SLR1 Parser

Implementazione di un Parser SLR1 per grammatiche EBNF

Cristina Lombardo: lombardo.cristina84@gmail.com

SUNTO

Programma java in grado di estrarre da una grammatica SLR1 in formato EBNF le tabelle Action Goto e in seguito definire un Parser in grado di verificare l'appartenenza di una stringa al linguaggio.

Università degli studi di Catania
A.a. 2014/15

Materia: Linguaggi e traduttori

Docente: Prof. Vincenza Carchiolo

Sommario

1 Introduzione 3

2 Requisiti 3

3 Scelte progettuali 3

3.1 Grammatica di input 3

3.2 Salvataggio ActionGoto 4

4 Implemetazione 4

4.1 Lettura file di input 4

4.2 Conversione EBNF -> BNF 4

4.2.1 Trasformazione {} 5

4.2.2 Trasformazione [] 6

4.2.3 Trasformazione () e | 6

4.3 First and Follow 6

4.4 Creazione del CFSM 6

4.5 Creazione ActionGoto 7

4.6 Parsing 7

Appendice A Esecuzione del programma 7

Appendice B Compilazione con Maven 8

1 Introduzione

Il presente documento ha come obiettivo quello di descrivere brevemente le funzionalità e l'implementazione di un programma Java in grado di interpretare le grammatiche EBNF e da esse definire un parser in grado di verificare se una data stringa di input appartiene o meno alla grammatica.

2 Requisiti

Il testo del progetto da implementare è il seguente:

Data una grammatica non contestuale verificare che:

1. *La grammatica non contestuale sia di tipo SLR(1)*
2. *Nel caso che la grammatica non sia SLR(1) salvare su un file ERRORE le condizioni di conflitto.*
3. *Nel caso la grammatica soddisfi le condizioni generare le tabelle GOTO ed ACTION e salvarle su un file OUTPUT.TXT. Il formato del file è libero.*
4. *Frase appartiene al linguaggio di ingresso.*

File di input: Grammatica non contestuale Il file di input contiene la grammatica non contestuale nella forma BNF Estesa (http://www.cs.cmu.edu/~pattis/misc/ebnf.pdf) . Ogni produzione è etichettata con un etichetta numerica seguita dal simbolo ":".

---

3 Scelte progettuali

Per rispettare i requisiti sono state attuate alcune scelte progettuali.

3.1 Grammatica di input

Il file di input contente la grammatica nel formato EBNF deve rispettare il formato seguente:

```
# Created by Cristina Lombardo
# Rules
#       "#" Comments
#       \ :Escape char
#       <E>, <F>, <T>, <X>: Non Terminal
#       i, +, -, ^: Terminals
#       \\: Escaped / used as terminal
#       \(: Special Char ( escaped used as terminal
#       () [] {} |: Special chars
```

```
##########

# Grammar

1: <E> ::= <X> {(+|-) <X>}
2: <X> ::= a|b|v

#Example of input string a+b-a-b
```

Le regole da rispettare nel creare un file di input per la grammatica EBNF sono le seguenti:

1. Gli elementi non terminali sono racchiusi da paretesi angolari
2. Un elemento non terminale deve rispettare il seguente pattern *[a-zA-Z]+[0-9]**
3. Ogni riga deve iniziare con *<numero>:* .
4. Le righe con # sono commenti.
5. \ è utilizzato come carattere di escape.

3.2 Salvataggio ActionGoto

Il salvataggio della tabella ActionGoto avviene come da requisiti all'interno del file OUTPUT.TXT nel formato standard CSV con valori separati da virgole.

4 Implementazione

Il programma effettua i seguenti passi:

1. Lettura file di input contenente la grammatica EBNF
2. Conversione della grammatica EBNF in BNF
3. Calcolo dei first e follow per ogni non terminale della grammatica
4. Produzione della macchina a stati finiti caratteristica in accordo alle regole SLR1
5. Creazione della tabella ActionGoto a partire dalla CFSM
6. Richiesta e verifica della stringa di input

4.1 Lettura file di input

Il programma legge un file di input nel formato specificato nel paragrafo 3.1.

Il file di input viene filtrato eliminando le righe che rappresentano un commento e le righe bianche.

Le righe contente la grammatica vengono trasformate in una lista di stringhe che verranno elaborate nei passi successivi.

4.2 Conversione EBNF -> BNF

In questo passo le stringhe estratte dal file di input vengono lette per essere trasformate in oggetti appartenenti alla struttura di classi definite in accordo alle regole della grammatica BNF Context Free o di tipo 2.

In fig. 1 viene mostrato il class diagram definito dal programma per le grammatiche context free. Si osservi che la classe produzione presenta l'attributo left che si compone da un unico non terminale in accordo alle grammatiche di tipo 2. La parte destra (right) invece è una lista di Element che rappresenta un'astrazione per Terminali e Non Terminali.

