

Crossroads

A Crossword Compiler and Solver

Andrea Reale

DEIS
Università di Bologna

Linguaggi e Modelli Computazionali LS
7 Aprile 2009

Outline

- 1 Introduzione
- 2 La Grammatica
- 3 Implementazione
- 4 Configurazione
- 5 Conclusione e possibili espansioni

Crossroads

Idee di progetto

- Ambiente per la descrizione e la risoluzione di cruciverba
- Permette di
 - ▶ Descrivere istanze di cruciverba
 - ▶ Trasformare la descrizione fornita in diverse rappresentazioni
- Fornisce strumenti di base per la ricerca di soluzioni

Obiettivi

e requisiti

Requisiti

- Semplicità d'uso
 - ▶ Sintassi intuitiva e “vicina” al linguaggio naturale
 - ▶ Complessità del linguaggio ridotta al minimo
- Modularità ed espandibilità
 - ▶ Esportazione del cruciverba in formati multipli
 - ▶ Possibilità di definire componenti intercambiabili

Cos'è un Cruciverba

e come descriverlo

Definition (da De Mauro - Paravia)

Gioco enigmistico consistente nell'indovinare, con l'aiuto di definizioni, le parole da collocare orizzontalmente e verticalmente in uno schema predisposto, in modo che ogni lettera occupi una casella e sia parte della parola con cui si incrocia

- Definizione poco puntuale
- Bisogna trovare un modello preciso che lo rappresenti
- Saranno trattati solo cruciverba con schema *rettangolare*

Il Modello

Dimensioni e schema

Definition

Dimensioni

- Coppia $(m, n) \in \mathbb{N} \times \mathbb{N}$
- Rispettivamente il numero di *righe* e di *colonne* del cruciverba

Schema

- Individuato dalla funzione:

$$b : \bar{m} \times \bar{n} \rightarrow \{0, 1\}$$

- Associa ad ogni casella del cruciverba 0 se la casella è bianca, 1 se è nera

Il Modello

Parole

Definition

Entry

- Insieme $S \subset \bar{m} \times \bar{n}$
- Costituita dalle sequenze di celle bianche - orizzontali o verticali - non interrotte da nere
- Ogni entry individua, in pratica, le celle in grado di contenere una singola parola

Dizionario

- Insieme D di stringhe appartenenti ad un linguaggio $L \subseteq \Sigma^*$

Il Modello

Parole

Definition

- Parola**
- Dati un'entry S_i ed un dizionario D_i una parola è una funzione:

$$p : S_i \rightarrow \Sigma$$

- Associa ad ogni cella un simbolo così che la loro concatenazione sia una stringa di D_i

- Soluzione**
- Associa ad ogni possibile entry S_i di uno schema una parola p_{S_i}

Il Modello

Soluzione

Soluzione Valida

- Sia X l'insieme delle coppie di entry aventi intersezione non vuota
 - ▶ Si noti che tale intersezione conterrà al più un elemento
- Una soluzione valida assegna ad ogni coppia di entry $(S_i, S_j) \in X$ parole p_{S_i} e p_{S_j} tali che:

$$p_{S_i}(r, c) = p_{S_j}(r, c)$$

laddove $S_i \cap S_j = \{(r, c)\}$

Cruciverba “abituali”

Cosa c'è di diverso

- Un solo dizionario usato per tutte le entry
 - ▶ Quello della lingua in cui il cruciverba è specificato
 - ▶ In più sigle o stringhe “speciali”
- Presenza di **definizioni**
 - ▶ Suggerimenti circa le associazioni *entry-parola*
 - ▶ Non indispensabili per trovare una soluzione
 - ▶ Usate per restringere il campo tra le soluzioni possibili

