

UNIVERSITÀ DEGLI STUDI DI CATANIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

**Analisi Top-Down LL(1) di grammatiche context-free**

Relazione di Linguaggi e Traduttori

**Docente: Studenti:**

Prof.ssa Vincenza Carchiolo Floriana Leone O55000246

Giuseppe Mastrosimone O55000262

Anno Accademico 2015/2016

**Introduzione**

Lo scopo di questo progetto è quello di implementare il funzionamento di un parser top-down di tipo LL(1), cioè di effettuare l’analisi sintattica di grammatiche non contestuali (context-free) andando a considerare un solo carattere di lookahead.

In particolare, l’applicazione sviluppata è in grado di verificare se una grammatica è context-free e in caso affermativo di procedere con l’analisi LL(1). Il parser implementato è capace di analizzare le grammatiche inserite, verificando di volta in volta se esse soddisfano i requisiti di un parser LL(1) e, in tal caso, di procedere alla generazione delle relative tabelle di parsing.

**Strumenti utilizzati**

L’applicativo è stato implementato utilizzando il linguaggio OO Java. In particolare, per lo sviluppo del codice sono stati usati:

* Eclipse Luna 4.4, come IDE per lo sviluppo del codice Java;
* Libreria Batik, integrata alla piattaforma di sviluppo per la realizzazione delle tabelle di parsing con grafica SVG di tipo XML-like.

**Descrizione del codice**

**Premessa sulla definizione delle grammatiche**

Al fine di effettuare l’analisi LL(1) è necessario per prima cosa definire le grammatiche alle quali è possibile applicarla. Per questo motivo l’utente ha la possibilità di inserire da tastiera i simboli terminali e non terminali e le relative produzioni in modo da definire la grammatica desiderata. In particolare, il formato della grammatica deve essere del tipo seguente: A->aBc, dove la parte sinistra deve contenere solo un non terminale (indicato con la lettera maiuscola) e quella destra può contenere invece un numero variabile di terminali e non terminali. Seguendo questo formato viene garantito che la grammatica inserita sia di tipo non contestuale, ma non si ha la certezza che sia di tipo LL(1).

Non terminali e terminali sono differenziati dal fatto che solamente i primi possono contenere lettere maiuscole, mentre i secondi possono essere espressi da qualsiasi sequenza di caratteri purché non contenga lettere maiuscole. Per fare questa differenziazione sono state definite le opportune espressioni regolari in modo da fare un controllo sul tipo di caratteri inserito.

**File delle grammatiche**

Una volta definito il formato delle grammatiche supportato dal programma sviluppato, è bene sapere come è possibile leggerlo o definirlo.

L’applicazione permette due modalità di definizione della grammatica di cui si vuole effettuare l’analisi LL(1): lettura della grammatica da file o scrittura della grammatica su file. Il tutto è reso possibile grazie ad un menù ben definito che consente all’utente di scegliere l’opzione di sua preferenza.

In particolare, se l’utente sceglie di definire la grammatica inserendo da tastiera i valori dei terminali, dei non terminali e delle produzioni, il tutto, oltre ad essere stampato a schermo, viene salvato su 2 file, ovvero: nome\_file.txt dal quale è possibile visualizzare direttamente la grammatica inserita e nome\_file (senza estensione, utilizzato dal programma per analizzare la grammatica in qualunque momento). Questo secondo file è utile, quindi, nel caso in cui l’utente decida di effettuare l’analisi LL(1) in un secondo momento, cioè se l’utente vuole effettuare l’analisi di una grammatica già precedentemente definita in tal modo.

E’ consentita, inoltre, un’altra modalità di lettura da file della grammatica definita, in questo caso, esternamente dall’utente stesso. Ciò vuol dire che l’utente può anche scegliere di scrivere la grammatica da analizzare direttamente su un file esterno, in formato .txt, e poi utilizzare il programma per caricare tale grammatica, la quale deve essere inserita seguendo le regole di definizione delle grammatiche supportate dal programma. In particolare, il file deve essere così composto:

