Indice

1 Grammatiche

Una grammatica è una quadrupla $\langle V, \Sigma, P, S \rangle$, dove

- V è l'insieme finito e non vuoto delle variabili, questi vengono anche chiamati non terminali. Questo lo possiamo chiamare anche l'alfabeto delle variabili.
- Σ è l'insieme finito e non vuoto dei termianli, o anche alfabeto dei terminali.
- P è l'insieme finito delle regole di produzione. Queste sono della forma

  $$\alpha \to \beta$$

  con $\alpha \in (V \cup \Sigma)^+$ e $\beta \in (V \cup \Sigma)^*$.
- S è un elemento di V che chiamiamo simbolo iniziale o assioma.

Per $x, y \in (V \cup \Sigma)^*$, diciamo che x deriva y, indicato con $x \Rightarrow y$ (opzionalmente indicando la grammatica $x \underset{G}{\Rightarrow} y$), se e solo se

$$\exists (\alpha \to \beta) \in P, \exists \eta, \delta \in (V \cup \Sigma)^* \mid x = \eta\alpha\delta \wedge y = \eta\beta\delta$$

Inoltre diciamo che x deriva y in k passi, con $k \in \mathbb{N}$, indicato $x \overset{k}{\Rightarrow} y$, se e solo se

$$\exists x_0, x_1, \ldots, x_k \mid x_0 = x \wedge x_k = y \wedge \forall i \in 1, \ldots, k x_{i-1} \Rightarrow x_i$$

Infine diciamo che x deriva y in un certo numero di passi, indicato con $x \overset{*}{\Rightarrow} y$, se e solo se

$$\exists k \geq 0 \mid x \overset{*}{\Rightarrow} y$$

e che x deriva y in almeno un passo, indicato con $x \overset{+}{\Rightarrow} y$, se e solo se

$$\exists k > 0 \mid x \overset{*}{\Rightarrow} y$$

Definiamo il linguaggio generato dalla grammatica G

$$L(G) = w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w$$

Una stringa di terminali e non terminali è detta forma sentenziale.

Due grammatiche G_1 e G_2 sono equivalenti se e solo se $L(G_1) = L(G_2)$.

Sia $\Sigma = (,)$. Costruiamo V, inizialmente contentente solo S. Ora la sequenza vuota è bilanciata, quindi $S \to \epsilon$. Una sequenza di parentesi bilanciata rimane bilanciata se inserita tra due parentesi, quindi $S \to (S)$. Infine la concatenazione di due sequenze bilanciate è ancora bilanciata, quindi $S \to SS$.
Ad esempio

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow ((S)) \\ &\Rightarrow ((SS)) \\ &\Rightarrow (((S)S)) \\ &\Rightarrow (((S)(S))) \\ &\Rightarrow ((()(S))) \\ &\Rightarrow ((()())) \end{aligned}$$

Esercizio: costruire il linguaggio delle sequenze correttamente bilanciate di quadre e tonde aperte e chiuse in cui le quadre non possono stare

Data la grammatica $\langle S, B, a, b, c, \ldots, S \rangle$.

$$
\begin{aligned}
S &\to aBSc \\
S &\to abc \\
Ba &\to aB \\
Bb &\to bb
\end{aligned}
$$

Eseguiamo una derivazione

$$
\begin{aligned}
S &\Rightarrow aBSc \\
&\Rightarrow aBabcc \\
&\Rightarrow aaBbcc \\
&\Rightarrow aabbcc \\
S &\Rightarrow aBSc \\
&\Rightarrow aBaBScc \\
&\Rightarrow aaBBScc \\
&\Rightarrow aaBBabccc \\
&\Rightarrow aaBaBbccc \\
&\Rightarrow aaaBBbccc \\
&\Rightarrow aaaBbbccc \\
&\Rightarrow aaabbbccc
\end{aligned}
$$

Si può dimostrare che il linguaggio generato da questa grammatica è $L = a^n b^n c^n \mid n > 0$.

