Linguaggi

a.a. 2021-2022

Corso di Fondamenti di Informatica - 1 modulo

Corso di Laurea in Informatica

Università di Roma "Tor Vergata"

Prof. Giorgio Gambosi

Alfabeto

Un insieme finito non vuoto Σ di simboli (detti caratteri) prende il nome di alfabeto.

Dato un alfabeto Σ, denotiamo come $\langle \Sigma^*, \circ, \varepsilon \rangle$ il monoide libero definito su Σ.

- Σ^* chiuso rispetto a $\circ$: $\forall x, y \in \Sigma^* : x \circ y \in \Sigma^*$
- $\circ$ associativa: $\forall x, y, z \in \Sigma^* : (x \circ y) \circ z = x \circ (y \circ z)$
- esiste un elemento neutro ε: $\forall x \in \Sigma^* : x \circ \varepsilon = \varepsilon \circ x = x$

Alfabeto

Gli elementi di Σ^* vengono detti parole o stringhe: si noti che, mentre Σ è finito, Σ^* è infinito.

L'elemento ε viene detto parola vuota.

L'operazione $\circ : \Sigma^* \times \Sigma^* \mapsto \Sigma^*$ definita sul monoide è chiamata concatenazione e consiste nel giustapporre due parole di Σ^*:

$$x_{i_1} \dots x_{i_n} \circ y_{j_1} \dots y_{j_m} = x_{i_1} \dots x_{i_n} y_{j_1} \dots y_{j_m},$$

con $x_{i_1}, \dots, x_{i_n}, y_{j_1}, \dots, y_{j_m} \in \Sigma$.

La concatenazione di due stringhe x e y è frequentemente indicata omettendo il simbolo $\circ$, cioè scrivendo xy anziché $x \circ y$.

Concatenazione

Con la notazione $|x|$ indichiamo la lunghezza di una parola x, ovvero il numero di caratteri che la costituiscono. Chiaramente $|\varepsilon| = 0$.

La concatenazione non gode della proprietà commutativa e quindi in generale:

$$x \circ y \neq y \circ x.$$

Un caso particolare di concatenazione è quello in cui la stringa viene concatenata con sé stessa: con x^h si denota la concatenazione di x con sé stessa iterata h volte.

Per convenzione, si pone $x^0 = \varepsilon$.

Linguaggi

Dato un alfabeto Σ, si definisce linguaggio un qualsivoglia sottoinsieme di Σ^*.

Poiché $\Sigma \subseteq \Sigma^*$, un alfabeto è a sua volta un linguaggio.

Si chiama linguaggio vuoto, e lo si indica con Λ, il linguaggio che non contiene stringa alcuna: $|\Lambda| = 0$.

Si noti che $\Lambda \neq \{\varepsilon\}$.

Intersezione di linguaggi

L'intersezione di due linguaggi L_1 e L_2 è il linguaggio $L_1 \cap L_2$ costituito dalle parole di L_1 e di L_2

$$L_1 \cap L_2 = \{x \in \Sigma^* \mid x \in L_1 \wedge x \in L_2\}$$

Unione di linguaggi

L'unione di due linguaggi L_1 e L_2 è il linguaggio $L_1 \cup L_2$ costituito dalle parole appartenenti ad almeno uno fra L_1 ed L_2

$$L_1 \cup L_2 = \{x \in \Sigma^* \mid x \in L_1 \vee x \in L_2\}$$

Si noti che $L_1 \cap \Lambda = \Lambda$ e $L_1 \cup \Lambda = L_1$.

Complementazione di linguaggi

Il complemento di un linguaggio L_1 è il linguaggio $\overline{L}_1 = \Sigma^* - L_1$ costituito dalle parole appartenenti a Σ^* ma non ad L_1

$$\overline{L}_1 = \{x \in \Sigma^* \mid x \notin L_1\}$$

Concatenazione di linguaggi

La concatenazione (o prodotto) di due linguaggi L_1 e L_2 è il linguaggio $L_1 \circ L_2$ delle parole costituite dalla concatenazione di una stringa di L_1 e di una stringa di L_2

$$L_1 \circ L_2 = \{x \in \Sigma^* \mid \exists y_1 \in L_1 \; \exists y_2 \in L_2 \; (x = y_1 \circ y_2)\} \, .$$

Si noti che $L \circ \{\varepsilon\} = \{\varepsilon\} \circ L = L$, e che $L \circ \Lambda = \Lambda \circ L = \Lambda$.

Potenza di linguaggi

La potenza L^h di un linguaggio è definita come

$$L^h = L \circ L^{h-1}, h \geq 1$$

con la convenzione secondo cui $L^0 = \{\varepsilon\}$.

Si noti che, in base alla suddetta convenzione, $\Lambda^0 = \{\varepsilon\}$.

Chiusura di linguaggi

Il linguaggio L^* definito da

$$L^* = \bigcup_{h=0}^{\infty} L^h$$

prende il nome di chiusura riflessiva del linguaggio L rispetto all'operazione di concatenazione, mentre l'operatore "$*$" prende il nome di iterazione o stella di Kleene.

Dato un qualunque linguaggio L, $\varepsilon \in L^*$ e $\Lambda^* = \{\varepsilon\}$.

