Linguaggi regolari

Corso di Fondamenti di Informatica - modulo I

Giorgio Gambosi

a.a. 2023-2024

Equivalenza tra ASF, RG e RE

ASF e grammatiche di tipo 3

Per ogni grammatica regolare $G = \langle V_T, V_N, P, S \rangle$, esiste un ASFND $A_N = \langle \Sigma, Q, \delta_N, q_0, F \rangle$ che riconosce il linguaggio che essa genera.

Viceversa, per ogni ASFND A_N esiste una grammatica regolare che genera il linguaggio che esso riconosce.

ASF e grammatiche di tipo 3

Sia $G = \langle V_T, V_N, P, S \rangle$ una grammatica di tipo 3, con al più la sola ε-produzione $S \longrightarrow \varepsilon$.

Definiamo una procedura che partire da G produca un ASFND $A_N = \langle \Sigma, Q, \delta_N, q_0, F \rangle$ equivalente (che accetta tutte e sole stringhe prodotte da G).

Da G a A_N

$$\Sigma = V_T$$

$$Q = \{q_I \mid I \in V_N\} \cup \{q_F\}$$

$$q_0 = q_S$$

$$F = \begin{cases} \{q_0, q_F\} & \text{se } S \longrightarrow \varepsilon \in P \\ \{q_F\} & \text{altrimenti} \end{cases}$$

Per ogni coppia $a \in V_T$ e $B \in V_N$,

$$\delta_N(q_B, a) = \begin{cases} \{q_C \mid B \longrightarrow aC \in P\} \cup \{q_F\} & \text{se } B \longrightarrow a \in P \\ \{q_C \mid B \longrightarrow aC \in P\} & \text{altrimenti.} \end{cases}$$

L'automa è, in generale, non deterministico.

Da G a A_N. Equivalenza di G e A_N

Per dimostrare l'equivalenza tra G e A_N, dobbiamo mostrare che per ogni $x \in \Sigma^*$ si ha che

$$S \underset{G}{\overset{*}{\Longrightarrow}} x \qquad \text{se e solo se} \qquad \overline{\delta}_N(q_S, x) \cap F \neq \emptyset$$

Questo è chiaramente vero se $x = \varepsilon$, in quanto $\overline{\delta}_N(q_0, \varepsilon) = q_0 \in F$, se e solo se $S \longrightarrow \varepsilon \in P$, per costruzione.

Nel caso $x \in \Sigma^+$ mostriamo, per induzione sulla lunghezza di x, la proprietà più generale

$$S \overset{*}{\Longrightarrow} xZ \qquad \text{se e solo se} \qquad q_Z \in \overline{\delta}_N(q_S, x)$$

Iniziamo da

$$S \stackrel{*}{\Longrightarrow} xZ \qquad \text{implica} \qquad q_Z \in \overline{\delta}_N(q_S, x)$$

Passo base: $| \, x \, | = 1$, per cui $x = a$, con $a \in \Sigma$. Allora abbiamo che $S \Longrightarrow aZ$ se e solo se $S \longrightarrow aZ \in P$ e quindi se e solo se, per costruzione dell'automa, $q_Z \in \delta_N(q_S, a)$.

Passo induttivo: $\mid x \mid > 1$, per cui $x = ya$, con $\mid y \mid = n \geq 1$ e $a \in \Sigma$.

Per l'ipotesi induttiva il risultato si assume valido per y, quindi

$$S \overset{*}{\Longrightarrow} yZ \qquad \text{se e solo se} \qquad q_Z \in \overline{\delta}_N(q_S, y)$$

Osserviamo che $S \overset{*}{\Longrightarrow} xZ'$ se e solo se esiste $Z \in V_N$ tale che $S \overset{*}{\Longrightarrow} yZ \Longrightarrow yaZ' = xZ'$. Ne deriva che

- $q_Z \in \overline{\delta}_N(q_S, y)$ per induzione
- $Z \longrightarrow aZ' \in P$, e quindi $q_{Z'} \in \delta_N(a, Z)$ per costruzione

Quindi, $q_{Z'} \in \overline{\delta}_N(q_S, ya) = \overline{\delta}_N(q_S, x)$

Da G a A_N. Equivalenza di G e A_N

Abbiamo verificato che $S \stackrel{*}{\Longrightarrow} xZ$ se e solo se $q_Z \in \overline{\delta}_N(q_S, x)$.

Osserviamo ora che $S \stackrel{*}{\Longrightarrow} x$ se e solo se esistono $Z \in V_N$, $y \in \Sigma^*$ e $Z \longrightarrow a \in P$ tali che $x = ya$ e $S \stackrel{*}{\Longrightarrow} yZ \stackrel{*}{\Longrightarrow} ya = x$.

Da quanto visto sopra, ciò è vero se e solo se $q_Z \in \overline{\delta}_N(q_S, y)$ e $q_F \in \delta_N(q_Z, a)$, e quindi se e solo se $q_F \in \overline{\delta}_N(q_S, ya) = \overline{\delta}_N(q_S, x)$.

In conclusione, per ogni linguaggio regolare (generato da una grammatica di tipo 3) esiste un ASFND che lo accetta (e quindi anche un ASFD che lo decide).

ASF e grammatiche di tipo 3

Sia $A = \langle \Sigma, Q, \delta, q_0, F \rangle$ un ASFD.

Definiamo una procedura che partire da A produca una grammatica di tipo 3 $G = \langle V_T, V_N, P, S \rangle$ equivalente (che genera tutte e sole stringhe accettate da A).

Se $q_0 \notin F$:

$V_T = \Sigma$

$V_N = \{A_i \mid \text{ per ogni } q_i \in Q\}$

$S = A_0$

per ogni regola di transizione $\delta(q_i, a) = q_j$ esiste $A_i \longrightarrow aA_j \in P$, e se $q_j \in F$ esiste anche $A_i \longrightarrow a \in P$

Se $q_0 \in F$:

$V_T = \Sigma$

$V_N = \{A_i \mid \text{ per ogni } q_i \in Q\} \cup \{A'_0\}$

$S = A'_0$

per ogni regola di transizione $\delta(q_i, a) = q_j$ esiste $A_i \longrightarrow aA_j \in P$, e se $q_j \in F$ esiste anche $A_i \longrightarrow a \in P$ (tutte le precedenti). Inoltre, per ogni $\delta(q_0, a) = q_j$ esiste $A'_0 \longrightarrow aA_j \in P$, e se $q_j \in F$ esiste anche $A'_0 \longrightarrow a \in P$ (A'_0 ha tutte le produzioni di A_0), infine, esiste $A'_0 \longrightarrow \varepsilon \in P$.

Come prima, per dimostrare l'equivalenza tra G e A_N, dobbiamo mostrare che per ogni $x \in \Sigma^*$ si ha che

$$\overline{\delta}(q_0, x) \in F \qquad \text{se e solo se} \qquad S \underset{G}{\overset{*}{\Longrightarrow}} x$$

Questo è chiaramente vero se $x = \varepsilon$, in quanto in tal caso necessariamente $q_0 \in F$ e, per costruzione, l'assioma di G è A_0' e $A_0' \longrightarrow \varepsilon \in P$.

