**Università degli studi di Bergamo**

**Facoltà di Ingegneria**

**Corso di Laurea Magistrale in Ingegneria Informatica**



**Linguaggi formali e compilatori**

**(COD.CORSO: 38070)**

**Anno accademico: 2018/2019**

**Settore scientifico-disciplinare: ING-INF/05**

**Dipartimento: Ingegneria gestionale, dell'informazione e della produzione**

**Titolo:**

**Riconoscitore di grammatiche LR(1)**

**Autori:**

**Luca Filice Matr: 1026838**

**Matteo Gusmini Matr: 1035426**

**Davide Presciani Matr: 1035189**

Indice

[Introduzione 3](#_Toc3734864)

[Funzionamento 4](#_Toc3734865)

[Conclusioni 5](#_Toc3734866)

# Introduzione

In informatica e in linguistica, una grammatica libera dal contesto è una grammatica formale in cui ogni regola sintattica è espressa sotto forma di derivazione di un simbolo a sinistra a partire da uno o più simboli a destra. Ciò può essere espresso con due simbolismi equivalenti:

1. S := α
2. S → α

dove S è un simbolo non terminale e α è una sequenza di simboli terminali e non terminali.

L'espressione "libera dal contesto" si riferisce al fatto che il simbolo non terminale S può sempre essere sostituito da α, indipendentemente dai simboli che lo precedono o lo seguono. Un linguaggio formale si dice libero dal contesto se esiste una grammatica libera dal contesto che lo genera.

Le grammatiche libere dal contesto sono abbastanza potenti da descrivere la sintassi della maggior parte dei linguaggi di programmazione; al tempo stesso, sono abbastanza semplici da consentire un parsing molto efficiente.

La notazione formale di Backus-Naur (BNF) è la sintassi più comunemente usata per descrivere grammatiche libere dal contesto.

Un parser LR è un parser di tipo Bottom-up per grammatiche libere da contesto, usate molto di frequente nei compilatori dei linguaggi di programmazione. Un Parser LR legge il proprio input partendo da sinistra verso destra, producendo una derivazione destra. A volte questo parser viene anche indicato col nome “Parser LR(k)” dove k si riferisce al numero di simboli letti (ma non "consumati") utilizzati per prendere le decisioni di parsing.

# Funzionamento

Questo riconoscitore di grammatiche LR(1) permette di verificare se una data grammatica in ingresso è o meno di tipo LR(1).

Inoltre, questo riconoscitore genera un elenco degli stati ognuno con relative regole di core e completamento complete di look-ahead e una lista delle transizioni tra i vari stati

Questo programma riconosce due diversi tipi di regole di produzione: una prima regola (pr) che deve contenere il non terminale ‘S0’ e tutte le altre regole di produzione (ar) che formano la grammatica.

I caratteri accettati dall’applicazione e le loro rispettive categorie sono:

|  |  |  |  |
| --- | --- | --- | --- |
| Simbolo | Categoria | Caratteri | |
| SZ | Stato iniziale | S0 |
| EQ | Simboli di derivazione | -> | := |
| NT | Non terminali | A … Z |
| CT | Caratteri terminali | a … z | 0 ... 9 | + | - | \* | / |
| TER | Terminatori | /swa | /cjswa |
| SC | Fine regola | ; |

La struttura del programma prevede all’inizio la prima regola che deve essere nella forma

SZ EQ NT SC

Successivamente si possono inserire un qualsiasi numero di regole maggiore di uno per formare la grammatica, queste regole devono essere nella forma:

NT EQ (NT|CT)\* SC

Per procedere all’analisi è sufficiente creare un file di tipo txt denominato “nomefile.txt” nella stessa cartella dell’applicativo jar procedendo poi a eseguire l’applicativo.

A questo punto il riconoscitore costruirà l’automa di tipo LR(1) secondo la grammatica fornita, andando a individuare eventuali conflitti che potrebbero rendere la grammatica stessa non LR(1).

Al termine dell’analisi, salvo che non siano stati trovati degli errori nel file passato in input (errori di tipo lessicale o grammaticale), il programma fornisce in output un elenco degli stati formati da regole core e di completamento con i relativi look-ahead, inoltre fornisce un elenco delle transizioni che indicano quale carattere viene consumato per passare da uno stato all’altro, infine viene riportato se la grammatica è o meno LR(1) ed in caso non lo sia riporta gli stati che contengono dei conflitti che rendono la grammatica non LR(1).

# Conclusioni