Si indica con L^+ la chiusura (non riflessiva) definita da

$$L^+ = \bigcup_{h=1}^{\infty} L^h$$

Risulta ovviamente $L^* = L^+ \cup \{\varepsilon\}$.

Espressioni regolari

Dato un alfabeto Σ e dato l'insieme di simboli

$$\{+, ^*, (,), \cdot, \emptyset\}$$

si definisce espressione regolare sull'alfabeto Σ una stringa

$$r \in (\Sigma \cup \{+, *, (,), ., \emptyset\})^+$$

tale che valga una delle seguenti condizioni:

1. $r = \emptyset$
2. $r \in \Sigma \cup \{\varepsilon\}$
3. $r = (s + t)$, oppure $r = (s \cdot t)$, oppure $r = s^*$, dove s e t sono espressioni regolari sull'alfabeto Σ.

Espressioni regolari

Le espressioni regolari consentono di rappresentare linguaggi mediante una opportuna interpretazione dei simboli che le compongono. Nella tabella si mostra la corrispondenza tra un'espressione regolare r e il linguaggio $\mathscr{L}(r)$ che essa rappresenta.

Espr. regolari	Linguaggi
$\emptyset$	Λ
a	$\{a\}$
$(s + t)$	$\mathscr{L}(s) \cup \mathscr{L}(t)$
$(s \cdot t)$	$\mathscr{L}(s) \circ \mathscr{L}(t)$
s^*	$\big(\mathscr{L}(s)\big)^*$

Espressioni regolari: esempio

L'espressione regolare $(a + b)^* a$ rappresenta il linguaggio

$$
\begin{aligned}
\mathcal{L}\left((a+b)^* a\right) &= \mathcal{L}\left(\left(a+b\right)^*\right) \circ \mathcal{L}\left(a\right) \\
&= \left(\mathcal{L}\left(a+b\right)\right)^* \circ \mathcal{L}\left(a\right) \\
&= \left(\mathcal{L}\left(a\right) \cup \mathcal{L}\left(b\right)\right)^* \circ \{a\} \\
&= \left(\{a\} \cup \{b\}\right)^* \circ \{a\} \\
&= \{a, b\}^* \circ \{a\} \\
&= \{x \mid x \in \{a, b\}^+, x \text{ termina con } a\}.
\end{aligned}
$$

Espressioni regolari: esempio

Sia $c = \{0, 1, \ldots, 9\}$.

L'espressione regolare che rappresenta i numeri reali nella forma $c_{i1}c_{i2}\ldots c_{in}.c_{f1}c_{f2}\ldots c_{fm}$ (dove c_{ij} rappresenta la j-esima cifra prima del punto decimale, e c_{fk} la k-esima cifra dopo il punto decimale) è

$$\big((1 + 2 + \ldots + 9)(0 + 1 + \ldots + 9)^* + 0\big).(0 + 1 + \ldots + 9)^+.$$

Espressioni regolari: problema

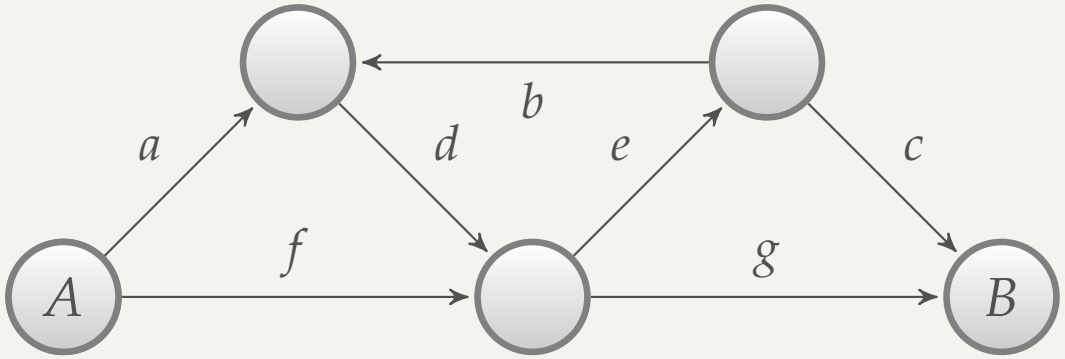
Determinare l'espressione regolare che, sull'alfabeto $\{a, b\}$, definisce l'insieme delle stringhe il cui terzultimo carattere è una b.

Espressioni regolari: problema

Determinare il linguaggio definito dall'espressione regolare $a^*\big((aa)^*b + (bb)^*a\big)b^*$.

Espressioni regolari: problema

Sia data la mappa stradale (con tratti stradali a senso unico e contrassegnati da caratteri dell'alfabeto) schematicamente indicata in figura. Fornire un'espressione regolare che definisca tutti i percorsi, anche passanti più volte per uno stesso nodo, tra A e B.



Equivalenza tra espressioni regolari

Due espressioni regolari r, s sono equivalenti ($r \equiv s$) se $L(r) = L(s)$.

Ad esempio, $a + b \equiv b + a$, $a + a \equiv a$, $aa^* \equiv a^*a$, $ab \not\equiv ba$.

Algebra delle espressioni regolari

Operatori

Assumiamo che $\cdot$ abbia precedenza su $+$. Quindi $a + b \cdot c \equiv a + (b \cdot c)$.