Nel caso $x \in \Sigma^+$ mostriamo, per induzione sulla lunghezza di x, entrambe le proprietà

$$A_i \overset{*}{\Longrightarrow} xA_j \qquad \text{se e solo se} \qquad \overline{\delta}(q_i, x) = q_j$$

$$A_i \overset{*}{\Longrightarrow} x \qquad \text{se e solo se} \qquad \overline{\delta}(q_i, x) \in F$$

Passo base: $\mid x \mid = 1$, ad esempio $x = a$. Abbiamo allora che

Per costruzione, $A_i \longrightarrow aA_j \in P$ (e quindi $A_i \Longrightarrow aA_j$) se e solo se $\delta(q_i, a) = q_j$ (e quindi $\overline{\delta}(q_i, a) = q_j$)

e inoltre che, per costruzione,

$A_i \longrightarrow a \in P$ (e quindi $A_i \Longrightarrow a$) se e solo se $q_j \in F$

Passo induttivo: $| x | = n > 1$.

Sia $x = ya$, con $|y| = n - 1$: per l'ipotesi induttiva, la proprietà è valida per y, e quindi

$$A_i \stackrel{*}{\Longrightarrow} yA_k \qquad \text{se e solo se} \qquad \overline{\delta}(q_i, y) = q_k$$

Supponiamo $A_i \stackrel{*}{\Longrightarrow} xA_j = yaA_j$: ciò è possibile se e solo se esiste A_k tale che $A_i \stackrel{*}{\Longrightarrow} yA_k$ e $A_k \longrightarrow aA_j \in P$

Da A a G. Equivalenza di G e A

Per l'ipotesi induttiva, $A_i \overset{*}{\Longrightarrow} yA_k$ se e solo se $\overline{\delta}(q_i, y) = q_k$.

Per costruzione, $A_k \longrightarrow aA_j \in P$ se e solo se $\delta(q_k, a) = q_j$.

Ne consegue che

$$A_i \overset{*}{\Longrightarrow} yA_k \Longrightarrow yaA_j = xA_j$$

se e solo se

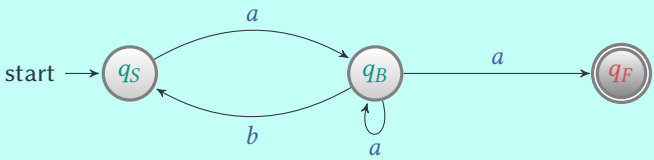
$$q_j = \delta(q_k, a) = \delta(\overline{\delta}(q_i, y), a) = \overline{\delta}(q_i, ya) = \overline{\delta}(q_i, x)$$

Esempio

Il linguaggio rappresentato da $a(a + ba)^*a$ è generato dalla grammatica

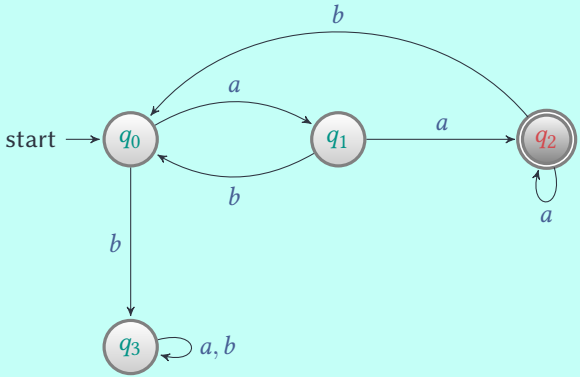
$$\begin{aligned} S &\longrightarrow aB \\ B &\longrightarrow aB \mid bS \mid a. \end{aligned}$$

ed è riconosciuto dall'ASFND



Esempio

A partire dall'ASFND è possibile derivare un ASFD equivalente



Esempio

E da questo una grammatica di tipo 3 equivalente a quella iniziale, dove $S = A_0$

$$\begin{aligned} A_0 &\longrightarrow aA_1 \\ A_1 &\longrightarrow bA_0 \mid aA_2 \mid a \\ A_2 &\longrightarrow aA_2 \mid bA_0 \mid a \end{aligned}$$

Per costruzione, questa grammatica ha, per ogni coppia $X \in V_N$ e $c \in V_T$, al più un $Y \in V_N$ tale che $X \longrightarrow cY \in P$.

Esercizio

Si consideri la grammatica regolare avente le seguenti produzioni:

$$
\begin{aligned}
S &\longrightarrow 0A \mid 1B \mid 0S \\
A &\longrightarrow aB \mid bA \mid a \\
B &\longrightarrow bA \mid aB \mid b.
\end{aligned}
$$

Si derivino un ASFND e un ASFD che riconoscono il linguaggio generato da tale grammatica. A partire dall'automa deterministico, derivare poi una grammatica di tipo 3 equivalente.

Teorema

Espressioni regolari e grammatiche di tipo 3

Teorema

$\mathscr{G}$ r $L(\mathscr{G}) = L(r)$

$\mathscr{G}$

Espressioni regolari e grammatiche di tipo 3

Consideriamo una grammatica $\mathcal{G}$ di tipo 3 ed il linguaggio L da essa generato, che per semplicità assumiamo non contenga la stringa vuota ε.

Se così non fosse, applichiamo le considerazioni seguenti al linguaggio $L - \{\varepsilon\}$, anch'esso regolare: una volta derivata un'espressione regolare r che lo definisce, l'espressione regolare che definisce L sarà chiaramente $r + \varepsilon$.

Espressioni regolari e grammatiche di tipo 3

Alla grammatica $\mathscr{G}$ possiamo far corrispondere un sistema di equazioni su espressioni regolari.

Estensione del linguaggio delle espressioni regolari con variabili $A, \ldots, Z$, associando una variabile ad ogni non terminale in $\mathscr{G}$.

Tali variabili potranno assumere valori nell'insieme delle espressioni regolari.

Espressioni regolari e grammatiche di tipo 3

Raggruppamento di tutte le produzioni che presentano a sinistra lo stesso non terminale. Per ogni produzione del tipo

$$A \longrightarrow a_1 B_1 \mid a_2 B_2 \mid \ldots \mid a_n B_n \mid b_1 \mid \ldots \mid b_m$$

equazione del tipo

$$A = a_1 B_1 + a_2 B_2 + \ldots + a_n B_n + b_1 + \ldots + b_m.$$

Espressioni regolari e grammatiche di tipo 3

Da una grammatica regolare si ottiene un sistema di *equazioni lineari destre*, in cui ogni monomio contiene una variabile a destra di simboli terminali.

Espressioni regolari e grammatiche di tipo 3

Risoluzione del sistema di equazioni su espressioni regolari estese:

individuazione dei valori (espressioni regolari normali, prive delle variabili che definiscono a loro volta espressioni regolari) che, una volta sostituiti alle variabili, soddisfano il sistema di equazioni.