Le grammatiche sono classificate in base alla forma delle produzioni, possiamo definire quattro tipi di grammatiche:

tipo 0	nessu
tipo 1	Se $\alpha \to \beta$ è una produzione, allora $\lvert\beta\rvert \geq \lvert\alpha\rvert$. Equivalentemente una grammatica in cui t
tipo 2	Se $\alpha \to \beta \in P$, all
tipo 3	Possiamo avere produzioni solo della

Ogni grammatica restringe il livello precedente

1.1 Classificazione dei linguaggi

Sia $\mathscr{L} \subseteq \Sigma^*$, diciamo che $\mathscr{L}$ è di tipo $0 \leq i \leq 3$, se e solo se $\exists G$ di tipo i tale che $\mathscr{L} = L(G)$. Questa classificazione genera una gerachia dei linguaggi, detta gerarchia di Chomksy, e chiamiamo

- i linguaggi di tipo 3 come linguaggi regolari
- i linguaggi di tipo 2 come linguaggi conxtext free o CF
- i linguaggi di tipo 1 come linguaggi context sensitive o CS
- i linguaggi di tipo 0 come linguaggi ricorsivamente enumerabili (RE) o semidecibili

I linguaggi di tipo 1, 2 e 3 sono anche detti decidibili o ricorsivi. (Un insieme ricorsivo è un insieme per cui posso avere una macchina che può rispondere sì o no alla domanda se un insieme appartenga)

E per ognuna di queste classi esiste un modello di macchina riconoscitrice:

- per i linguaggi di tipo 3 sono gli automi a stati finiti
- per i linguaggi di tipo 2 sono gli automi a pila
- per i linguaggi di tipo 1 sono gli automi limitati linearmente, questi hanno un nastro finito che possono modificare
- per i linguaggi di tipo 0 sono le macchine di Turing, nella variante più semplice queste sono macchine con un nastro infinito su cui inizialmente c'è scritto l'input. Per questi linguaggi se la stringa appartiene al linguaggio il riconoscitore termina sempre con sì, mentre se la stringa non appartiene al linguaggio il riconoscitore può anche non terminale ($\perp$)

Sia G una grammatica di tipo 1 e w una stringa di terminali di lunghezza $n = |w|$, voglio sapere se w apparteiene a G. Siccome la grammatica di tipo 1 è monotonica (le regole forme sentenziali non possono decrescere), queste non possono mai superare n durante la derivazione di w. Chiamiamo

$$T_i = \gamma \in (V \cup \Sigma)^{\leq n} \mid S \overset{\leq i}{\Rightarrow} \gamma$$

Costruiamo gli insiemi T_i per $i = 0, 1, \ldots$

$$T_0 = S$$
$$\vdots$$
$$T_i = T_{i-1} \cup \gamma \in (V \cup \Sigma)^{\leq n} \mid \exists \beta \in T_{i-1}.\beta \Rightarrow \gamma$$

Allora

$$T_0 \subseteq T_1 \subseteq \cdots \subseteq T_{i-1} \subseteq T_i \subseteq \cdots \subseteq (V \cup \Sigma)^{\leq n}$$

Dato che $(V \cup \Sigma)^{\leq n}$ è un insieme finito, quindi esiste un certo i tale che $T_i = T_{i-1}$. A questo punto ho trovato tutte le forme sentenziali di lunghezza n, e per vedere se w appartiene al linguaggio basta controllare che $w \in T_i$.

Quello che posso fare è considerare l'insieme

$$U_i = \gamma \in (V \cup \Sigma)^* \mid S \overset{\leq i}{\Rightarrow} \gamma$$

Quindi ancora

$$U_0 = S$$
$$\vdots$$
$$U_i = U_{i-1} \cup \gamma \in (V \cup \Sigma)^* \mid \exists \beta \in U_{i-1}.\beta \Rightarrow \gamma$$

$$U_0 \subseteq U_1 \subseteq \cdots \subseteq U_{i-1} \subseteq U_i \subseteq \cdots \subseteq (V \cup \Sigma)^*$$

Quindi che nei linguaggi di tipo 0 perdo la monotonicità delle derivazioni, non posso più trovare una i tale che $U_i = U_{i-1}$. Però per ogni U_i posso controllare se $w \in U_i$, infatti se la grammatica genera w, prima o poi questa la troverò, se non la genera non la troverò mai.

Si dicono ricorsivamente enumerabili perché significa che con un programma posso elencare tutti gli elementi del linguaggi.