Inoltre, rappresentiamo l'operatore $\cdot$ con la concatenazione degli operandi: $ab \equiv a \cdot b$.

Proprietà di + e ·

1. $+$ è commutativa ($r + s \equiv s + r$), associativa ($r + (s + t) \equiv (r + s) + t$), con elemento neutro $\emptyset$ ($r + \emptyset \equiv r$), idempotente ($r + r \equiv r$)
2. $\cdot$ è associativa ($r(st) \equiv (rs)t$), con elemento neutro ε ($r\varepsilon \equiv r$) e elemento nullo $\emptyset$ ($r\emptyset \equiv \emptyset$)
3. $\cdot$ si distribuisce su $+$ ($r(s + t) \equiv rs + rt$)
4. $+$ <u>non</u> si distribuisce su $\cdot$ ($r + st \not\equiv (r + s)(r + t)$)

Alcune proprietà derivate di + e ·

1. $\emptyset^* \equiv \varepsilon^* = \varepsilon$
2. $r^* \equiv r^* r^* \equiv (r^*)^* \equiv r + r^*$
3. $r^* \equiv \varepsilon + r^* \equiv \varepsilon + rr^* \equiv (\varepsilon + r)^* \equiv (\varepsilon + r)r^*$
4. $r^* \equiv (r + r^2 + \cdots + r^k)^* \equiv \varepsilon + r + r^2 + \cdots + r^{k-1} + r^k r^*$ per ogni $k \geq 1$
5. $r^* r \equiv r r^*$
6. $(r + s)^* \equiv (r^* + s^*)^* \equiv (r^* s^*)^* \equiv (r^* s)^* r^* \equiv r^* (s r^*)^*$
7. $r(sr)^* \equiv (rs)^* r$
8. $(r^* s)^* \equiv \varepsilon + (r + s)^* s$
9. $(r s^*)^* \equiv \varepsilon + r(r + s)^*$

Esempio di dimostrazione di equivalenza

Dimostrazione che $(a + aa)(a + b)^* \equiv a(a + b)^*$

$$
\begin{aligned}
(a + aa)(a + b)^* &\equiv (a + aa)a^*(ba^*)^* \text{ in quanto } (r + s)^* \equiv r^*(sr^*)^* \\
&\equiv a(\varepsilon + a)a^*(ba^*)^* \text{ in quanto } r \equiv r\varepsilon \\
&\equiv aa^*(ba^*)^* \text{ in quanto } (\varepsilon + r)r^* \equiv r^* \\
&\equiv a(a + b)^* \text{ in quanto } (r + s)* \equiv r^*(sr^*)^*
\end{aligned}
$$

Esempio di dimostrazione di equivalenza

Dimostrazione che $a^*(b + ab^*) \equiv b + aa^*b^*$

$$
\begin{aligned}
a^*(b + ab^*) &\equiv (\varepsilon + aa^*)(b + ab^*) \text{ in quanto } r^* \equiv \varepsilon + rr^* \\
&\equiv b + ab^* + aa^*b + aa^*ab^* \text{ per distributività} \\
&\equiv b + (ab^* + aa^*ab^*) + aa^*b \text{ per associatività e commutatività di } + \\
&\equiv b + (\varepsilon + aa^*)ab^* + aa^*b \text{ in quanto } r \equiv r\varepsilon \\
&\equiv b + a^*ab^* + aa^*b \text{ in quanto } r^* \equiv \varepsilon + rr^* \\
&\equiv b + aa^*b^* + aa^*b \text{ in quanto } r^*r \equiv rr^* \\
&\equiv b + aa^*(b^* + b) \text{ per distributività} \\
&\equiv b + aa^*b^* \text{ in quanto } r^* \equiv r^* + r
\end{aligned}
$$

Esempio di dimostrazione di non equivalenza

Dimostrazione che $(a + b)^* \not\equiv a^* + b^*$.

Basta osservare che $ab \in L((a + b)^*) - L(a^* + b^*)$

Esempio di dimostrazione di non equivalenza

Dimostrazione che $(a + b)^* \not\equiv a^*b^*$.

Basta osservare che $ba \in L((a + b)^*) - L(a^*b^*)$

Esempio di semplificazione

Semplificazione dell'espressione regolare $aa(b^* + a) + a(ab^* + aa)$:

$$\begin{aligned} aa(b^* + a) + a(ab^* + aa) &\equiv aa(b^* + a) + aa(b^* + a) \text{ distributività} \\ &\equiv aa(b^* + a) \text{ in quanto } r + r \equiv r \end{aligned}$$

Grammatiche

Una grammatica formale $\mathcal{G}$ è una quadrupla $\mathcal{G} = \langle V_T, V_N, P, S \rangle$ in cui:

1. V_T è un insieme finito e non vuoto di simboli terminali
2. V_N è un insieme finito e non vuoto di simboli non terminali
3. P è una relazione binaria di cardinalità finita su

$$(V_T \cup V_N)^* \circ V_N \circ (V_T \cup V_N)^* \times (V_T \cup V_N)^*.$$

   P è detta insieme delle produzioni.
   Una coppia $\langle \alpha, \beta \rangle \in P$, si indica generalmente con la notazione $\alpha \longrightarrow \beta$;
4. $S \in V_N$ è detto assioma

Grammatiche

Si consideri la grammatica $\mathcal{G} = \langle \{a, b\}, \{S, B, C\}, P, S \rangle$, avente le seguenti regole di produzione

1. $S \longrightarrow aS$
2. $S \longrightarrow B$
3. $B \longrightarrow bB$
4. $B \longrightarrow bC$
5. $C \longrightarrow cC$
6. $C \longrightarrow c$.