Espressioni regolari e grammatiche di tipo 3

$$\begin{aligned} A &\longrightarrow aA \mid bB \\ B &\longrightarrow bB \mid c \end{aligned}$$

corrisponde al sistema di equazioni

$$\begin{cases} A &= aA + bB \\ B &= bB + c. \end{cases}$$

Espressioni regolari e grammatiche di tipo 3

Per risolvere il sistema è possibile utilizzare, le trasformazioni algebriche applicabili sulle operazioni di unione e concatenazione (distributività, fattorizzazione, ecc.), oltre alle seguenti due regole.

Sostituzione di una variabile con un'espressione regolare estesa.

Con riferimento all'esempio precedente abbiamo

$$\begin{cases} A &= aA + b(bB + c) = aA + bbB + bc \\ B &= bB + c. \end{cases}$$

Eliminazione della ricursione.

L'equazione $B = bB + c$ si risolve in $B = b^*c$. Infatti, sostituendo a destra e sinistra abbiamo

$$b^*c = b(b^*c) + c = b^+c + c = (b^+ + \varepsilon)c = b^*c.$$

Più in generale abbiamo che un'equazione del tipo

$$A = \alpha_1 A + \alpha_2 A + \ldots + \alpha_n A + \beta_1 + \beta_2 + \ldots + \beta_m$$

si risolve in

$$A = (\alpha_1 + \alpha_2 + \ldots + \alpha_n)^*(\beta_1 + \beta_2 + \ldots + \beta_m),$$

dove $\alpha_1, \ldots, \alpha_n$, β_1, ..., β_m sono espressioni regolari estese.

Grammatica regolare

$$
\begin{aligned}
A_0 &\longrightarrow aA_1 \mid a \\
A_1 &\longrightarrow bA_3 \mid bA_2 \\
A_2 &\longrightarrow aA_2 \mid bA_0 \mid b \\
A_3 &\longrightarrow bA_3 \mid aA_2.
\end{aligned}
$$

da cui si ottiene il seguente sistema lineare. $\begin{cases} A_0 = aA_1 + a \\ A_1 = bA_3 + bA_2 \\ A_2 = aA_2 + bA_0 + b \\ A_3 = bA_3 + aA_2 \end{cases}$

per eliminazione della ricursione su A_3:

$$\begin{cases} A_0 &= aA_1 + a \\ A_1 &= bA_3 + bA_2 \\ A_2 &= aA_2 + bA_0 + b \\ A_3 &= b^*aA_2 \end{cases}$$

per eliminazione della ricursione su A_2:

$$\begin{cases} A_0 &= aA_1 + a \\ A_1 &= bA_3 + bA_2 \\ A_2 &= a^*(bA_0 + b) \\ A_3 &= b^*aA_2 \end{cases}$$

per sostituzione di A_2 nell'equazione relativa ad A_3

Espressioni regolari e grammatiche di tipo 3

per sostituzione di A_2 nell'equazione relativa ad A_3 $\begin{cases} A_0 &= aA_1 + a \\ A_1 &= bA_3 + bA_2 \\ A_2 &= a^*(bA_0 + b) \\ A_3 &= b^*aa^*(bA_0 + b) \end{cases}$

per sostituzione di A_2 e A_3 nell'equazione relativa ad A_1

$$\begin{cases} A_0 &= aA_1 + a \\ A_1 &= b(b^*aa^*(bA_0 + b)) + b(a^*(bA_0 + b)) \\ A_2 &= a^*(bA_0 + b) \\ A_3 &= b^*aa^*(bA_0 + b) \end{cases}$$

per sostituzione di A_2 e A_3 nell'equazione relativa ad A_1.

per fattorizzazione nell'equazione relativa ad A_1:

$$\begin{cases} A_0 &= aA_1 + a \\ A_1 &= b(b^*aa^* + a^*)(bA_0 + b) \\ A_2 &= a^*(bA_0 + b) \\ A_3 &= b^*aa^*(bA_0 + b) \end{cases}$$

per sostituzione di A_1 nell'equazione relativa ad A_0:

$$\begin{cases} A_0 &= a(b(b^*aa^* + a^*)(bA_0 + b)) + a \\ A_1 &= b(b^*aa^* + a^*)(bA_0 + b) \\ A_2 &= a^*(bA_0 + b) \\ A_3 &= b^*aa^*(bA_0 + b) \end{cases}$$

Espressioni regolari e grammatiche di tipo 3

per fattorizzazione nell'equazione relativa ad A_0:

$$\begin{cases} A_0 &= ab(b^*aa^* + a^*)bA_0 + ab(b^*aa^* + a^*)b + a \\ A_1 &= b(b^*aa^* + a^*)(bA_0 + b) \\ A_2 &= a^*(bA_0 + b) \\ A_3 &= b^*aa^*(bA_0 + b) \end{cases}$$

Espressioni regolari e grammatiche di tipo 3

$$
\begin{aligned}
S &\longrightarrow aS \mid bA \mid \varepsilon \\
A &\longrightarrow aA \mid bS \mid \varepsilon
\end{aligned}
$$

Eliminazione della produzione $A \longrightarrow \varepsilon$:

$$\begin{aligned} S &\longrightarrow aS \mid bA \mid \varepsilon \mid b \\ A &\longrightarrow aA \mid bS \mid a. \end{aligned}$$

Espressioni regolari e grammatiche di tipo 3

$$
\begin{aligned}
S &= aS + bA + b + \varepsilon \\
A &= aA + bS + a
\end{aligned}
$$

Espressioni regolari e grammatiche di tipo 3

$$
\begin{aligned}
S &= aS + bA + b + \varepsilon \\
A &= a^*(bS + a)
\end{aligned}
$$

Espressioni regolari e grammatiche di tipo 3

$$
\begin{aligned}
S &= aS + ba^*(bS + a) + b + \varepsilon \\
A &= a^*(bS + a)
\end{aligned}
$$

Espressioni regolari e grammatiche di tipo 3

$$
\begin{aligned}
S &= (a + ba^*b)S + ba^*a + b + \varepsilon \\
A &= a^*(bS + a)
\end{aligned}
$$

Espressioni regolari e grammatiche di tipo 3

$$
\begin{aligned}
S &= (a + ba^*b)^*(ba^*a + b + \varepsilon) \\
A &= a^*(bS + a)
\end{aligned}
$$

Esercizio

Si consideri la seguente grammatica:

$$
\begin{aligned}
A &\longrightarrow aB \mid bC \mid a \\
B &\longrightarrow aA \mid bD \mid b \\
C &\longrightarrow ab \mid aD \mid a \\
D &\longrightarrow aC \mid bB \mid b
\end{aligned}
$$

che genera le stringhe contenenti un numero dispari di a o un numero dispari di b.

Si costruisca l'espressione regolare corrispondente.

Espressioni regolari e ASF

Dato un ASFD $\mathscr{A}$, esiste una espressione regolare r tale che $L(\mathscr{A}) = L(r)$, che descrive cioè il linguaggio riconosciuto da $\mathscr{A}$.

Sia $\mathscr{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$ un ASFD e sia L il linguaggio da esso riconosciuto. Assumiamo $F = \{q_F\}$.

Sia $n = |Q|$ e sia $\langle q_0, \dots, q_{n-1} \rangle$ un qualunque ordinamento degli stati tale che $q_{n-1} = q_F$.