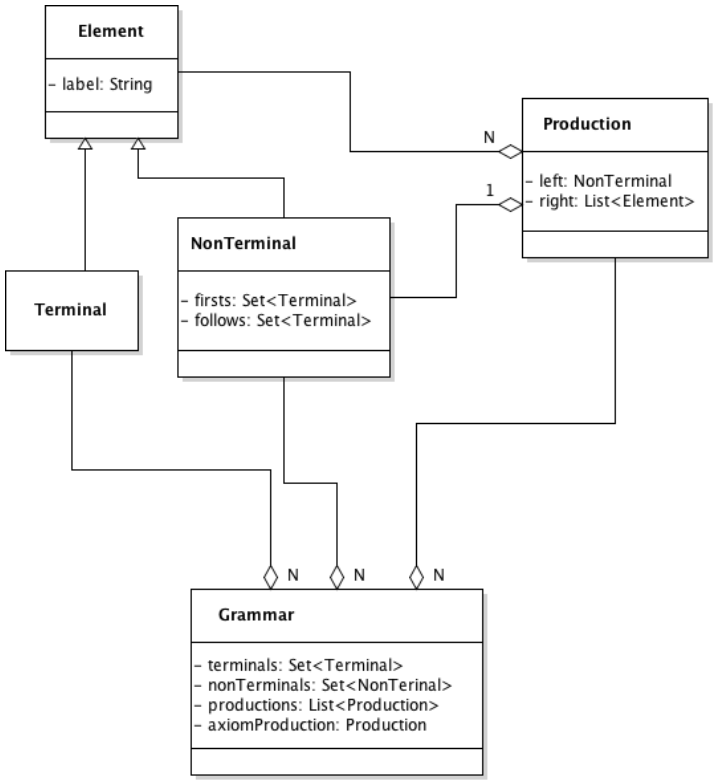

fig. 1: Context Free Grammar Class Diagram

La traduzione da grammatica EBNF a BNF viene effettuata seguendo alcune regole.

4.2.1 Trasformazione {}

La parentesi graffa implica la ripetizione di 0 o + volte della grammatica presente al suo interno. Essa ha un'analogia con la stella di kleen.

Un esempio di trasformazione è il seguente:

File di input

```
1: <E> ::= a{b}
```

Traduzione BNF

```
E -> a £1
£1 -> b £1
£1 -> ε
```

Dove E sarà la rappresentazione grafica di un oggetto di classe NonTerminal, a la rappresentazione grafica di un oggetto di classe Terminal. £1 è un non terminale creato a fronte della trasformazione, si è scelto di utilizzare £ come simbolo poiché non è possibile individuare nel file di input un non terminale che presenti tale carattere.

4.2.2 Trasformazione []

La parentesi quadra implica il ripetersi degli elementi dentro parentesi 1 o 0 volte.

File di input

```
1: <E> ::= a[b]
```

Traduzione BNF

```
E -> a £1
£1 -> b
£1 -> ε
```

4.2.3 Trasformazione () e |

File di input

```
1: <E> ::= a(b|c)
```

Traduzione BNF

```
E -> a £1
£1 -> b
£1 -> c
```

4.3 First and Follow

Definita la grammatica in accordo alla struttura mostrata in fig. 1 il programma estrarrà l'insieme dei first e dei follow per ogni non terminale appartenente alla grammatica.

Il risultato verrà salvato all'interno di un opportuno HashSet all'interno dell'oggetto NonTerminal.

Le funzioni per il calcolo dei first e dei follow sono funzioni ricorsive.

4.4 Creazione del CFSM

La macchina a stati caratteristica viene creata in accordo al digramma delle classi mostrato in fig. 2.

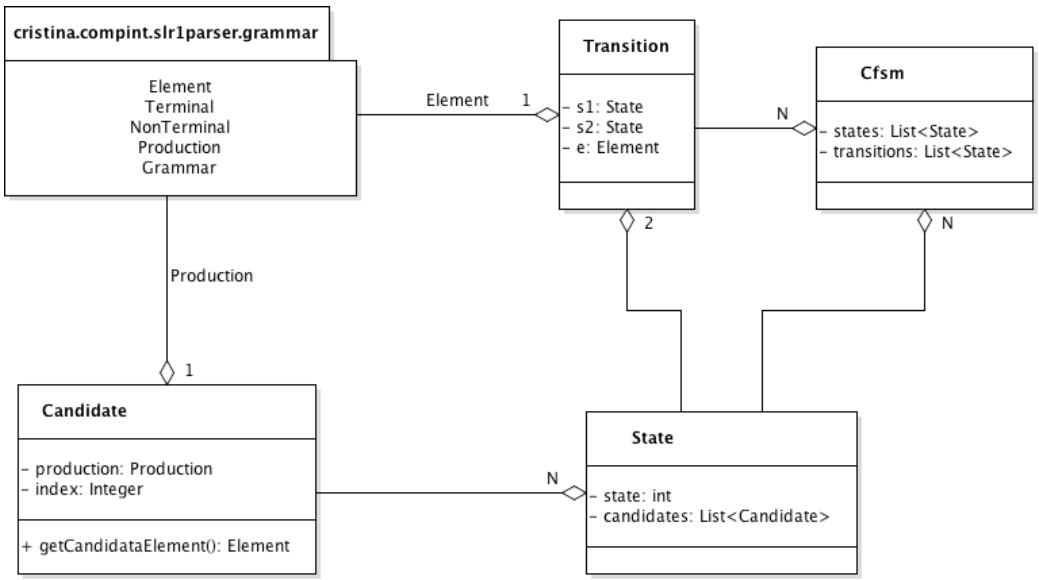

fig. 2: CFSM Class Diagram

Da evidenziare il metodo getCandidateElement() che torna l'elemento candidato oppure null nel caso siamo in presenza di una candidata di riduzione. La candidata sarà di riduzione nel caso di fine produzione, oppure nel caso in cui la produzione produca ε.