Con questa grammatica si possono generare le stringhe del linguaggio

$$L(\mathcal{G}) = \{a^n b^m c^h \mid n \geq 0,\ m, h \geq 1\}.$$

Grammatiche

Un insieme di produzioni aventi stessa parte sinistra

$$\begin{aligned} \alpha &\longrightarrow \beta_1 \\ \alpha &\longrightarrow \beta_2 \\ &\dots \\ \alpha &\longrightarrow \beta_n, \end{aligned}$$

viene convenzionalmente indicato, in maniera più compatta, come

$$\alpha \longrightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n.$$

Inoltre, l'unione $V_T \cup V_N$ viene indicata con V.

ε-produzioni

Una regola del tipo $\alpha \longrightarrow \varepsilon$, dove $\alpha \in V^* \circ V_N \circ V^*$, prende il nome di ε-produzione o ε-regola.

Derivazioni dirette

Data una grammatica $\mathcal{G} = \langle V_T, V_N, P, S \rangle$, la derivazione diretta (rispetto a $\mathcal{G}$) è una relazione su $(V^* \circ V_N \circ V^*) \times V^*$ così definita:

la coppia $\langle \phi, \psi \rangle$ appartiene alla relazione se e solo se esistono $\alpha \in V^* \circ V_N \circ V^*$ e $\beta, \gamma, \delta \in V^*$ tali che

$$\phi = \gamma\alpha\delta$$
$$\psi = \gamma\beta\delta$$
$$\alpha \longrightarrow \beta \in P$$

In questo caso, scriviamo $\phi \underset{\mathcal{G}}{\Longrightarrow} \psi$

Derivazioni

Data una grammatica $\mathcal{G}$, una derivazione (in $\mathcal{G}$) è una sequenza di stringhe $\phi_1, \ldots, \phi_n \in V^*$ tali che

$$\forall i \in \{1, \ldots, n-1\} : \phi_i \underset{\mathcal{G}}{\Longrightarrow} \phi_{i+1}$$

La relazione di *derivabilità* (rispetto a $\mathcal{G}$) è la chiusura transitiva e riflessiva della derivazione diretta: essa si rappresenta con la notazione $\overset{*}{\underset{\mathcal{G}}{\Longrightarrow}}$.

Scrivendo $\phi \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} \psi$ indichiamo l'esistenza di (almeno) una derivazione da ϕ a ψ.

Forma di frase

Data una grammatica $\mathcal{G}$, si definisce forma di frase (in $\mathcal{G}$) una qualunque stringa $\phi \in V^*$ tale che $S \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} \phi$.

Linguaggio generato

Il linguaggio generato da una grammatica $\mathcal{G}$ è l'insieme $L(\mathcal{G}) \subseteq \Sigma^*$ tale che

$$L(\mathcal{G}) = \left\{ x \mid x \in V_T^* \wedge S \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x \right\}.$$

$L(\mathcal{G})$ è l'insieme delle stringhe di caratteri terminali che si possono ottenere a partire dall'assioma mediante l'applicazione di un numero finito di passi di derivazione diretta.

Esempio

Il linguaggio

$$L = \{a^n b^n c^n \mid n \geq 1\}$$

può essere generato dalla grammatica $\mathcal{G}$ in cui $V_T = \{a, b, c\}$ e $V_N = \{S, B, C, F, G\}$ e le regole di P sono le seguenti:

$$\begin{aligned} S &\longrightarrow aSBC \\ CB &\longrightarrow BC \\ SB &\longrightarrow bF \\ FB &\longrightarrow bF \\ FC &\longrightarrow cG \\ GC &\longrightarrow cG \\ G &\longrightarrow \varepsilon. \end{aligned}$$

Esempio

La stringa *aabbcc* si può generare con la derivazione:

$$
\begin{aligned}
S &\Longrightarrow aSBC \\
&\Longrightarrow aaSBCBC \\
&\Longrightarrow aaSBBCC \\
&\Longrightarrow aabFBCC \\
&\Longrightarrow aabbFCC \\
&\Longrightarrow aabbcGC \\
&\Longrightarrow aabbccG \\
&\Longrightarrow aabbcc.
\end{aligned}
$$

Quindi, $S \overset{*}{\Longrightarrow} aabbcc$, ed in particolare che $S \overset{i}{\Longrightarrow} aabbcc$ per ogni $i \geq 8$.

Esempio

Come dimostrare che $L(\mathcal{G}) = \{a^n b^n c^n \mid n \geq 1\}$?

- Dimostrare che ogni x del tipo $a^n b^n c^n$ è derivabile in $\mathcal{G}$
- Dimostrare che ogni $z \in V_T^* = \{a, b, c\}^*$ derivabile in $\mathcal{G}$ ha la forma $a^n b^n c^n$

Esempio

La grammatica $\mathcal{G} = \langle \{a, b\}, \{S, A\}, P, S \rangle$ con produzioni P

$$\begin{aligned} S &\longrightarrow Ab \\ A &\longrightarrow Sa \end{aligned}$$

genera il linguaggio vuoto Λ.