Definiamo ora come

$$R_{ij}^{k} \qquad 0 \leq i, j \leq n-1; 0 \leq k \leq n$$

l'insieme delle stringhe tali da portare $\mathscr{A}$ da q_i a q_j senza transitare per nessuno stato q_h con $h \geq k$.

Abbiamo cioè che $x = a_1, \ldots, a_m \in R_{ij}^{k}$ se e solo se:

1. $\overline{\delta}(q_i, x) = q_j$;
2. se $\overline{\delta}(q_i, a_1 \ldots a_l) = q_{i_l}$ allora $i_l < k$, per $1 \leq l \leq m-1$.

Per $k = 0$ si ha:

$$R_{ij}^{0} = \begin{cases} \bigcup\{a\} & \text{tali che } \delta(q_i, a) = q_j, \text{ se ne esiste almeno uno;} \\ \emptyset & \text{altrimenti.} \end{cases}$$

Per $0 < k \leq n$, se $x \in R_{ij}^{k}$ è una stringa che conduce da q_i a q_j senza transitare per nessuno stato q_h con $h \geq k$, possono verificarsi due casi:

1. x conduce da q_i a q_j senza transitare neanche per q_{k-1}, dal che deriva che $x \in R_{ij}^{k-1}$.
2. x conduce da q_i a q_j transitando per q_{k-1}

Nel secondo caso la sequenza degli stati attraversati può essere divisa in varie sottosequenze:

1. una prima sequenza, da q_i a q_{k-1} senza transitare per nessuno stato q_h con $h \geq k-1$, la corrispondente sottostringa di x appartiene quindi a $R^{k-1}_{i(k-1)}$
2. $r \geq 0$ sequenze, ognuna delle quali inizia e termina in q_{k-1} senza transitare per nessuno stato q_h con $h \geq k-1$, le corrispondenti sottostringhe di x appartengono quindi ciascuna a $R^{k-1}_{(k-1)(k-1)}$
3. una sequenza finale, da q_{k-1} a q_j senza transitare per nessuno stato q_h con $h \geq k-1$, la corrispondente sottostringa di x appartiene quindi a $R^{k-1}_{(k-1)j}$.

In conseguenza, ne deriva la relazione

$$R^k_{ij} = R^{k-1}_{ij} \cup R^{k-1}_{i(k-1)} \circ (R^{k-1}_{(k-1)(k-1)})^* \circ R^{k-1}_{(k-1)j}$$

Dalle osservazioni precedenti deriva che è possibile costruire tutti gli insiemi R_{ij}^{k} a partire da $k = 0$ e derivando poi man mano i successivi.

Osserviamo anche che $L = R_{0(n-1)}^{n}$

Ogni insieme di stringhe R_{ij}^k può essere descritto per mezzo di una opportuna espressione regolare r_{ij}^k, infatti abbiamo che, per $k = 0$,

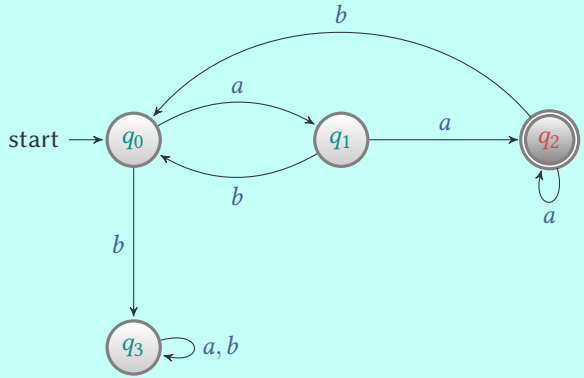
$$r_{ij}^0 = \begin{cases} a_{i_1} + \ldots + a_{i_l} & \text{dove } \delta(q_i, a_{i_k}) = q_j, k = 1, \ldots, l; \\ \emptyset & \text{se } l = 0. \end{cases}$$

Per $k \geq 1$, abbiamo che, dalla relazione tra R^k_{ij}, $R^{k-1}_{i(k-1)}$, $R^{k-1}_{(k-1)(k-1)}$ e $R^{k-1}_{(k-1)j}$, deriva che

$$r^k_{ij} = r^{k-1}_{ij} + r^{k-1}_{i(k-1)}(r^{k-1}_{(k-1)(k-1)})^* r^{k-1}_{(k-1)j}$$

Quindi, il linguaggio L sarà descritto dall'espressione regolare

$$r^n_{0(n-1)}$$


b
a
start
q_0
q_1
a
q_2
b
a
b
q_3
a, b

Assumiamo l'ordinamento $q_1 = q_0, q_2 = q_1, q_3 = q_3, q_4 = q_2$ tra gli stati. Allora:

$$r_{00}^0 = \emptyset;\ r_{01}^0 = a;\ r_{02}^0 = b;\ r_{03}^0 = \emptyset;$$

$$r_{10}^0 = b;\ r_{11}^0 = \emptyset;\ r_{12}^0 = \emptyset;\ r_{13}^0 = a;$$

$$r_{20}^0 = \emptyset;\ r_{21}^0 = \emptyset;\ r_{22}^0 = a + b;\ r_{23}^0 = \emptyset;$$

$$r_{30}^0 = b;\ r_{31}^0 = \emptyset;\ r_{32}^0 = \emptyset;\ r_{33}^0 = a;$$

$$
\begin{aligned}
r^1_{00} &= r^0_{00} + r^0_{00}(r^0_{00})^* r^0_{00} = \emptyset + \emptyset(\emptyset)^*\emptyset = \emptyset \\
r^1_{01} &= r^0_{01} + r^0_{00}(r^0_{00})^* r^0_{01} = a + \emptyset(\emptyset)^* a = a \\
r^1_{02} &= r^0_{02} + r^0_{00}(r^0_{00})^* r^0_{02} = b + \emptyset(\emptyset)^* b = b \\
r^1_{03} &= r^0_{03} + r^0_{00}(r^0_{00})^* r^0_{03} = \emptyset + \emptyset(\emptyset)^*\emptyset = \emptyset \\
r^1_{10} &= r^0_{10} + r^0_{10}(r^0_{00})^* r^0_{00} = b + b(\emptyset)^*\emptyset = b \\
r^1_{11} &= r^0_{11} + r^0_{10}(r^0_{00})^* r^0_{01} = \emptyset + b(\emptyset)^* a = ba \\
r^1_{12} &= r^0_{12} + r^0_{10}(r^0_{00})^* r^0_{02} = \emptyset + b(\emptyset)^* b = bb \\
r^1_{13} &= r^0_{13} + r^0_{10}(r^0_{00})^* r^0_{03} = a + b(\emptyset)^*\emptyset = a \\
&\dots
\end{aligned}
$$

$$r_{00}^{1} = \emptyset;\ r_{01}^{1} = a;\ r_{02}^{1} = b;\ r_{03}^{1} = \emptyset;$$

