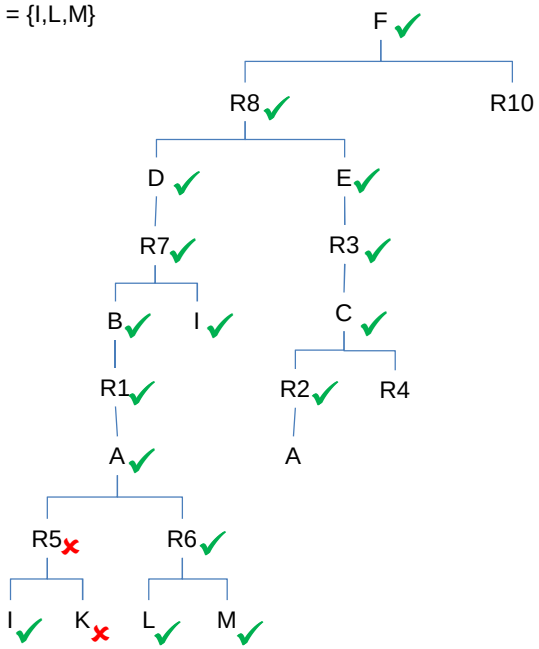
Infering rules (they are true as well!...)
representing the know-how of the problem

Is a given fact (e.g. F) true or not ?

- Iterate on all rules concluding to the fact:
  - Check if conditions are true
  - If yes, decide that the fact is true
- Checking if a condition is true is recursion!

Infering rules:
R1: A→B
R2: A→C
R3: C→E
R4: M→C
R5: I&K→A
R6: M&L→A
R7: I&B→D
R8: E&D→F
R9: K&F→H
R10: L&E→F

Initial facts = {I,L,M}
F ?



If all rules are of the form :

IF a set of facts that shall be true
THEN a set of facts that will be considered as true

- i.e. only the « and » operator is used

$$truth\_value(F) = OR_{R,\ F \in C(R)}\ fire\_ability(R)$$

$$fire\_ability(R) = AND_{F,\ F \in P(R)}\ truth\_value(F)$$

C(R)=set of facts as conclusion of a rule R
P(R)=set of facts in the condition part of rule R

```
bool truth_value: fact      F      IN
                  rules_set rules  IN/OUT
                  facts_set facts  IN/OUT

IF F ∈ facts THEN RETURN true

WHILE there_is_a_concluding_rule(rules,F) DO

   rule r ← select_a_concluding_rule(rules,F)
   rules ← rules - r
   IF fire_ability(r,rules,facts)
          THEN facts ← facts + F
               RETURN true

ENDWHILE

RETURN false
```

 99

```
bool fire_ability: rule      R          IN
                   rules_set rules      IN/OUT
                   facts_set facts      IN/OUT

facts_set facts_to_check ← condition(R)

WHILE facts_to_check ≠ ∅ DO

   fact F ← select_fact(facts_to_check)
   IF truth_value(F,rules,facts) = false
          THEN RETURN false
   facts_to_check ← facts_to_check - F

ENDWHILE

RETURN true
```

 100

After the « basic » rules and chaining algorithms…

Let's see more issues!

« OR » in conditions

- IF A OR B THEN C
- IF A THEN C
  IF B THEN C

« NOT »

- Domain of value :
  { true , undetermined }
  or
  { true , false , undetermined }

General boolean expressions

- IF <bool_expression> THEN A

```
bool fire_ability: rule R                IN
                       rules_set rules   IN/OUT
                       facts_set facts   IN/OUT

      Evaluate according to the semantic tree of the
      boolean expression

RETURN true
```

Multiple conclusions

- IF A THEN X AND Y AND Z
- IF A THEN X OR Y
  - At least one of 'X' and 'Y' shall be set to 'true'
  - How to do if we don't know that X or Y is false?
- IF A THEN X XOR Y
  - Exactly one shall be true
  - How to do if both are 'undetermined'?

Interactive backward chaining

- Idea:
  ask the user to help the system
- Principle 1:
  the system is an expert, and shall investigate as much as possible before requiring any help
- Principle 2:
  ask as few help as possible