Esempio

Dimostriamo che la grammatica $\mathcal{G} = \langle \{a, b, c\}, \{S, A\}, P, S \rangle$ con produzioni P

$$\begin{aligned} S &\longrightarrow aSc \mid A \\ A &\longrightarrow bAc \mid \varepsilon, \end{aligned}$$

genera il linguaggio

$$L = \left\{ a^n b^m c^{n+m} \mid n, m \geq 0 \right\}.$$

Esempio

Proviamo che $\mathscr{L}(\mathscr{G}) \subseteq L$.

Chiaramente, tutte le forme di frase costruite da $\mathscr{G}$ sono del tipo

- $a^k S c^k$, $k \geq 0$, oppure
- $a^k b^j A c^{k+j}$, $k \geq 0$ e $j \geq 0$.

Quindi, ogni parola z generata da $\mathscr{G}$ è ottenuta tramite una derivazione del tipo

$$S \overset{k}{\Longrightarrow} a^k S c^k \overset{1}{\Longrightarrow} a^k A c^k \overset{j}{\Longrightarrow} a^k b^j A c^{k+j} \overset{1}{\Longrightarrow} a^k b^j c^{k+j} = z.$$

Ogni parola derivata dunque appartiene a L.

Esempio

Proviamo ora che $L \subseteq \mathcal{L}(\mathcal{G})$.

Basta osservare che ogni stringa $z \in L$ è del tipo $a^n b^m c^{n+m}$ e che essa viene generata da $\mathcal{G}$ attraverso la derivazione

$$S \overset{m}{\Longrightarrow} a^m S c^m \overset{1}{\Longrightarrow} a^m A c^m \overset{n}{\Longrightarrow} a^m b^n A c^{m+n} \overset{1}{\Longrightarrow} a^m b^n c^{m+n} = z.$$

Esiste dunque in $\mathcal{G}$ una derivazione che costruisce z.

Esercizio

Data la grammatica $\mathcal{G} = \langle \{a\}, \{S, I, F, M\}, P, S \rangle$ con produzioni P

$$\begin{aligned} S &\longrightarrow a \mid aa \mid IaF \\ aF &\longrightarrow Maa \mid MaaF \\ aM &\longrightarrow Maa \\ IM &\longrightarrow Ia \mid aa \end{aligned}$$

dimostrare che $\mathcal{G}$ genera il linguaggio $\left\{a^{2^n} \mid n \geq 0\right\}$.

Esempi di derivazione

$S \Longrightarrow a$

$S \Longrightarrow aa$

$S \Longrightarrow IaF \Longrightarrow IMaa \Longrightarrow aaaa$

$S \Longrightarrow IaF \Longrightarrow IMaaF \Longrightarrow IaaaF \Longrightarrow IaaMaa \Longrightarrow IaMaaaa \Longrightarrow IMaaaaaa \Longrightarrow aaaaaaaa$

Esercizio

Definire una grammatica che generi il linguaggio

$$L = \{a^n b^m c^p \mid n, p \geq 0, m \geq 1; n = m \vee m = p\}$$

e dimostrare la sua correttezza.

Equivalenza tra grammatiche

Due grammatiche $\mathcal{G}_1$ e $\mathcal{G}_2$ si dicono equivalenti se generano lo stesso linguaggio, vale a dire se $L(\mathcal{G}_1) = L(\mathcal{G}_2)$.

Esercizio

Dimostrare che la grammatica con produzioni

$$S \longrightarrow aS \mid b$$

e la grammatica con produzioni

$$\begin{aligned} S &\longrightarrow b \mid Ab \\ A &\longrightarrow Aa \mid a \end{aligned}$$

sono equivalenti.

Grammatiche di tipo 0

Dette anche non limitate, definiscono la classe di linguaggi più ampia possibile.

In esse le produzioni sono del tipo più generale:

$$\alpha \longrightarrow \beta,\ \alpha \in V^* \circ V_N \circ V^*,\ \beta \in V^*.$$

Queste grammatiche ammettono anche derivazioni che "accorciano" le forme di frase, come ad esempio quelle che si ottengono applicando le ε-produzioni.

I linguaggi generabili da grammatiche di tipo 0 si dicono linguaggi di tipo 0.

Esempio

La grammatica

$$\mathcal{G} = \langle \{a, b\}, \{S, A\}, P, S \rangle,$$

in cui P è

$$\begin{aligned} S &\longrightarrow aAb \\ aA &\longrightarrow aaAb \\ A &\longrightarrow \varepsilon, \end{aligned}$$

è di tipo 0 e genera il linguaggio $L = \{a^n b^n \mid n \geq 1\}$.

Grammatiche di tipo 1

Dette anche contestuali o context sensitive (CS), ammettono qualunque regola di produzione che non riduca la lunghezza delle stringhe, cioè tutte le produzioni del tipo:

$$\alpha \longrightarrow \gamma, \alpha \in V^* \circ V_N \circ V^*, \gamma \in V^+, |\alpha| \leq |\gamma| \ .$$

I linguaggi generabili da grammatiche di tipo 1 si dicono linguaggi di tipo 1, contestuali, o context sensitive (CS).