$$r_{10}^{1} = b;\ r_{11}^{1} = ba;\ r_{12}^{1} = bb;\ r_{13}^{1} = a;$$

$$r_{20}^{1} = \emptyset;\ r_{21}^{1} = \emptyset;\ r_{22}^{1} = a + b;\ r_{23}^{1} = \emptyset;$$

$$r_{30}^{1} = b;\ r_{31}^{1} = ba;\ r_{32}^{1} = bb;\ r_{33}^{1} = a;$$

$$r^2_{00} = a(ba)^*b;\ r^2_{01} = a + a(ba)^*ba;\ r^2_{02} = b + a(ba)^*bb;\ r^2_{03} = a(ba)^*a;$$

$$r^2_{10} = b + ba(ba)^*b;\ r^2_{11} = ba + ba(ba)^*ba;\ r^2_{12} = bb + ba(ba)^*bb; r^2_{13} = a + ba(ba)^*a;$$

$$r^2_{20} = \emptyset;\ r^2_{21} = \emptyset;\ r^2_{22} = a + b;\ r^2_{23} = \emptyset;$$

$$r^2_{30} = b + ba(ba)^*b;\ r^2_{31} = ba + ba(ba)^*ba;\ r^2_{32} = bb + ba(ba)^*bb;\ r^2_{33} = a + ba(ba)^*a;$$

Il linguaggio accettato dall'automa sarà descritto dall'espressione regolare

$$r^4_{03}$$

State elimination

Procedura iterativa di eliminazione degli stati su un automa non deterministico generalizzato equivalente, in cui:

1. la funzione di transizione è definita su $Q \times E$, dove E è l'insieme delle espressioni regolari su Σ , per cui gli archi sono etichettati con e.r.
2. lo stato iniziale non ha archi entranti, per cui $\nexists q \in Q, e \in E : q_0 \in \delta_N(q, e)$
3. esiste un solo stato finale q_F senza archi uscenti, per cui $\nexists e \in E : \delta_N(q_F, e) \neq \emptyset$

State elimination

Dato un qualunque automa $\mathscr{A}$ non deterministico, un automa generalizzato $\mathscr{A}'$ equivalente può essere immediatamente ottenuto:

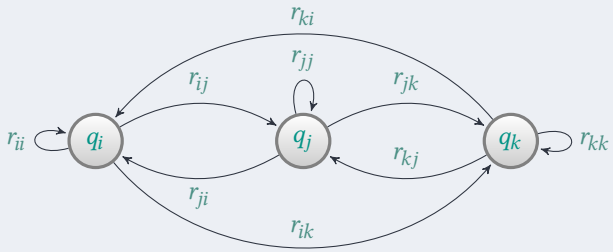
1. mantenendo gli stati di $\mathscr{A}$
2. introducendo, per ogni arco del grafo di transizione di $\mathscr{A}$ etichettato con l'insieme $a_1, \ldots, a_k$, un arco nel grafo di transizione di $\mathscr{A}'$ etichettato $a_1 + \ldots + a_k$
3. se lo stato iniziale q_0 di $\mathscr{A}$ ha archi entranti, introducendo in $\mathscr{A}'$ un nuovo stato iniziale $\overline{q}_0$ senza archi entranti, e la ε-transizione $\delta'_N(\overline{q}_0, \varepsilon) = \{q_0\}$
4. se esistono più stati finali in F, o se il solo stato finale ha archi uscenti, introducendo un ulteriore stato q_F, ponendo $F' = q_F$ e introducendo la ε-transizione $\delta'_N(q, \varepsilon) = \{q_F\}$ per ogni $q \in F$

State elimination

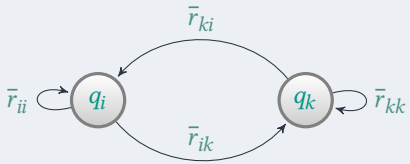
- Dato un automa nondeterministico (con ε-transizioni) $\mathscr{A}$ con insieme di stati Q, e dato uno stato q non iniziale né finale, è possibile ottenere un automa generalizzato equivalente $\mathscr{A}'$ con stati $Q - \{q\}$ effettuando una opportuna operazione di eliminazione dello stato
- L'eliminazione dello stato viene effettuata considerando tutti i possibili cammini di lunghezza 3 passanti per q (sequenze q_i, q, q_j per le quali esistono archi da q_i a q e da q a q_j)
- Per ogni cammino, le etichette degli archi interessati vengono modificate come mostrato di seguito
- Al termine, rimangono lo stato iniziale e quello finale, collegati da un arco, la cui etichetta fornisce l'espressione regolare cercata

State elimination

Da

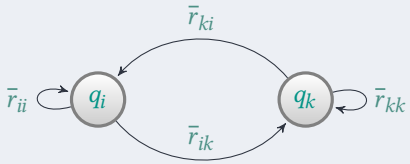


a



State elimination

Le espressioni regolari risultanti possono comunque essere complesse



$$\bar{r}_{ii} = r_{ij}(r_{jj} + r_{jk}r_{kk}^*r_{kj})^*r_{ji} + r_{ij}(r_{jj} + r_{jk}r_{kk}^*r_{kj})^*r_{jk}r_{kk}^*r_{ki} + r_{ik}(r_{kj}r_{jj}^*r_{jk} + r_{kk})^*r_{kj}r_{jj}^*r_{ji} + r_{ik}r_{ki}$$
$$\bar{r}_{kk} = r_{kj}(r_{jj} + r_{kj}r_{ii}^*r_{ij})^*r_{jk} + r_{kj}(r_{jj} + r_{ji}r_{ii}^*r_{ij})^*r_{ji}r_{ii}^*r_{ik} + r_{ki}(r_{ij}r_{jj}^*r_{ji} + r_{ii})^*r_{ij}r_{jj}^*r_{jk} + r_{ki}r_{ik}$$
$$\bar{r}_{ik} = r_{ik} + r_{ij}r_{jj}^*r_{jk}$$
$$\bar{r}_{ki} = r_{ki} + r_{kj}r_{jj}^*r_{jk}$$

In effetti, se esistono n cammini $q_iq_jq_h$ ($h = k_1, \dots, k_n$), allora si ha che

$$\bar{r}_{ik} = r_{ik_1} + r_{ij}r_{jj}^*r_{jk_1} + r_{ik_2} + r_{ij}r_{jj}^*r_{jk_2} + \dots + r_{ik_n} + r_{ij}r_{jj}^*r_{jk_n}$$

lo stesso, evidentemente, vale per $\bar{r}_{ki}$

Chiusura dei linguaggi regolari

Proprietà di chiusura dei linguaggi regolari

Dato l'insieme dei linguaggi regolari, ci chiediamo quali siano le sue proprietà di chiusura rispetto a varie operazioni.

Ci chiediamo quindi se, data una qualunque operazione $\odot$:

- se $\odot$ è unaria, $\odot L$ è un linguaggio regolare per ogni L regolare
- se $\odot$ è binaria, $L_1 \odot L_2$ è un linguaggio regolare per ogni coppia L_1, L_2 di linguaggi regolari
- non consideriamo operatori di "arità" maggiore

Le dimostrazioni di tali proprietà presentano tutte uno stesso schema, in cui, a partire dagli ASFD che riconoscono i linguaggi regolari dati, viene derivato un automa (deterministico o non deterministico) che riconosce il linguaggio risultante.