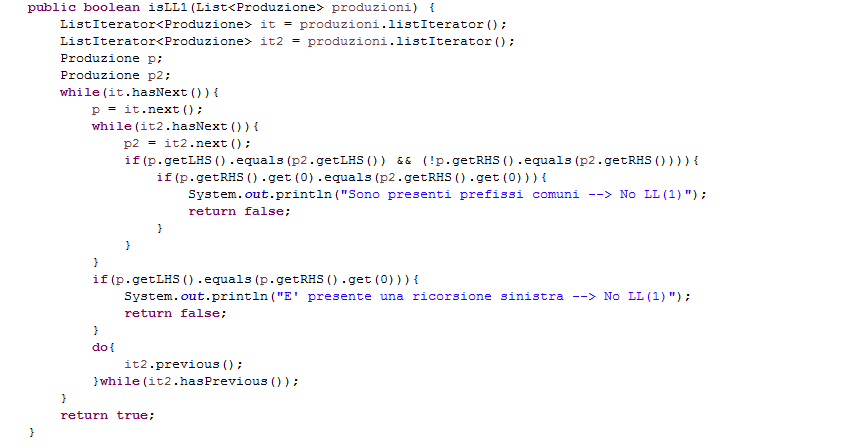
* Nella prima riga devono essere digitati i terminali separati da una virgola;
* Nella seconda riga, allo stesso modo, devono essere inseriti i non terminali, ad esclusione dello start symbol;
* Nella terza riga è inserito il simbolo iniziale;
* Dalla quarta riga devono essere inserite le produzioni della grammatica da analizzare, una produzione per ogni riga, nel formato precedentemente specificato, cioè del tipo: A->aBc;
* Alla fine di ogni riga bisogna inserire il carattere di punto e virgola “;”.

**Parser LL(1)**

Il cuore del programma è sicuramente centrato nella classa ParserLL.java. Infatti, è qui che viene implementata tutta la logica relativa al calcolo degli insiemi di First, Follow e quindi dei Predict.

Lo scopo, infatti, è proprio quello di creare l’insieme dei predict per ogni produzione una volta definita la grammatica da analizzare e per fare ciò il primo passo è quello di calcolare i relativi first e follow.

In realtà, prima ancora di fare questi calcoli è stata implementata una funzione (isLL1) in cui viene effettuato un controllo preliminare affinché la grammatica sia non ambigua, in quanto nel caso in cui siano presenti o dei prefissi comuni o ricorsione sinistra sappiamo già a priori che la grammatica non potrà mai risultare di tipo LL(1). Pertanto, nel caso in cui si rilevi una grammatica ambigua ciò viene notificato all’utente e non si procede con il calcolo di first, follow e predict.

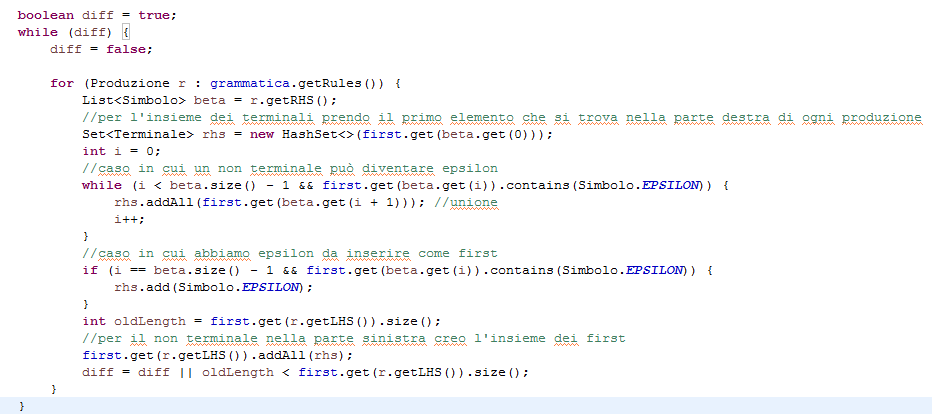


**Figura 1 –** Metodo isLL(1)

Dunque, come mostrato in Figura 1 viene fatto un controllo per ogni produzione che ha la parte sinistra (non terminali) uguale e parte destra diversa e se il primo simbolo della parte destra del terminale è coincidente per le due produzioni considerate allora abbiamo sicuramente dei prefissi comuni. Mentre se in una produzione il primo simbolo (non terminale) della parte destra è uguale al simbolo della rispettiva parte sinistra allora abbiamo una ricorsione sinistra.

**Calcolo dell’insieme dei First**

Una volta verificato che la grammatica non è ambigua si può procedere con il primo passo che è quello relativo al calcolo dei first per ogni non terminale presente nella parte sinistra della produzione.