Durante la creazione della CFSM vengono verificate le ambiguità Shift-Reduce; nel caso uno stato presenti un'ambiguità non eliminabile verrà lanciata un'eccezione di tipo cristina.compint.slr1parser.exception.CfmsException contente un messaggio di errore e lo stato nel quale si è verificata l'eccezione. In tal caso non sarà possibile creare il Parser SLR1 in quanto la grammatica non è di tipo SLR1, per cui il programma terminerà.

4.5 Creazione ActionGoto

In fig. 3 è mostrato il class diagram per l'implementazione della tabella ActionGoto.

Come si vede la tabella ActionGoto necessita di tutte le classi definite precedentemente.

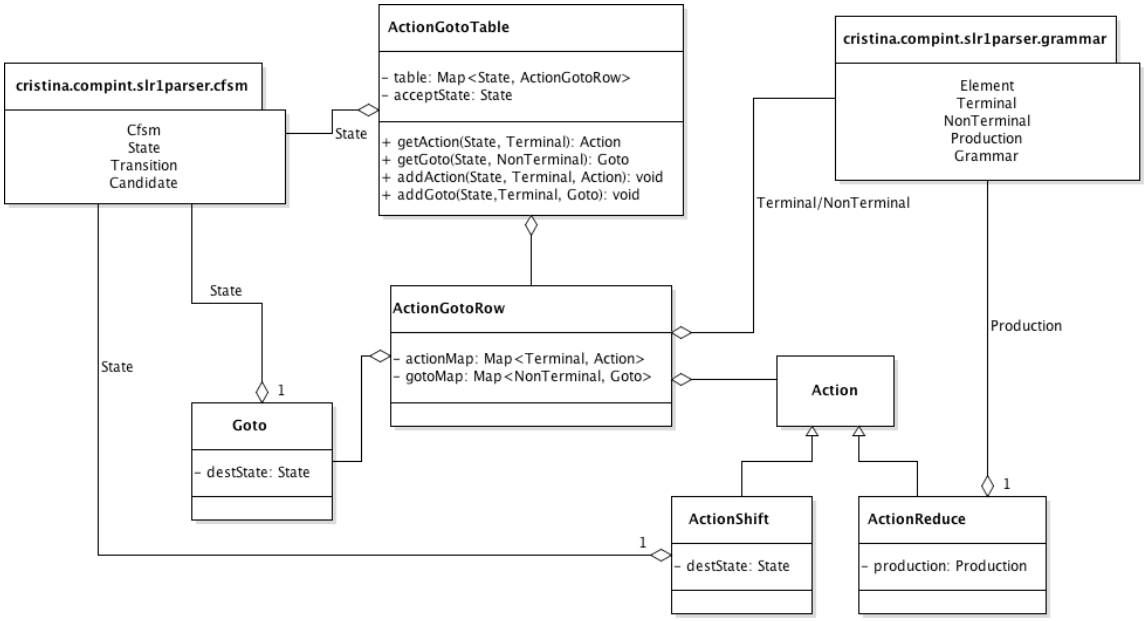

fig. 3: ActionGoto Class Diagram

4.6 Parsing

L'ultimo passo del programma consiste nel richiedere una stringa di input e verificare se la stringa inserita appartiene o meno al linguaggio. Tale attività viene effettuata a partire dalla tabella ActionGoto. Per effettuare il parsing viene definito uno stack di oggetti StackElement. Un oggetto StackElement si compone da una coppia (Element, State). Lo stack è utilizzato durante le operazioni di parsing della stringa di input.

Appendice A Esecuzione del programma

Per l'esecuzione del programma sono necessari i seguenti prerequisiti:

- JDK 1.8
- Git

Copia del repository remoto:

```
$ git clone https://github.com/cristinalombardo/slr1parser.git
```

Esecuzione del programma

```
$ cd slr1parser/dist
$ java -jar slr1parser.jar ebnf-simple.txt
```

Appendice B Compilazione con Maven

Il programma può essere compilato utilizzando Maven che permette la gestione automatica delle dipendenze e la successiva compilazione. Il programma qui descritto non presenta dipendenze, ma utilizza esclusivamente le librerie fornite con il JDK.

Copia del repository remoto:

```
$ git clone https://github.com/cristinalombardo/slr1parser.git
```

Compilazione con maven

```
$ cd slr1parser/slr1parser
$ mvn clean compile assembly:single
```

Esecuzione del programma compilato

```
$ cd target
$ java -jar slr1parser.jar ebnf-simple.txt
```