Chiusura dei linguaggi regolari: unione

Dati L_1 e L_2, la loro unione $L_1 \cup L_2$ è un linguaggio regolare.

Infatti, siano $A_1 = \langle \Sigma_1, Q_1, \delta_{N_1}, q_{0_1}, F_1 \rangle$ e $A_2 = \langle \Sigma_2, Q_2, \delta_{N_2}, q_{0_2}, F_2 \rangle$, due ASFD che accettano L_1 e L_2. Costruiamo da A_1 e A_2 un automa $A = \{\Sigma, Q, \delta_N, q_0, F\}$ che riconosce il linguaggio $L_1 \cup L_2$.

Chiusura dei linguaggi regolari: unione

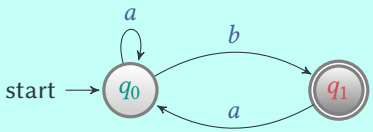
Costruzione:

1. $\Sigma = \Sigma_1 \cup \Sigma_2$.
2. $Q = Q_1 \cup Q_2 \cup \{q_0\}$.
3. $F = F_1 \cup F_2$, oppure $F = F_1 \cup F_2 \cup \{q_0\}$ se uno dei due automi A_1, A_2 riconosce anche la stringa vuota.
4. La funzione di transizione δ_N è definita come:

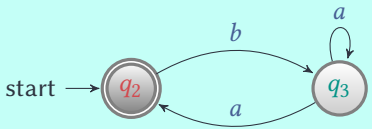
$$\begin{aligned} \delta_N(q, a) &= \delta_{N_1}(q, a) \text{ se } q \in Q_1, a \in \Sigma_1 \\ \delta_N(q, a) &= \delta_{N_2}(q, a) \text{ se } q \in Q_2, a \in \Sigma_2 \\ \delta_N(q_0, a) &= \delta_{N_1}(q_{0_1}, a) \cup \delta_{N_2}(q_{0_2}, a), a \in \Sigma. \end{aligned}$$

Esempio

A_1

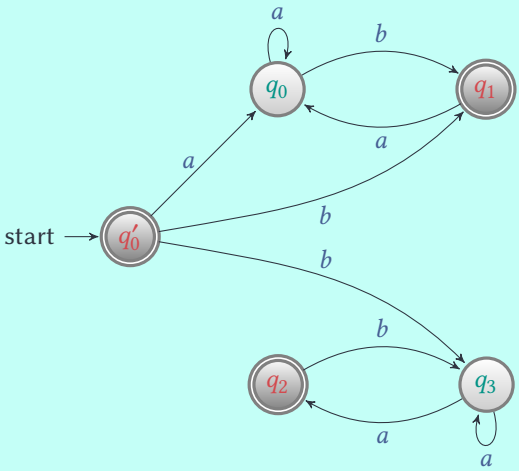


A_2



Esempio

A



Chiusura dei linguaggi regolari: complemento

Dato un linguaggio regolare L, il suo complemento $\overline{L}$ è un linguaggio regolare.

Infatti, sia $A = \langle \Sigma, Q, \delta, q_0, F \rangle$ un automa deterministico che riconosce L, con δ funzione totale: l'automa

$$\overline{A} = \langle \Sigma, Q, \delta, q_0, \{Q - F\} \rangle;$$

riconosce allora il linguaggio $\overline{L}$.

Infatti, ogni stringa che porta l'automa A in uno stato finale porta l'automa $\overline{A}$ in uno stato non finale e, viceversa, ogni stringa che porta $\overline{A}$ in uno stato finale porta A in uno stato non finale

Chiusura dei linguaggi regolari: intersezione

Dati due linguaggi regolari L_1 e L_2, la loro intersezione $L = L_1 \cap L_2$ è un linguaggio regolare.

È sufficiente osservare che, per la legge di De Morgan,

$$L = L_1 \cap L_2 \equiv \overline{\overline{L_1} \cup \overline{L_2}}.$$

Chiusura dei linguaggi regolari: concatenazione

Dati due linguaggi regolari L_1 e L_2, la loro concatenazione $L = L_1 \circ L_2$ è un linguaggio regolare.

Infatti, siano $A_1 = \langle \Sigma_1, Q_1, \delta_{N_1}, q_{0_1}, F_1 \rangle$ e $A_2 = \langle \Sigma_2, Q_2, \delta_{N_2}, q_{0_2}, F_2 \rangle$, due ASFD che accettano L_1 e L_2. Costruiamo da A_1 e A_2 un automa $A = \{\Sigma, Q, \delta_N, q_0, F\}$ che riconosce il linguaggio $L_1 \circ L_2$.

Chiusura dei linguaggi regolari: concatenazione

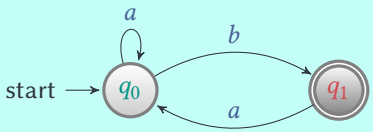
Costruzione:

1. $\Sigma = \Sigma_1 \cup \Sigma_2$;
2. $Q = Q_1 \cup Q_2$;
3. $F = \begin{cases} F_2 & \text{se } \varepsilon \notin L(A_2), \\ F_1 \cup F_2 & \text{altrimenti;} \end{cases}$
4. $q_0 = q_{01}$;
5. δ_N è definita come:

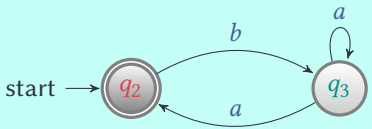
$$\begin{aligned} \delta_N(q,a) &= \delta_1(q,a), \forall q \in Q_1 - F_1, a \in \Sigma_1 \\ \delta_N(q,a) &= \delta_1(q,a) \cup \delta_2(q_{02},a), \forall q \in F_1, a \in \Sigma \\ \delta_N(q,a) &= \delta_2(q,a), \forall q \in Q_2, a \in \Sigma_2. \end{aligned}$$

Esempio

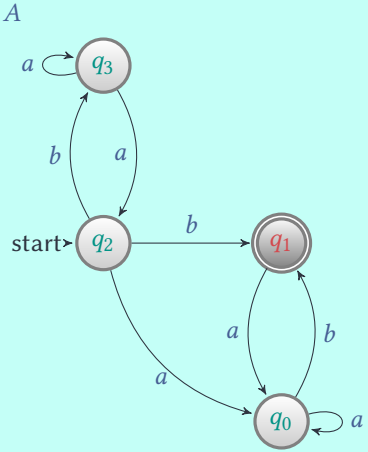
A_1



A_2



Esempio



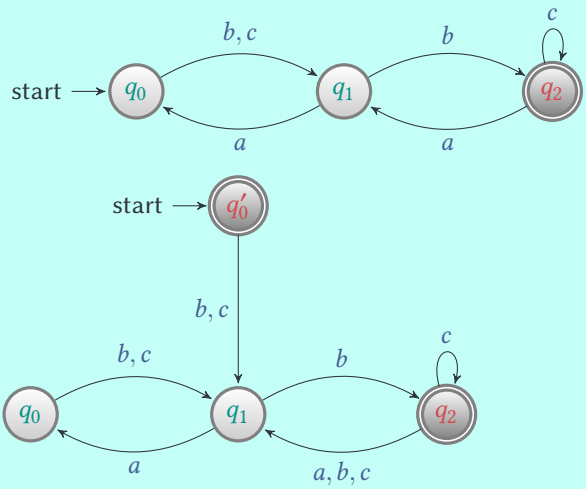
Chiusura dei linguaggi regolari: iterazione

Dato un linguaggo regolare L, la sua iterazione L^* è un linguaggio regolare.