Separazione di responsabilità

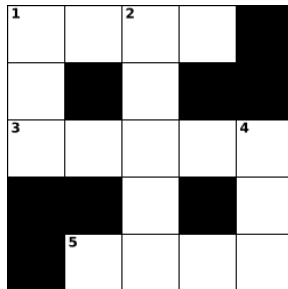
- Diversi gradi di separazione possibili
- **Descrizione vs Risoluzione**
 - ▶ Un linguaggio per descrivere un cruciverba
 - ▶ Un linguaggio per descrivere come risolverlo (Prolog)
- **Schema vs Definizioni**
 - ▶ Un linguaggio per descrivere lo schema
 - ▶ Un linguaggio per fornire le definizioni associate alle entry

L'effetto finale

Schema

Example (cross1.schema)

```
crossword exampleCwd:
  size is 5 x 5
  holes schema:
    4
    1 3 4
    none
    0 1 3
    0
end
```



L'Effetto Finale

Definizioni

Example (cross1.defs)

```
crossword exampleCwd:
  definitions language is it:
    across:
      def "National Aeronautics and Space Administration"
      def "Skateboarding trick"
      def "Il Nicolas protagonista di Con Air"
    down:
      def "Thomas Anderson in Matrix"
      def "Lo e` il tabasco"
      def "Netbook ASUS"
end
```

Across:

1. National Aeronautics and Space Administration
3. Skateboarding trick
5. Il Nicolas protagonista di Con Air

Down:

1. Thomas Anderson in Matrix
2. Lo e` il tabasco
4. Netbook ASUS

La Grammatica dello Schema

- Header
 - ▶ Intestazione e identificativo
- Dimensioni
- Specifica della funzione di schema
 - ▶ Una linea per ogni riga del cruciverba
 - ▶ Un intero per ogni **hole** sulla riga → ne indica la colonna
 - ▶ *none* se la riga è completamente bianca
 - ▶ Le ultime righe possono essere omesse se non hanno holes
- Informazioni necessarie (e sufficienti) per determinare lo schema

Estratti dalla grammatica

File dello schema

Example (Estratti)

```

<SCHEMA_FILE> ::= <HEAD><INDENT><SCHEMA_DEF><DEDENT><END>
<HEAD> ::= "crossword" <ID> ":" <NL>
<SCHEMA_DEF> ::= <DIM_DEF><HOLES_SCHEMA>
<DIM_DEF> ::= "size" "is" <NATURAL> "x" <NATURAL> <NL>
<HOLES_SCHEMA> ::= "holes" "schema" ":" <NL> <INDENT>
                  <HOLES_LINES><DEDENT>
<HOLES_LINES> ::= <LINE><NL> | <LINE><NL><HOLES_LINES>
<LINE> ::= "none" | <HOLES_SEQ>
...

```

La Grammatica delle Definizioni

- Header
 - ▶ Intestazione e identificativo
- Specifica delle definizioni
 - ▶ Possibilità di definire definizioni localizzate
 - ▶ Una serie di definizioni per ogni lingua
 - ▶ Quanto è utile in pratica?
- Definizioni orizzontali e verticali separate

Estratti dalla grammatica

File delle definizioni

Example (Estratti)

```

<DEFS_FILE>          ::= <HEAD><INDENT><DEFS_SKEL><DEDENT><END><NL>
<HEAD>               ::= "crossword" <ID> ":" <NL>
<DEFS_SKEL>          ::= <UNLOCALIZED_BODY> | <LOCALIZED_BODIES>
...
<UNLOCALIZED_BODY> ::= "defintions" ":" <NL>
                    <INDENT><DEF_BODY><DEDENT>
<LOCALIZED_BODY>   ::= "definitions" "language" "is"
                    <LANG_CODE> ":" <NL>
                    <INDENT><DEF_BODY><DEDENT>
<DEF_BODY>         ::= "across" ":" <NL><INDENT><DEFS><DEDENT>
                    "down" ":" <NL><INDENT><DEFS><DEDENT>
...