Esempio

Le produzioni

$$\begin{aligned} S &\longrightarrow aSa \mid aAb \mid aAa \\ aA &\longrightarrow aa \\ Ab &\longrightarrow aab \end{aligned}$$

appartengono ad una grammatica di tipo 1.

Esempio

Il linguaggio $\{a^n b^n \mid n \geq 1\}$ può venir generato da una grammatica di tipo 1 avente le produzioni

$$\begin{aligned} S &\longrightarrow aBS \mid ab \\ Ba &\longrightarrow aB \\ Bb &\longrightarrow bb. \end{aligned}$$

Dunque il linguaggio $\{a^n b^n \mid n \geq 1\}$ è contestuale.

Definizione alternativa

Il termine “linguaggio contestuale” deriva dal fatto che sono generabili da grammatiche aventi produzioni “contestuali” del tipo

$$\beta_1 A \beta_2 \longrightarrow \beta_1 \gamma \beta_2,\ A \in V_N,\ \beta_1, \beta_2 \in V^*,\ \gamma \in V^+,$$

in cui si esprime il fatto che A può essere sostituito da γ in una forma di frase solo se si trova nel contesto $\langle \beta_1, \beta_2 \rangle$.

Grammatiche di tipo 2

Dette anche non contestuali o context free (CF), ammettono solo produzioni del tipo:

$$A \longrightarrow \beta, \quad A \in V_N, \, \beta \in V^+$$

cioè produzioni in cui ogni non terminale A può essere riscritto in una stringa β indipendentemente dal contesto in cui esso si trova.

esempio

Grammatica che genera espressioni aritmetiche di somme e moltiplicazioni in una variabile i:

$$\begin{aligned} E &\longrightarrow E + T \mid T \\ T &\longrightarrow T * F \mid F \\ F &\longrightarrow i \mid (E). \end{aligned}$$

Generare la stringa $(i + i) * i$ utilizzando la grammatica dell'esempio precedente.

Esempio

Il linguaggio $\{a^n b^n \mid n \geq 1\}$ può essere generato dalla grammatica di tipo 2 con produzioni

$$S \longrightarrow aSb \mid ab.$$

Dunque, il linguaggio $\{a^n b^n \mid n \geq 1\}$ è context free.

Esempio

Il linguaggio delle parentesi ben bilanciate è di tipo 2. Esso può essere generato dalla grammatica $\mathcal{G} = \langle \{(,)\}, \{S\}, P, S \rangle$, dove P è l'insieme delle produzioni

$$S \quad \longrightarrow \quad () \mid SS \mid (S).$$

Esercizio

Definire una grammatica non contestuale per generare tutte le stringhe palindrome, cioè quelle che risultano uguali se lette da destra verso sinistra o da sinistra verso destra.

Grammatiche di tipo 3

Dette anche lineari destre o regolari, ammettono solo produzioni del tipo:

$$A \longrightarrow \delta, \quad A \in V_N, \delta \in (V_T \circ V_N) \cup V_T.$$

Il termine "regolare" deriva dal fatto che i corrispondenti linguaggi sono rappresentabili per mezzo di espressioni regolari.

I linguaggi generabili da grammatiche di tipo 3 vengono detti linguaggi di tipo 3 o regolari.

Esempio

La grammatica

$$\mathcal{G} = \langle \{a, b\}, \{S\}, P, S \rangle$$

in cui P contiene le produzioni

$$S \longrightarrow aS \mid b$$

è una grammatica di tipo 3 e genera il linguaggio regolare $L = \{a^n b \mid n \geq 0\}$.

Esercizio

Definire una grammatica regolare per il linguaggio delle stringhe sull'alfabeto $\{a, b\}$ che contengono un numero dispari di a.

Grammatiche lineari sinistre

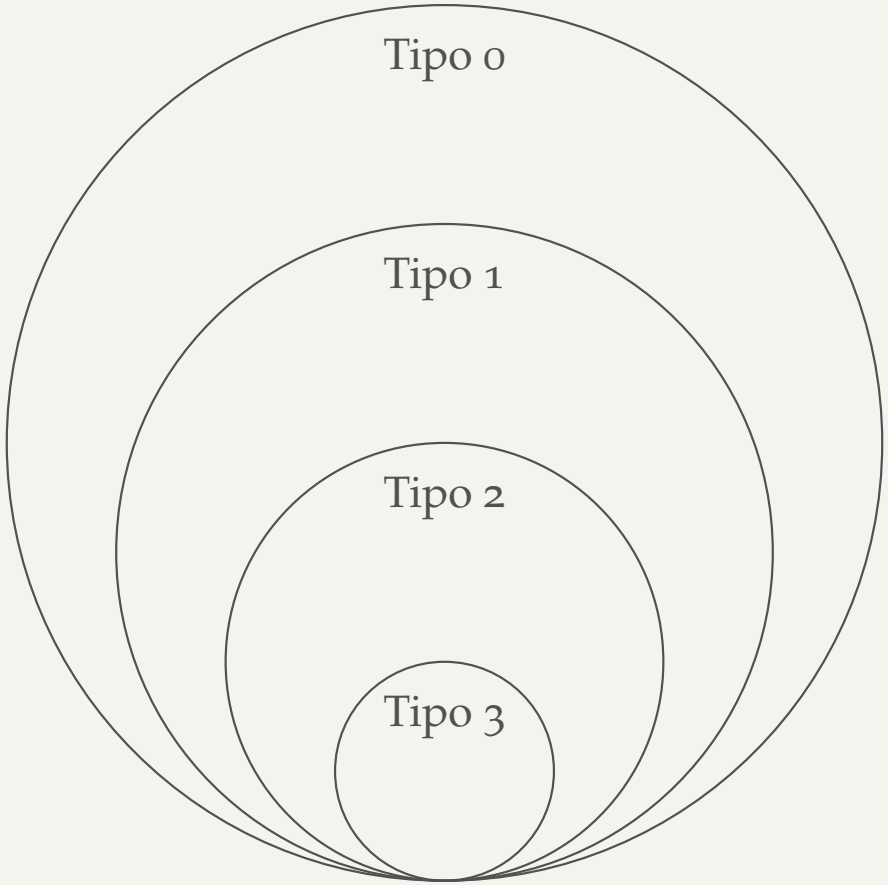
Ilinguaggi regolari si possono definire anche mediante grammatiche lineari sinistre caratterizzate da regole del tipo:

$$A \longrightarrow \delta, \;\; A \in V_N, \delta \in (V_N \circ V_T) \cup V_T.$$

Le grammatiche lineari sinistre differiscono dalle grammatiche lineari destre in quanto l'unico simbolo non terminale che può comparire nella parte destra δ della produzione appare come primo simbolo.

Gerarchia di Chomsky

Per $0 \leq n \leq 2$, ogni grammatica di tipo $n+1$ è anche di tipo n: pertanto l'insieme dei linguaggi di tipo n contiene tutti i linguaggi di tipo $n+1$, formando quindi una gerarchia, detta Gerarchia di Chomsky



Gerarchia di Chomsky

Un linguaggio L viene detto strettamente di tipo n se esiste una grammatica $\mathcal{G}$ di tipo n che genera L e non esiste alcuna grammatica $\mathcal{G}'$ di tipo $m > n$ che possa generarlo.

Gerarchia di Chomsky

TIPO	PRODUZIONI	DOVE
TIPO 0	$\alpha \longrightarrow \beta$	$\alpha \in V^* \circ V_N \circ V^*,\ \beta \in V^*$
TIPO 1	$\alpha \longrightarrow \beta$	$\lvert\ \alpha\ \rvert \leq \lvert\ \beta\ \rvert,$ $\alpha \in V^* \circ V_N \circ V^*,\ \beta \in V^+$
TIPO 2	$A \longrightarrow \beta$	$A \in V_N,\ \beta \in V^+$
TIPO 3	$A \longrightarrow \delta$	$A \in V_N,\ \delta \in (V_T \cup (V_T \circ V_N))$

Grammatiche con ε-produzioni

Dalle definizioni, non è possibile generare la stringa vuota con grammatiche di tipo 1, 2 o 3.

Linguaggi contenenti ε possono però essere generati apportando lievi modifiche (aggiunta di opportune ε-produzioni) alle grammatiche non contestuali e regolari.

Grammatiche con ε-produzioni

Se una grammatica $\mathcal{G} = \langle V_T, V_N, P, S \rangle$ di tipo 1, 2 o 3 genera un linguaggio L, per poter generare il linguaggio $L \cup \{\varepsilon\}$ è sufficiente utilizzare la grammatica

$$\mathcal{G}' = \langle V_T, V_N \cup \{S'\}, P', S' \rangle,$$

dove

$$P' = P \cup \{S' \longrightarrow \varepsilon\} \cup \{S' \longrightarrow S\}.$$

Grammatiche con ε-produzioni

Data la grammatica CF $\mathcal{G}$, con assioma S e produzioni:

$$\begin{aligned} S &\longrightarrow aBSc \mid abc \\ Ba &\longrightarrow aB \\ Bb &\longrightarrow bb. \end{aligned}$$

risulta $L(\mathcal{G}) = \{a^n b^n c^n \mid n \geq 1\}$.

Il linguaggio $L(\mathcal{G}) \cup \{\varepsilon\} = \{a^n b^n c^n \mid n \geq 0\}$ è derivato dalla la grammatica $\mathcal{G}'$ con assioma S' e produzioni :

$$\begin{aligned} S' &\longrightarrow S \mid \varepsilon \\ S &\longrightarrow aBSc \mid abc \\ Ba &\longrightarrow aB \\ Bb &\longrightarrow bb \end{aligned}$$

Grammatiche con ε-produzioni

Aggiungere la ε-produzione direttamente come produzione dell'assioma può avere effetti indesiderati

$$\begin{aligned} S &\longrightarrow Ub \\ U &\longrightarrow ab \mid S \end{aligned}$$

genera ab^*bb. Ma

$$\begin{aligned} S &\longrightarrow Ub \mid \varepsilon \\ U &\longrightarrow ab \mid S \end{aligned}$$

genera $ab^*bb \cup \{\varepsilon\} \cup b^*b$

Grammatiche di tipo 1 e ε-produzioni

L'aggiunta non controllata di ε-produzioni può aumentare in modo sostanziale il potere generativo della grammatica.