Infatti, sia $A = \langle \Sigma, Q, \delta_N, q_0, F \rangle$ un ASFD che accetta L. Costruiamo da A un automa $A' = \{\Sigma, Q \cup \{q_0'\}, \delta', q_0', F \cup \{q_0'\}\}$ che riconosce il linguaggio L^* ponendo

$$\begin{aligned} \delta'(q, a) &= \delta(q, a), \ \forall q \in Q - F \\ \delta'(q, a) &= \delta(q, a) \cup \delta(q_0, a), \ \forall q \in F \\ \delta'(q_0', a) &= \delta(q_0, a). \end{aligned}$$

Esempio



Decidibilità di predicati

Decidibilità di predicati su LR

Teorema

Decidibilità di predicati su LR

Osservazione: se un automa con n stati accetta qualche stringa, allora accetta una stringa di lunghezza inferiore ad n.

Sia z la stringa più breve accettata dall'automa, se fosse $\mid z \mid \geq n$ allora, in base al pumping lemma, z potrebbe essere scritta come uvw e anche la stringa uw, più breve di z, sarebbe accettata dall'automa.

Decidibilità di predicati su LR

Dato un automa A con $n = |Q|$ stati e $s = |\Sigma|$ simboli di input, per decidere se $L(\mathscr{A}) = \Lambda$ basta verificare se A accetta almeno una delle

$$\sum_{i=0}^{n-1} s^i = \frac{s^n - 1}{s - 1} = \theta(s^{n-1})$$

stringhe di lunghezza inferiore ad n.

Il tempo necessario per tale operazione sarà evidentemente proporzionale a:

$$\sum_{i=0}^{n-1} i s^i = s \sum_{i=1}^{n} \frac{ds^i}{ds} = s \frac{d}{ds} \sum_{i=1}^{n} s^i = s \frac{d}{ds} \frac{s^n - 1}{s - 1} = \frac{n s^{n+2} - s^{n+1} + s}{(s-1)^2} = \theta(s^n)$$

L'algoritmo ha quindi complessità esponenziale.

Decidibilità di predicati su LR

Osservazione: dato il grafo di transizione di A, $L(\mathscr{A})$ è non vuoto se e solo se esiste almeno uno stato finale raggiungibile da q_0.

Algoritmo polinomiale: visita del grafo di transizione, avente n nodi e $m \leq ns$ archi

La visita richiede tempo $\theta(n + m) = \theta(n)$ lineare nel numero di stati.

Decidibilità di predicati su LR

Teorema

Decidibilità di predicati su LR

Proprietà 1: se A accetta una z lunga almeno n allora, per il pumping lemma, accetta infinite stringhe.

Decidibilità di predicati su LR

Proprietà 2: se $L(\mathscr{A})$ è infinito, allora esiste $z \in L(\mathscr{A})$ tale che $n \leq\mid z \mid< 2n$.

- se $L(\mathscr{A})$ è infinito esiste chiaramente $z \in L(\mathscr{A})$ con $|z| \geq n$, altrimenti $L(\mathscr{A})$ sarebbe finito (e composto di stringhe di lunghezza al più $n-1$)

Decidibilità di predicati su LR

- assumiamo, senza perdere generalità, che z sia una stringa di lunghezza minima tra tutte stringhe in $L(\mathscr{A})$ lunghe almeno n. Due casi sono possibili:
  - $|z| < 2n$, e quindi la proprietà è vera
  - $|z| \geq 2n$: in questo caso, per il pumping lemma (visto che $|z| > n$) possiamo scrivere $z = uvw$, con $1 \leq| v |\leq n$. Per il pumping lemma, $uw \in L(\mathscr{A})$ e $|uw| < |uvw|$: data l'ipotesi che z abbia lunghezza minima tra tutte le stringhe di $L(\mathscr{A})$ lunghe almeno n, ne consegue che deve essere $|uw| < n$. Ma questo è impossibile, in quanto $|uw| = |z| - |v| \geq 2n - n = n$ in quanto per ipotesi $|z| \geq 2n$ e $|v| \leq n$. Ne deriva che la stringa in $L(\mathscr{A})$ di lunghezza minima tra tutte quelle lunghe almeno n non può essere lunga di $2n$ o più, e quindi la proprietà è vera

Decidibilità di predicati su LR

In conclusione:

$L(\mathscr{A})$ è infinito se e solo se esiste $z \in L(\mathscr{A})$ tale che $n \leq\mid z \mid< 2n$.

Dato un automa A con n stati e s simboli di input, per decidere se $L(\mathscr{A})$ è infinito basta verificare se A accetta almeno una delle

$$\sum_{i=n}^{2n-1} s^i = \sum_{i=0}^{2n-1} s^i - \sum_{i=0}^{n-1} s^i = \frac{(s^{2n}-1)-(s^n-1)}{s-1} = \frac{s^n(s^n-1)}{s-1} = \theta(s^{2n-1})$$

stringhe di lunghezza compresa tra n e $2n$.

Applicando le stesse considerazioni viste sopra tempo necessario per tale operazione sarà evidentemente proporzionale a:

$$\sum_{i=n}^{2n-1} is^i = \sum_{i=0}^{2n-1} is^i - \sum_{i=0}^{n-1} is^i = \frac{2ns^{2n+2} - s^{2n+1} + s}{(s-1)^2} - \frac{ns^{n+2} - s^{n+1} + s}{(s-1)^2} = \theta(s^{2n})$$

L'algoritmo ha quindi complessità esponenziale.

Decidibilità di predicati su LR

Osservazione: dato il grafo di transizione di A, $L(\mathscr{A})$ è infinito se e solo se esiste almeno un ciclo a partire da uno stato su un cammino da q_0 a uno stato finale.

Algoritmo polinomiale: visita del grafo di transizione, avente n nodi e $m \leq ns$ archi.

La visita richiede tempo $\theta(n + m) = \theta(n)$ lineare nel numero di stati.

Decidibilità di predicati su LR

Dalla possibilità di decidere se un linguaggio regolare è vuoto, ne deriva immediatamente la decidibilità di altri predicati sui linguaggi regolari.

Teorema

Decidibilità di predicati su LR

Per dimostrare l'equivalenza di due linguaggi, basta dimostrare che la loro intersezione coincide con la loro unione, cioè che la loro differenza simmetrica (l'unione dell'intersezione del primo con il complemento del secondo e dell'intersezione del secondo con il complemento del primo) è il linguaggio vuoto.