```

Considerazioni sulla grammatica

Classificazioni

- Grammatica di **tipo 2**
- Linguaggio generato di **tipo 3**
 - ▶ Non c'è *self-embedding*
 - ▶ Tutta la grammatica esprimibile con una *regexp*
- Grammatica **LL(1)**
 - ▶ Adatta a parsing *top-down*
- Nessuna *ϵ -rule*

Overview

Putting pieces together

- Interfaccia CrosswordParser
 - ▶ Parsing di schema/definizioni
 - ▶ Il parsing deve produrre una istanza della classe **Crossword**
- Classe Crossword in Java
 - ▶ Descrive una istanza di cruciverba
 - ▶ Utilizzabile per manipolarla ed esplorarla
- Interfaccia CrosswordExporter
 - ▶ `public T toExternalRepresentation(Crossword input)`
- Interfaccia SolvingEngine
 - ▶ `public void solve(Crossword toSolve)`

Scanner e Parser

L'implementazione

- Parser Prolog
 - ▶ Uso di tuProlog con DCGLibrary
 - ▶ Copia-Incolla della grammatica in Prolog :)
 - ▶ Genera l'AST come termine Prolog
- Scanner realizzato “a mano” in ambiente Java
 - ▶ Genera gli *atomi* per il parser
 - ▶ Gestisce l'indentazione tramite uno stack interno
- L'AST è esplorato con un opportuno predicato Prolog
 - ▶ I parametri del cruciverba sono passati all'ambiente Java
 - ▶ L'*unificazione* torna comoda

Controlli per la correttezza

- Controlli sintattici di base effettuati a livello di Scanner
 - ▶ Presenza di simboli non permessi
 - ▶ Correttezza dell'identazione
- Correttezza sintattica dei singoli token lato Prolog
- Errori di semantica sono rilevati alla costruzione dell'oggetto Crossword

Gli Exporter

Prolog, SVG, XHTML

- **Prolog Exporter**

- ▶ Genera la teoria in grado di risolvere il cruciverba
- ▶ Utilizza Google per costruire i dizionari per ciascuna entry

- **SVG Exporter**

- ▶ Genera l'immagine vettoriale che rappresenta il cruciverba
- ▶ Se l'istanza è risolta include le soluzioni
- ▶ Con o senza definizioni

- **XHTML Exporter**

- ▶ Genera una pagina Xhtml per il cruciverba
- ▶ Fa uso del SVG Exporter

Crossword Solver

sometimes it works

- Dizionari di dimensioni ridotte per ogni entry
 - ▶ Cerca la definizione tramite **Google** AJAX Search API
 - ▶ Estrae dai risultati i termini più frequenti
- Genera la teoria risolutiva tramite PrologExporter
 - ▶ Utilizza **tuProlog** per cercare una soluzione

Configurazione

File di configurazione

- Insieme di coppie chiave valore
 - ▶ Scelta delle classi che implementano le interfacce
 - ▶ Numerosi parametri per il solver
- **JSON** (Javascript Object Notation)
 - ▶ Linguaggio estremamente semplice
 - ▶ Parser già disponibili
- Configurazione di **WebHarvest** tramite file XML
 - ▶ Specifica come processare le pagine Web
 - ▶ Possibilità di utilizzare espressioni regolari
 - ▶ Selezione dei nodi con XPath

Conclusioni

Limiti e possibili soluzioni

- Non riesce a risolvere definizioni *tricky*
 - ▶ Implementazione di regole ad-hoc
- Performance del parser piuttosto povere
 - ▶ Reimplementazione del parser con strumenti più performanti (e.g. JavaCC)
- I tempi di costruzione dei dizionari sono non trascurabili
 - ▶ Tempi di accesso alla rete. Ci si può far poco

Conclusioni

Limiti e possibili soluzioni

- Il tempo di risoluzione è esponenziale nel numero di entry
 - ▶ Ridurre le dimensioni dei dizionari
 - ▶ Migliorarne la qualità (Thesauri, regole ad-hoc)
 - ▶ Uso più furbo dei vincoli per la risoluzione
- Migliorare la gestione degli errori
 - ▶ Difficoltà di dare descrizioni esaustive degli errori lato Prolog
- Migliorare l'interazione con l'utente
 - ▶ Risoluzione step-by-step
 - ▶ GUI interattiva
 - ▶ Possibilità di accettare suggerimenti

