7 - Linguaggi regolari, espressioni regolari e Teorema di Kleene

Linguaggi regolari ed espressioni regolari

Definizione di linguaggio regolare
Sia X un alfabeto finito, un linguaggio L è regolare se è finito oppure se può essere ottenuto grazie alle operazioni di unione, concatenazione e iterazione:

1. $L = L_1 \cup L_2$ con L_1, L_2 regolari
2. $L = L_1 \cdot L_2$ con L_1, L_2 regolari
3. $L = L_1^*$ con L_1 regolare
   Si noti che $\varnothing$ e $\{\lambda\}$ sono linguaggi regolari, per denotarli usiamo il simbolo $\mathcal{L}_{REG}$

Espressioni regolari

Un espressione regolare non è nient'altro che una stringa fatta da simboli
Definizione
Sia X un alfabeto finito, una stringa R di alfabeto $X \cup \{\lambda, +, *, \cdot, \varnothing, (,)\}$ (con $X \cap \{\lambda, +, *, \cdot, \varnothing, (,)\} = \varnothing$) è una **espressione regolare** di alfabeto X se e solo se vale una delle seguenti condizioni:

1. $R = \varnothing$
2. $R = \lambda$
3. $R = a$, per ogni $a \in X$ (tutti i simboli)
4. $R = (R_1 + R_2)$ con R_1, R_2 espressioni regolari di alfabeto X
5. $R = (R_1 \cdot R_2)$ con R_1, R_2 espressioni regolari di alfabeto X
6. $R = (R_1)^*$ con R_1 espressione regolare di alfabeto X

Espressioni regolari e linguaggi regolari

Ad ogni espressione regolare R si denota un linguaggio regolare $S(R)$ definito nel modo seguente:

Espressione regolare	Linguaggio regolare corrispondente
`∅`	`∅` (linguaggio vuoto)
`λ`	`{λ}` (linguaggio contenente solo la stringa vuota)
`a` (dove `a ∈ X`)	`{a}` (linguaggio contenente il simbolo `a`)
`(R₁ + R₂)`	$S(R_1) \cup S(R_2)$ (unione dei linguaggi)
`(R₁ · R₂)`	$S(R_1) \cdot S(R_2)$ (concatenazione)

Espressione regolare	Linguaggio regolare corrispondente
`(R₁)*`	$(S(R_1))*$ (chiusura di Kleene)

Teorema di Kleene

Il **Teorema di Kleene** afferma l'equivalenza tra tre diverse definizioni di linguaggi regolari:

$$\mathcal{L}_3 \equiv \mathcal{L}_{FSL} \equiv \mathcal{L}_{REG}$$

Esistono 3 dimostrazioni possibili:

1. $\mathcal{L}_3 \subseteq \mathcal{L}_{FSL}$
2. $\mathcal{L}_{FSL} \subseteq \mathcal{L}_{REG}$
3. $\mathcal{L}_{REG} \subseteq \mathcal{L}_3$

Andremo a trattare soltanto il primo, linguaggi generati da grammatiche lineari destre (terzo tipo della gerarchia di Chomsky)

Dimostrazione teorema di Kleene

Sia $L \in \mathcal{L}_3, \exists G = (X, V, S, P)$ (con grammatica G di tipo 3) tale che $L = L(G)$.
Si costruisce un automa a stati finiti $M = (Q, \delta, q_0, F)$ tale che $T(M) = L(G)$, grazie a questo algoritmo andremo a dimostrare il Teorema di Kleene

Algoritmo: Costruzione di un automa a stati finiti non deterministico equivalente ad una grammatica lineare destra

Data:

- $G = (X, V, S, P)$ una grammatica lineare destra
  L'automa accettore a stati finiti equivalente $M = (Q, \delta, q_0, F)$ viene costruito come segue:

1. X come l'alfabeto di ingresso
2. $Q = V \cup \{q\}$, con $q \notin V$
3. $q_0 = S$
4. $F = \{q\} \cup \{B \mid B \to \lambda \in P\}$
5. La funzione di transizione $\delta : Q \times X \to 2^Q$ è definita nel modo seguente:
   - **(V.a)** $\forall B \to aC \in P, C \in \delta(B, a)$
   - **(V.b)** $\forall B \to a \in P, q \in \delta(B, a)$

L'algoritmo può generare un automa non deterministico per effetto dei passi V.a e V.b, si può facilmente constatare che, se $w = x_1, x_2 \ldots x_n \in L(G)$, w può essere generata da una derivazione del tipo:
$S \Rightarrow x_1X_2 \Rightarrow x_1x_2X_3 \Rightarrow x_1x_2 \ldots x_{i-1}X_i \Rightarrow x_1x_2 \ldots x_n$
Dalla definizione data, l'automa M, esaminando la stringa $w = x_1x_2 \ldots x_n$ compie una serie

di mosse (o transizioni) che lo portano dallo stato S ad $X_2, X_3 \ldots X_i$ e q ; pertanto $L(G) \subseteq T(M)$.
In modo del tutto analogo, ogni w in $T(M)$ comporta una sequenza di mosse dell'automa a cui corrisponde una derivazione in G, e pertanto $T(M) \subseteq L(G)$.
Se ne deduce che: $L(G) = T(M)$

Pumping lemma per i linguaggi regolari

Sia $M = (Q, \delta, q_0, F)$ un automa a stati finiti con n stati (cioè $|Q| = n$) e sia z una stringa appartenente a $T(M)$, con lunghezza $|z| \geq n$. Allora z può essere scritta nella forma $z = uvw$, con le seguenti proprietà:

- $v \neq \lambda$
- $|uv| \leq n$
- $\forall i \geq 0$, $uv^iw \in T(M)$

Una formulazione alternativa è la seguente:
sia $L = T(M)$ un linguaggio regolare, con $M = (Q, \delta, q_0, F)$ un automa a stati finiti. Allora:
$\forall z \in L, |z| \geq n \Rightarrow \exists u, v, w \in X^* \text{ t.c. } z = uvw$ e

- $v \neq \lambda$
- $|uv| \leq n$
- $\forall i \geq 0$, $uv^iw \in L$

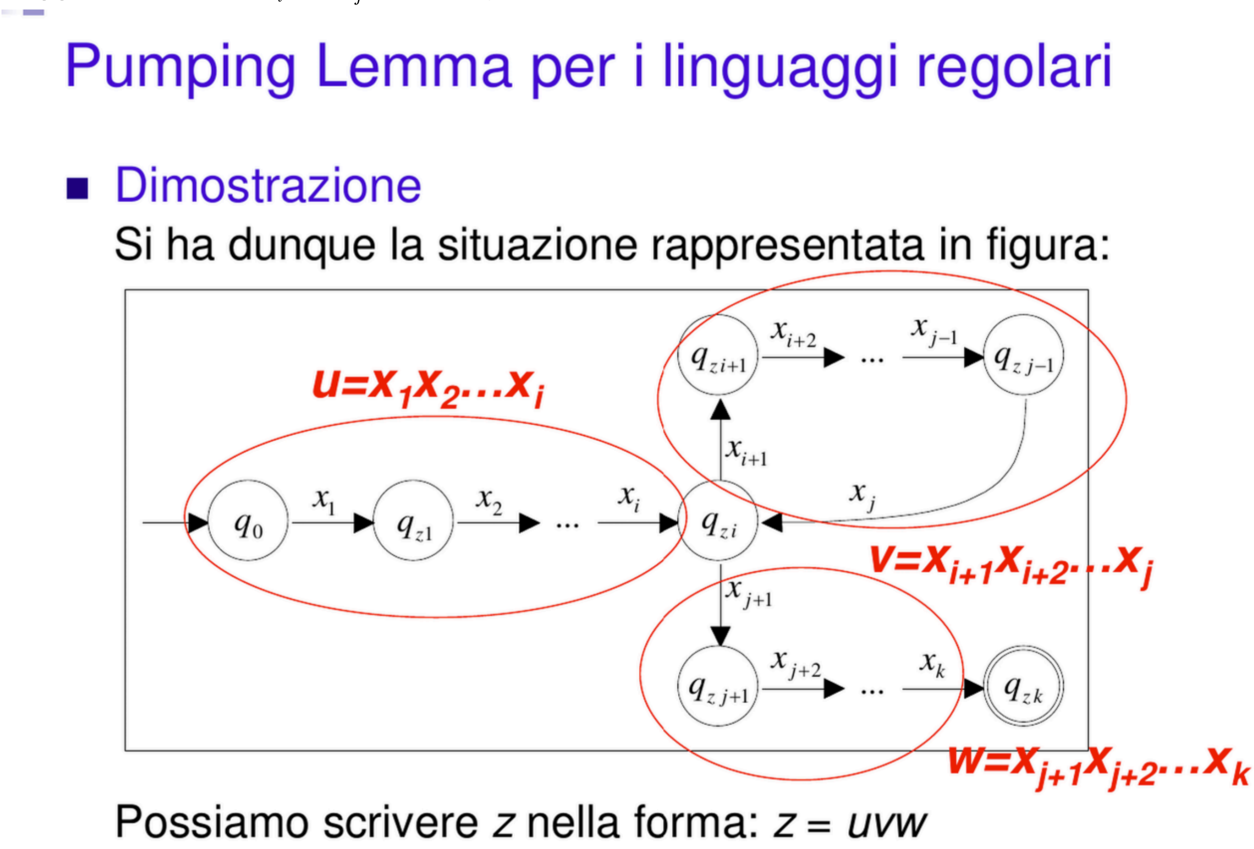
Dimostrazione

Sia $z = x_1x_2 \ldots x_k \in T(M)$. Consideriamo il comportamento dell'automa M quando elabora l'ingresso z. Questo può essere rappresentato come una sequenza di stati:

$$q_0 \xrightarrow{x_1} q_{z_1} \xrightarrow{x_2} q_{z_2} \xrightarrow{x_3} \cdots \xrightarrow{x_k} q_{z_k}$$

Se $|z| \geq n$, devono comparire almeno $n+1$ stati in questa sequenza. Tuttavia, poiché M ha solo n stati distinti, per il principio dei cassetti almeno uno stato nella sequenza $q_0, q_{z_1}, q_{z_2}, \ldots, q_{z_k}$ deve ripetersi.

Supponiamo che $q_{z_i} = q_{z_j}$ con $i < j$.



Quindi $z = uvw$.

Dal momento che $q_{z_i} = q_{z_j}$, l'automa ripete un ciclo quando legge v. Di conseguenza, passando da q_0 con l'ingresso uv^iw, per ogni $i \geq 0$, l'automa raggiunge ancora uno stato finale, poiché la sequenza delle transizioni che porta a uno stato finale si conserva.

Pertanto, ogni stringa della forma uv^iw per $i \geq 0$ appartiene a $T(M)$, cioè:

$\forall i \geq 0, uv^iw \in T(M)$