Dimostrazione di non regolarità

Pumping lemma

Ogni stringa sufficientemente lunga appartenente ad un linguaggio regolare presenta delle regolarità: in particolare, contiene una sottostringa che può essere ripetuta quanto si vuole, ottenendo sempre stringhe del linguaggio.

Più precisamente:

sia L un linguaggio regolare : allora $\exists n > 0$ tale che per ogni $\forall z \in L : \mid z \mid \geq n$ possiamo scrivere $z = uvw$, con $\mid uv \mid \leq n$, $\mid v \mid \geq 1$ e ottenere che $\forall i \geq 0$, $uv^i w \in L$.

Pumping lemma: interpretazione come gioco a due

Se L è regolare, Alice vince sempre questo gioco con Bob:

1. Alice fissa un intero $n > 0$ opportuno
2. Bob sceglie una stringa $z \in L$ con $| z | > n$
3. Alice divide z in tre parti uvw con $| uv | \leq n$ e $| v | \geq 1$
4. Bob sceglie un intero $i \geq 0$
5. Alice mostra a Bob che $uv^i w \in L$

Pumping lemma: dimostrazione

Se L è regolare, sia A l'ASFD che lo decide e che ha il minimo numero n di stati.

Una stringa $z \in L$ di lunghezza $m \geq n$ in input a A gli fa eseguire m transizioni e quindi attraversare $m + 1 > n$ stati, quindi esiste almeno uno stato che viene attraversato più volte.

Pumping lemma: dimostrazione

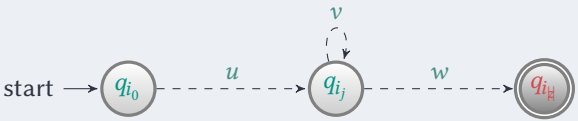
Indichiamo con $q_{i_0}, q_{i_1}, \dots, q_{i_{m+1}}$ la sequenza di stati, non tutti distinti, attraversati (chiaramente, $q_{i_0} = q_0$ e $q_{i_{m+1}} \in F$) e con q_{i_j} il primo stato della sequenza che ricompare in seguito, ad esempio come q_{i_k}.

Sia u il prefisso (eventualmente nullo) di z tale che $\overline{\delta}(q_0, u) = q_{i_j}$ e sia $z = ux$, per cui $\overline{\delta}(q_{i_j}, x) = q_{i_{m+1}}$.

Pumping lemma: dimostrazione

A quindi esegue una sotto-computazione $q_{i_j}, q_{i_{j+1}}, \dots, q_{i_k}$ di una computazione di accettazione (di z) che inizia e termina nello stesso stato. Si noti che la sotto-sequenza deve prevedere almeno due stati, per cui $q_{i_k} - q_{i_j} > 1$. Si noti inoltre che non possono essere stati attraversati più di n stati prima di arrivare a q_{i_k}, perché altrimenti questo non sarebbe il primo stato a comparire di nuovo.

Sia v il prefisso di x tale che $\overline{\delta}(q_{i_j}, v) = q_{i_k}$ e sia $x = vw$: da quanto detto, $| uv | \leq n$ e $\overline{\delta}(q_{i_j}, w) = q_{i_{|z|}}$



Pumping lemma: dimostrazione

Una computazione in cui questa sotto-sequenza è eseguita più volte è ancora una computazione di accettazione.

Per ogni $i \geq 0$ abbiamo infatti

$$\begin{aligned}\overline{\delta}(q_0, uv^i w) &= \overline{\delta}\Big(\overline{\delta}(q_0, u), v^i w\Big) = \overline{\delta}\left(q_{i_j}, v^i w\right) = \overline{\delta}\Big(\overline{\delta}(q_{i_j}, v), v^{i-1} w\Big) \\ &= \overline{\delta}\left(q_{i_j}, v^{i-1} w\right) = \ldots = \overline{\delta}(q_{i_j}, w) = q_{i_{m+1}} \in F\end{aligned}$$

il che mostra che ogni stringa del tipo $uv^i w$ appartiene ad L.

Pumping lemma

Evidenzia il fatto che gli automi finiti: non possono contare. Il numero di situazioni diverse che possono memorizzare è finito.

Fornisce soltanto una condizione necessaria perché un linguaggio sia regolare: non può essere utilizzato per mostrare la regolarità di un linguaggio, ma solo per dimostrarne la non regolarità.

$$L \text{ regolare} \implies \text{pumping lemma verificato}$$
$$\text{pumping lemma non verificato} \implies L \text{ non regolare}$$

Pumping lemma

Sia L un linguaggio e supponiamo che $\forall n > 0$ si abbia che $\exists z \in L \ : \mid z \mid \geq n$ tale che comunque dividiamo z in $z = uvw$, con $\mid uv \mid \leq n$, $\mid v \mid \geq 1$, $\exists i \geq 0$ tale che $uv^i w \notin L$. Allora, L non è regolare.

Pumping lemma: interpretazione come gioco a due

Se L non è regolare, Alice vince sempre questo gioco con Bob:

1. Bob fissa un intero $n > 0$
2. Alice sceglie una stringa opportuna $z \in L$, con $| z | > n$
3. Bob divide z in tre parti uvw con $| uv | \leq n$ e $| v | \geq 1$
4. Alice sceglie un intero $i \geq 0$ e mostra a Bob che $uv^i w \notin L$

Esempio

Consideriamo il linguaggio $L = a^k b^k$, $k > 0$: per mostrare che L non è regolare, interpretiamo il ruolo di Alice nel gioco.

1. Bob fissa un intero $n > 0$
2. Scegliamo la stringa $z = a^n b^n$
3. Bob divide z in tre parti uvw con $| uv | \leq n$ e $| v | \geq 1$: per la struttura di z, necessariamente $uv = a^h$, con $0 < h \leq n$. Quindi, $v = a^l$, per $0 < l < h$, e corrispondentemente $u = a^{h-l}$; inoltre, $w = a^{n-h} b^n$.
4. Scegliamo l'intero 2 e mostriamo a Bob che

$$uv^2w = a^{h-l} a^l a^l a^{n-h} b^n = a^{n+l} b^n \notin L$$

Esempio

Si consideri il linguaggio $L = \{w\tilde{w} \mid w \in \{a, b\}^*\}$, ove si è indicata con $\tilde{w}$ la stringa ottenuta invertendo i caratteri presenti in w.

Dimostrare, utilizzando il pumping lemma, che tale linguaggio non è regolare.

Esempio

Interpretiamo il ruolo di Alice nel gioco.

1. Bob fissa un intero $n > 0$
2. Scegliamo la stringa $z = a^n bba^n$
3. Bob divide z in tre parti uvw con $| uv | \leq n$ e $| v | \geq 1$: per la struttura di z, necessariamente $uv = a^h$, con $0 < h \leq n$. Quindi, $v = a^l$, per $0 < l < h$, e corrispondentemente $u = a^{h-l}$; inoltre, $w = a^{n-h} bba^n$.
4. Scegliamo l'intero 2 e mostriamo a Bob che

$$uv^2 w = a^{h-l} a^l a^l a^{n-h} bba^n = a^{n+l} bba^n \notin L$$