**Figura 2 –** Calcolo dei First

Per calcolare i first in generale basta prendere il primo simbolo che sta nella parte destra della produzione. Questo però vale solo nel caso in cui il primo simbolo è un terminale. Se invece il primo simbolo è un non terminale si devono considerare 2 casi:

1. *Caso in cui il non terminale può diventare ɛ*: in questo caso si prendono come i first l’insieme dei first del non terminale insieme all’insieme dei first del simbolo successivo;
2. *Caso in cui il non terminale non può diventare ɛ*: si prendono come first l’insieme dei first del non terminale.

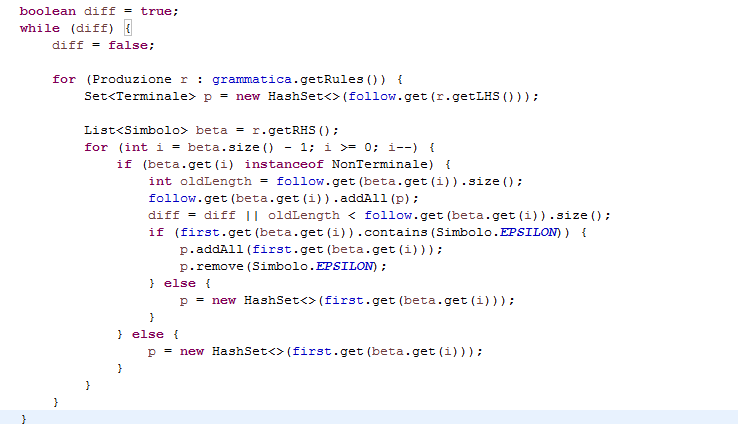
È bene tener presente che *ɛ* può essere contenuto nell’insieme dei first, diversamente dall’insieme dei follow e dei predict.

**Calcolo dei Follow**

Il passo successivo consiste nel calcolo dell’insieme dei Follow. Il follow rappresenta il simbolo terminale che si può trovare a seguito di un non terminale delle varie produzioni.

In questo caso è necessario considerare un simbolo particolare, che nel nostro caso viene denominato come EOF e rappresenta il fine stringa ($).

Per la definizione dei follow si va a considerare ogni non terminale che si trova nella parte destra della produzione e si vede quali sono i simboli che lo seguono.



**Figura 3 –** Calcolo dei Follow

In generale per calcolare i follow di un non terminale si considera il first del simbolo successivo, se non è presente il simbolo successivo si prende il follow del non terminale nella parte sinistra della produzione.

Tuttavia, è bene tenere presente che nell’insieme dei follow non può essere presente *ɛ,* e questo spiega perché viene rimosso dall’insieme dei follow.

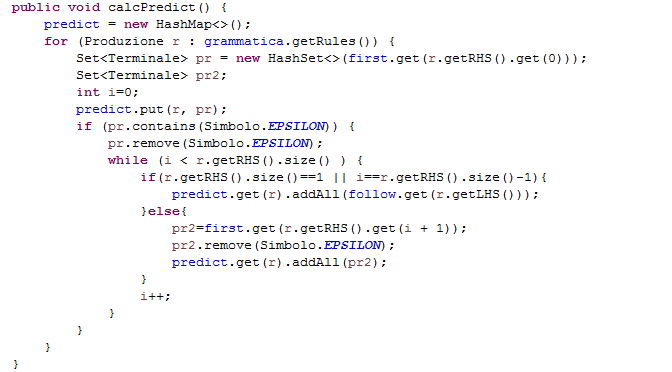
Come per i first si devono considerare due casi, a seconda se il simbolo successivo può essere *ɛ* o meno:

1. Se il simbolo successivo al non terminale di riferimento non può diventare *ɛ* si prende semplicemente il suo insieme dei first;
2. Altrimenti, oltre al suo insieme dei first si prende il follow del non terminale a sinistra della produzione.

**Calcolo dell’insieme dei Predict**

Calcolati i first e i follow è necessario definire l’insieme dei predict che ci indicano per ogni produzione quale carattere di lookahead andare a considerare quando facciamo l’analisi sintattica.