Teorema

Data una grammatica $\mathcal{G} = \langle V_T, V_N, P, S \rangle$ di tipo 0, esiste una grammatica $\mathcal{G}'$ equivalente a $\mathcal{G}$, ottenuta estendendo una grammatica di tipo 1 con opportune ε-produzioni.

Grammatiche di tipo 1 e ε-produzioni

La grammatica $\mathcal{G}' = \langle V'_T, V'_N, P', S' \rangle$ è caratterizzata da: $V'_T = V_T$, $V'_N = V_N \cup \{X\}$, con $X \notin V_N$, $S' = S$ e P' ottenuto da P aggiungendo la produzione $X \longrightarrow \varepsilon$ e sostituendo ad ogni produzione $\phi \longrightarrow \psi$ con $|\phi| > |\psi|$ o la produzione

$$\phi \longrightarrow \psi \underbrace{X \ldots X}_{|\phi|-|\psi|}$$

È semplice verificare che con la grammatica $\mathcal{G}'$ sono derivabili tutte e sole le stringhe di V_T^* derivabili con la grammatica $\mathcal{G}$.

Grammatiche di tipo 2 e 3 e ε-produzioni

L'aggiunta indiscriminata di ε-produzioni non altera il potere generativo delle grammatiche.

Si può dimostrare che data una grammatica di tipo 2 o 3 estesa con ε-produzioni, ne possiamo sempre costruire una equivalente, dello stesso tipo, che usa ε-produzioni solo a partire dall'assioma (nel caso che ε appartenga al linguaggio da generare) o non ne usa affatto (in caso contrario).

Grammatiche di tipo 2 e 3 e ε-produzioni

Teorema

Data una grammatica $\mathcal{G} = \langle V_T, V_N, P, S \rangle$ il cui insieme di produzioni P comprende soltanto produzioni di tipo non contestuale e produzioni vuote, esiste una grammatica non contestuale $\mathcal{G}'$ tale che $L(\mathcal{G}') = L(\mathcal{G}) - \{\varepsilon\}$

Grammatiche di tipo 2 e 3 e ε-produzioni

Derivazione di $\mathcal{G}'$ da $\mathcal{G}$.

1. determinazione dei simboli che si annullano, cioè i non terminali da cui è possibile derivare ε
2. per ogni produzione $A \longrightarrow \alpha$ di P, con l'esclusione delle ε-produzioni, se nessun simbolo di α si annulla $A \longrightarrow \alpha$ va in P'; altrimenti a P' si aggiungono tutte le possibili produzioni ottenute da $A \longrightarrow \alpha$ eliminando da α un sottoinsieme dei simboli che si annullano, considerati con la propria molteplicità.

Grammatiche di tipo 2 e 3 e ε-produzioni

Per provare che $L(\mathcal{G}') = L(\mathcal{G}) - \{\varepsilon\}$ basta mostrare, per induzione sulla lunghezza della derivazione, che $\forall A \in V_N$ e $\forall w \in V_T^+$:

$$A \overset{*}{\underset{\mathcal{G}'}{\Longrightarrow}} w \text{ se e solo se } (w \neq \varepsilon \ \wedge \ A \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} w).$$

Nel caso particolare in cui la grammatica è regolare è possibile dimostrare un risultato analogo.

Forma normale di Backus

Le grammatiche formali sono utilizzate per la definizione di linguaggi di programmazione, caratterizzati come l'insieme di tutte le stringhe (programmi) derivabili dalla grammatica.

Per la definizione sintattica dei linguaggi di programmazione vengono adottate grammatiche non contestuali, tradizionalmente rappresentate mediante una notazione specifica, particolarmente suggestiva, denominata Forma Normale di Backus (Backus Normal Form, BNF, detta anche Backus-Naur Form).

Forma normale di Backus

Notazione per grammatiche context free resa più espressiva e succinta:

1. I simboli non terminali costituiti da stringhe (ad es.: `<espressione>`);
2. $\longrightarrow$ sostituito da ::=
3. Parentesi graffe $\{\ldots\}$ impiegate per indicare l'iterazione illimitata. Con $\{\ldots\}^n$ si indica l'iterazione per un numero di volte pari al più ad n.
4. Parentesi quadre $[\ldots]$ utilizzate per indicare l'opzionalità (possibile assenza di una parte di stringa)
5. Parentesi tonde $(\ldots)$ utilizzate per indicare la fattorizzazione, vale a dire la messa in comune di una sottoespressione

Esempio

Identificatore:

$$
\begin{aligned}
\texttt{<id>} &::= \texttt{<alfa>}\ \{\texttt{<alfanum>}\}^{n-1} \\
\texttt{<alfanum>} &::= \texttt{<alfa>} \mid \texttt{<cifra>} \\
\texttt{<alfa>} &::= \mathbf{A} \mid \mathbf{B} \mid \cdots \mid \mathbf{Z} \mid \mathbf{a} \mid \mathbf{b} \mid \cdots \mid \mathbf{z} \\
\texttt{<cifra>} &::= \mathbf{0} \mid \mathbf{1} \mid \cdots \mid \mathbf{9}
\end{aligned}
$$