Grammatiche e linguaggi

Andrea Canale

December 27, 2024

Contents

1 Grammatica **1**

2 Derivazioni **2**

3 Sintassi BNF **2**

4 Tipi di grammatiche **3**

4.1 Grammatiche di tipo 0 3

4.2 Grammatiche di tipo 1 3

4.3 Grammatiche di tipo 2 3

4.4 Grammatiche di tipo 3 3

5 Automi a stati finiti non deterministici **3**

6 Da automi non deterministici a automi deterministici **5**

1 Grammatica

Una grammatica è costituita dà:

- Un insieme finito di simboli non terminali N
- Un insieme finito di simboli terminali I
- Un insieme di produttorie
- Un simboli di partenza σ

Denotiamo con $G = (N, T, P, \sigma)$

Una produttoria associa un simbolo non terminale ad un simbolo terminale oppure ad una

stringa formata da simboli terminali e non terminali

Un linguaggio è l'insieme di tutte le parole che si possono formare con questa grammatica.

Una parola è formata da una grammatica quando non abbiamo più simboli non terminali nella stringa.

Esempio:

Data la grammatica:

$$Frase \rightarrow SN\ SV$$

$$SN \rightarrow Articolo\ N$$

$$N \rightarrow Studente \mid Libro$$

$$Articolo \rightarrow Il \mid lo$$

$$SV \rightarrow Leggi\ SN \mid Brucia\ SN$$

Dove Frase è il simbolo d'inizio, possiamo scrivere la frase: Lo studente legge il libro. Notiamo che sulla sinistra abbiamo i simboli non terminali e sulla destra abbiamo le produttorie, dove la pipe significa "oppure"

2 Derivazioni

Una stringa s_1 è derivabile da un altra s_2 se nella loro grammatica esiste una produttoria(o un insieme di produttorie) tale che si riesce ad ottenere la stringa s_2 partendo da s_1.

Il linguaggio generato da una grammatica consiste in tutte le stringhe derivate in maniera finita. **Una parola derivata infinite volte non ne fa parte.**

Si dice che una parola è derivata da una grammatica se si può scrivere come attraverso una grammatica.

3 Sintassi BNF

La sintassi BNF(o backus normal form oppure Backus-Naur form) è una sintassi per produrre grammatiche comoda per essere trascritta su computer.

La produttoria $S \rightarrow T$ si scrive come $S ::= T_1|T_2|...|T_n$

4 Tipi di grammatiche

4.1 Grammatiche di tipo 0

Grammatiche molto generali, si basana su insieme ricorsivamente numerali.

4.2 Grammatiche di tipo 1

$A\sigma B \to A\sigma$

Grammatiche contestuali(context-sensitive), c'è almeno una termine nonterminale in una stringa. Inoltre la parola vuota non può essere usata in una produttoria.

4.3 Grammatiche di tipo 2

$A \to \sigma$

Grammatiche context-free. C'è sempre un simbolo non terminale sulla sinistra. A destra c'è qualsiasi simbolo.

4.4 Grammatiche di tipo 3

Grammatiche regolari. Formati da:

$A \to \sigma$ oppure $A \to aB$ oppure $A \to \lambda$

Sono equivalenti agli automi a stati finiti.

A e B in questi esempi sono non terminali mentre σ è un terminale.

Un linguaggio si identifica in base al tipo di grammatica.

Notiamo inoltre che due grammatiche sono equivalenti se generano lo stesso linguaggio.

Notiamo che una grammatica regolare è anche context-free e che una grammatica context-free senza forme $A \to \lambda$ è una grammatiche context-sensitive. Una grammatica di questo tipo è anche regolare

5 Automi a stati finiti non deterministici

Possiamo convertire una grammatica in automi a stati finiti che ha le seguenti caratteristiche:

Grammatica	Automi
Insieme dei simboli terminali	Alfabeto di input
Stati	Insieme dei simboli non terminali
Stato iniziale	Simboli di partenza
Funzione di transizione	Produttorie
Stati accettanti	Simboli S tali che $S \to \lambda$

L'insieme delle stringhe generate da una grammatica è lo stesso accettato da un automa di questo tipo.

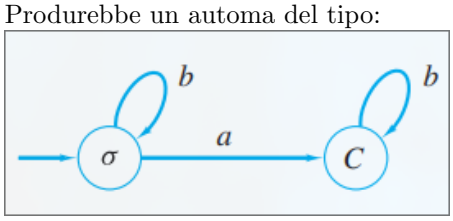
Tuttavia alcune grammatiche non possono essere rappresentate con automi a stati finiti deterministici, ad esempio:

Consider the regular grammar defined by $T = \{a, b\}$ and $N = \{\sigma, C\}$, with productions

$$\sigma \to b\sigma,\ \sigma \to aC,\ C \to bC,\ C \to b,$$

and starting symbol σ.

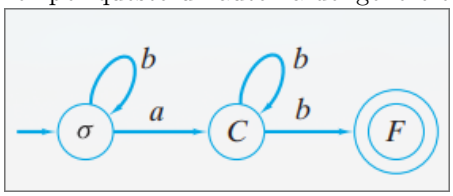
Produrebbe un automa del tipo:



Tuttavia la produttoria $C \to b$ non può essere rappresentata. Allora introduciamo uno stato ausialiario F tale che:

$$C \to bF \text{ e } F \to \lambda$$

Tuttavia questo stato non ha successive transizioni e quindi il suo comportamento è "imprevedibile" per questo un automa del genere è detto non deterministico.



Un automa a stati finiti non deterministico è caratterizzato da:

- Un insieme finito di simboli I in entrata
- Un iniseme finito di stati S
- Una funzione $f : SxI \to P(S)$ per lo stato successivo
- Una sottoinsieme di S di stati accettanti A
- Uno stato iniziale σ

Notiamo che non tutti gli stati hanno una funziona di transizione associata.

Perchè una stringa sia accettata in un automa a stati finiti non deterministici dobbiamo controllare che:

- Esiste una sequenza di stati che determina la stringa(tra le varie possibili)
- Lo stato finale è uno stato accettante

Data una grammatica regolare G, un automa non deterministico a stati finiti appropriato, accetta l'intero linguaggio generato da G.

Un automa di questo tipo accetta solo linguaggi generati da grammatiche regolari

Un linguaggio è regolare se e solo se esiste un automa non deterministico che accetta tutte le stringhe di quel linguaggio.

L'intersezione tra linguaggi regolare è regolare. Inoltre l'automa prodotto da questa intersezione è l'automa definito dall'unione dell'input I_1, I_2

Possono esistere più di 2^n stati.

6 Da automi non deterministici a automi deterministici

Per convertire un automa a stati finiti non deterministici ad un automa a stati finiti deterministici seguiamo la seguente tabella:

Non deterministico	Deterministico
Simboli in input	Simboli in input
Stati	Insieme potenza di tutti gli stati
Stato iniziale	Stato iniziale
Funzione di transizione	$\bigcup_{S \in X} f(S, x) = Y$
Stati accettanti	Insieme potenza degli stati accettanti