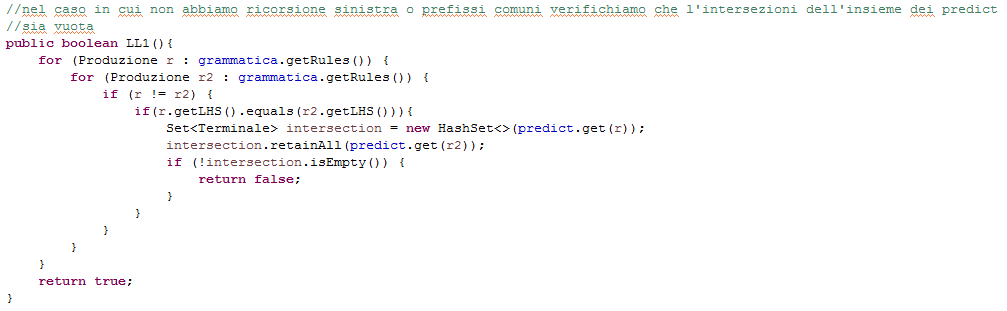
Diversamente dai first e i follow, i predict vanno calcolati per ogni singola produzione della nostra grammatica. Questa operazione corrisponde al punto cruciale per definire se la grammatica è di tipo LL(1) o meno e una volta fatta questa analisi sarà possibile creare opportunamente la tabella di parsing per una determinata grammatica che soddisfi i requisiti per essere LL(1).



**Figura 4 –** Calcolo dell’insieme dei Predict

Per calcolare l’insieme dei predict per ogni produzione si prende il first di ogni del primo simbolo che sta nella parte destra della produzione. Come per i follow, anche in questo caso non è possibile contenere il carattere *ɛ* che sarà quindi rimosso nel caso in cui è contenuto nell’insieme dei first. Inoltre sempre in questo caso, oltre all’insieme dei first (escluso *ɛ*) si inserisce anche l’insieme dei follow del non terminale che sta nella parte sinistra della produzione.

Una volta calcolati i predict per ogni produzione è necessario effettuare un ulteriore controllo per verificare se effettivamente la grammatica è di tipo LL(1), controllo necessario per generare una tabella di parsing senza conflitti.



**Figura 5 –** Metodo LL1()

Per verificare ciò, pertanto, è necessario effettuare l’intersezione dell’insieme dei predict delle produzioni che hanno la stessa parte sinistra. Se l’intersezione risulta vuota, allora non avremo conflitti e quindi sarà possibile generare correttamente la tabella di parsing. In caso contrario, avremo la certezza che la grammatica non è di tipo LL(1) e non sarà possibile generare una tabella di parsing senza che vi siano dei conflitti.

**Interfaccia**

**Menù**

Come già anticipato, quando il programma viene mandato in esecuzione viene avviato un menù.



Come mostrato in figura ci sono 3 opzioni, di cui 2 sono relative al funzionamento del programma stesso: lettura della grammatica da un file e scrittura della grammatica su file. Di seguito verranno illustrati i passaggi relativi al funzionamento dell’applicativo.

**Esempio 1: Inserimento di una grammatica da tastiera**

Qui verranno descritti i passaggi che permettono all’utente di definire la grammatica da analizzare, tenendo in considerazione che essa è rappresentata da una quadrupla contenente {T, V, S, P}, ovvero l’insieme dei terminali (T), dei non terminali (V), lo start symbol o assioma (S) e l’insieme P delle produzioni.

Passo 1)

Digitando ‘2’ verrà visualizzata la seguente richiesta di inserimento:



Come è indicato, bisogna digitare il numero di terminali che deve contenere la grammatica da inserire.

Passo 2)

Questo passo verrà ripetuto un numero di volte uguale al valore scelto precedentemente:



Una volta completato il passo 2, tutti i terminali digitati saranno inseriti nella lista dei terminali usati per definire la nuova grammatica. I passi successivi, allo stesso modo, permetteranno di inserire i restanti elementi della quadrupla che definisce una grammatica.

Passo 3)



Passo 4)



Passo 5)



In questo passo verrà scelto il simbolo iniziale, che verrà quindi inserito nell’insieme dei non terminali.

Passo 6)



Passo 7)



Qui viene visualizzato a schermo l’insieme dei non terminali che sono stati inseriti in modo da agevolare l’utente nell’inserimento della parte sinistra delle produzioni.

Passo 8)



Passo 9)

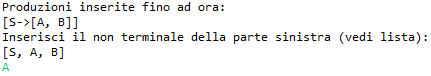


Come per la parte sinistra, viene visualizzato a schermo l’insieme dei simboli che sono stati inseriti in modo da agevolare l’utente nell’inserimento della parte destra delle produzioni. Dopo essere stato inserito l’ultimo simbolo apparirà il seguente messaggio:



Passo 10)

Una volta inserita una produzione, se ne devono essere aggiunte altre, apparirà una schermata di questo tipo:



Vengono cioè visualizzate le produzioni che sono state inserite fino ad ora, così da dare all’utente maggior sicurezza nella prosecuzione dell’esecuzione del programma. Quindi il programma prosegue ripetendo i passi 8 – 10 fino al completamento dell’inserimento delle produzioni. Dopodiché verrà mostrato il messaggio:

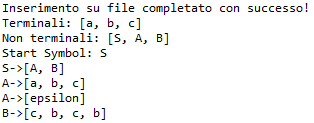


Passo 11)

Adesso deve essere inserito il nome del file sul quale sarà salvata la grammatica appena inserita, in modo da poterla visualizzare ed analizzare in un secondo momento.



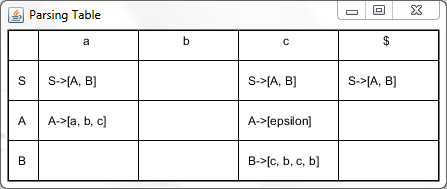
Una volta digitato il nome del file, esso verrà creato e il suo contenuto verrà stampata a schermo:



Inoltre, il programma proseguirà andando ad effettuare l’analisi LL(1) della grammatica, mostrando gli insieme dei first, follow e predict:



Infine, se la grammatica è LL(1), viene generata la tabella di parsing:



A questo punto il programma torna al menù iniziale, permettendo di creare una nuova grammatica da tastiera o di sceglierne da un file già presente.

**Esempio 2: Caricamento di una grammatica da file esterno:**

Qui verranno descritti invece i passaggi che permettono all’utente di caricare una grammatica da un file.

Passo 1)

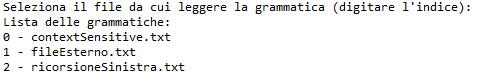
Digitando ‘1’ verrà visualizzata la seguente schermata:



in cui bisogna specificare se si vuole leggere una grammatica attraverso un file esterno oppure se si vuole caricare una grammatica precedentemente creata attraverso questo programma.

Passo 2)

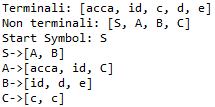
Digitando “s” si confermerà l’intenzione di caricare la grammatica da un file esterno e apparirà la lista dei file presenti nella directory dedicata a contenerli:



Passo 3)

A questo punto bisognerà scegliere il file da cui estrarre la grammatica, ad esempio il file “file Esterno.txt”.

Digitando “1”, il programma leggerà il file analizzando la grammatica e stampandola a schermo, mostrando quindi gli elementi della quadrupla che caratterizza la grammatica:



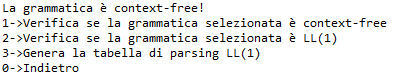
Verrà quindi richiesto all’utente quali azioni effettuare su questa grammatica:



Nei passi successivi sono mostrati i risultati dati digitando i comandi “1”, “2” e “3”.

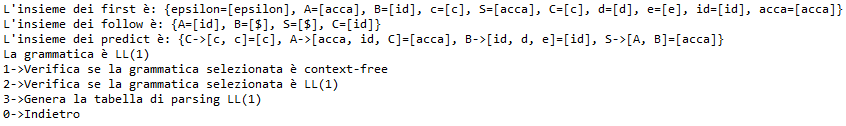
Passo 4A)

Digitando “1” il programma controlla se la grammatica caricata è context-free e stampa a schermo il risultato ottenuto, dopodiché torna a chiedere all’utente quale azione vuole effettuare:



Passo 4B)

Digitando “2” verrà effettuato un’analisi LL(1) sulla grammatica in questione e, in caso affermativo, verranno mostrati gli insiemi di first, follow e predict:



Vengono poi nuovamente mostrate le opzioni all’utente sulle azioni da effettuare sulla grammatica.

Passo 4C)

Digitando “3” si procederà alla generazione della tabella di parsing LL(1) relativa a questa grammatica, che verrà visualizzata in una nuova finestra:

