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)

Espressione regolari equivalenti

Due espressioni regolari R_1 e R_2 su X sono equivalenti se e solo se $S(R_1) = S(R_2)$

Proprietà delle espressioni regolari

Proprietà 1 – Associatività dell'operazione "+" (unione)

L'operazione di unione tra espressioni regolari è associativa, quindi si ha che:

$$(R_1 + R_2) + R_3 = R_1 + (R_2 + R_3) = R_1 + R_2 + R_3$$

Proprietà 2 – Commutatività dell'operazione "+"

L'ordine delle espressioni non influisce sull'unione:

$$R_1 + R_2 = R_2 + R_1$$

Proprietà 3 – $\emptyset$ è l'elemento neutro per "+"

L'unione di un'espressione con l'insieme vuoto restituisce l'espressione stessa:

$$\emptyset + R_1 = R_1 + \emptyset = R_1$$

Proprietà 4 – Idempotenza dell'operazione "+"

Unendo un'espressione con sé stessa si ottiene ancora l'espressione:

$$R_1 + R_1 = R_1$$

Proprietà 5 – Associatività della concatenazione

La concatenazione è associativa:

$$(R_1 \cdot R_2) \cdot R_3 = R_1 \cdot (R_2 \cdot R_3) = R_1 \cdot R_2 \cdot R_3$$

Proprietà 6 – Non commutatività della concatenazione

In generale, la concatenazione non è commutativa:

$$R_1 \cdot R_2 \neq R_2 \cdot R_1$$

Proprietà 7 – λ è l'elemento neutro per la concatenazione

La concatenazione con la stringa vuota λ lascia invariata l'espressione:

$$\lambda \cdot R_1 = R_1 \cdot \lambda = R_1$$

Proprietà 8 – $\emptyset$ è l'elemento assorbente per la concatenazione

Concatenando $\emptyset$ con qualsiasi espressione si ottiene $\emptyset$:

$$\emptyset \cdot R_1 = R_1 \cdot \emptyset = \emptyset$$

Proprietà 9 – Distributività sinistra della concatenazione rispetto a “+”

La concatenazione si distribuisce a sinistra sull'unione:

$$R_1 \cdot (R_2 + R_3) = R_1 \cdot R_2 + R_1 \cdot R_3$$

Proprietà 10 – Distributività destra della concatenazione rispetto a “+”

Anche a destra la concatenazione si distribuisce:

$$(R_2 + R_3) \cdot R_1 = R_2 \cdot R_1 + R_3 \cdot R_1$$

Proprietà 11 – Chiusura di Kleene su R_1 equivale a unione con sé stessa iterata

La chiusura di Kleene di un’espressione è equivalente a:

$$R_1^* = \lambda + R_1 + R_1 \cdot R_1 + R_1 \cdot R_1 \cdot R_1 + \ldots$$

Proprietà 12 – Chiusura di λ e $\emptyset$

Le chiusure di Kleene sulle espressioni λ e $\emptyset$ risultano:

$$\lambda^* = \{\lambda\}, \quad \emptyset^* = \{\lambda\}$$

Proprietà 13 – Chiusura su somma iterata

Una concatenazione di un numero arbitrario di ripetizioni di unione di espressioni è ancora esprimibile con la chiusura:

$$(R_1 + R_2 + \ldots + R_n)^* = R_1^* + R_2^* + \ldots + R_n^* + \ldots$$

Proprietà 14 – Variante della proprietà 11 con concatenazione

Anche in presenza di chiusure concatenate con λ, si ha:

$$\lambda + R_1^* = R_1^* = R_1^* + \lambda$$

Proprietà 15 – Non generalità della distribuzione di chiusura su somma

In generale non vale che:

$$(R_1 + R_2)^* = R_1^* + R_2^*$$

Proprietà 16 – Chiusura concatenata con se stessa

Concatenando una chiusura con sé stessa non cambia nulla:

$$R_1^* \cdot R_1^* = R_1^*$$

Proprietà 17 – Espressione equivalente con distribuzione delle chiusure

Si ha l'equivalenza:

$$(R_1 \cdot R_2)^* \cdot R_1 = R_1 \cdot (R_2 \cdot R_1)^*$$

Proprietà 18 – Equivalenza tra forme distribuite di chiusura con somma

Vale anche:

$$R_1^* \cdot (R_2 + R_1 \cdot R_1^* \cdot R_2)^* = (R_1 + R_1 \cdot R_1^* \cdot R_2)^*$$

Proprietà 19 – Equivalenza alternativa per la proprietà 18

Un'altra forma equivalente è:

$$(R_1 + R_1 \cdot R_2)^* \cdot R_1 = R_1 \cdot (R_2 \cdot R_1)^*$$

Proprietà 20 – Equivalenza condizionata con esclusione di λ

La seguente equivalenza è vera **se e solo se** $\lambda \notin S(R_2)$, ovvero se λ **non appartiene** al linguaggio generato da R_2:

$$(R_1 \cdot R_2)^* \cdot R_1 = R_1 \cdot (R_2 \cdot R_1)^* \quad \text{se e solo se} \quad \lambda \notin S(R_2)$$

Dimostrazioni delle proprietà

Le proprietà da 1 a 5 e da 7 a 14 si possono dimostrare ricorrendo alla funzione S, che associa ad ogni espressione regolare il linguaggio corrispondente. Tuttavia, per la maggior parte delle proprietà da 1 a 20, si può usare una tecnica generale detta **dimostrazione mediante riparsificazione**, che consiste nel mostrare che ogni parola generata da un'espressione può essere riorganizzata in modo da soddisfare l'altra espressione, e viceversa, dimostrando quindi l'equivalenza dei linguaggi.
[(GUARDARE ESEMPI DALLE SLIDE)]

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

Da grammatica ad automa

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

    $$G = (X, V, S, P)$$

    l'automa accettore a stati finiti equivalente ($T(M) = L(G)$) viene costruito come segue:

    $$M = (Q, \delta, q_0, F)$$

    - (I) X come alfabeto di ingresso;
    - (II) $Q = V \cup \{q\},\ q \notin V$
    - (III) $q_0 = S$
    - (IV) $F = \{q\} \cup \{B \mid B \rightarrow \lambda \in P\}$
    - (V) $\delta : Q \times X \rightarrow 2^Q \quad \ni' \quad V.a \quad \forall B \rightarrow aC \in P,\ C \in \delta(B,a)$

      $V.b \quad \forall B \rightarrow a \in P,\ q \in \delta(B,a)$

Da automa a grammatica

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

- Sia dato un automa accettore a stati finiti:

  $$M = (Q, \delta, q_0, F)$$

  con alfabeto di ingresso X.
  La grammatica lineare destra G equivalente a M, ossia tale che $L(G) = T(M)$, si costruisce come segue:
  - (I) X = alfabeto di ingresso di M
  - (II) $V = Q;$
  - (III) $S = q_0;$
  - (IV) $P = \{q \to xq' \mid q' \in \delta(q,x)\} \cup \{q \to x \mid \delta(q,x) \in F\} \cup \{q_0 \to \lambda \mid q_0 \in F\}$

Pumping lemma per i linguaggi regolari

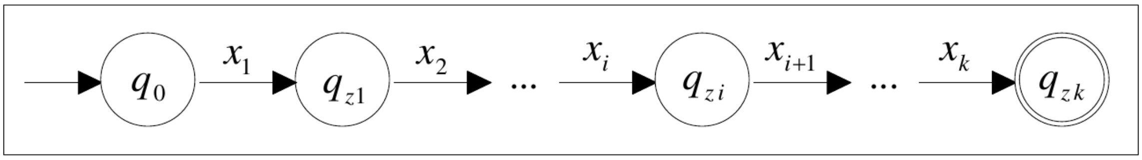
Pumping Lemma per i linguaggi regolari

- Sia $M = (Q, \delta, q_0, F)$ un automa accettore a stati finiti con n stati ($|Q|=n$) e sia $z \in T(M)$, $|z| \geq n$.
  Allora z può essere scritta come uvw, e $uv^*w \subset T(M)$ (ossia $\forall i,\ i \geq 0 : uv^iw \in T(M)$).
- Una formulazione alternativa è la seguente:

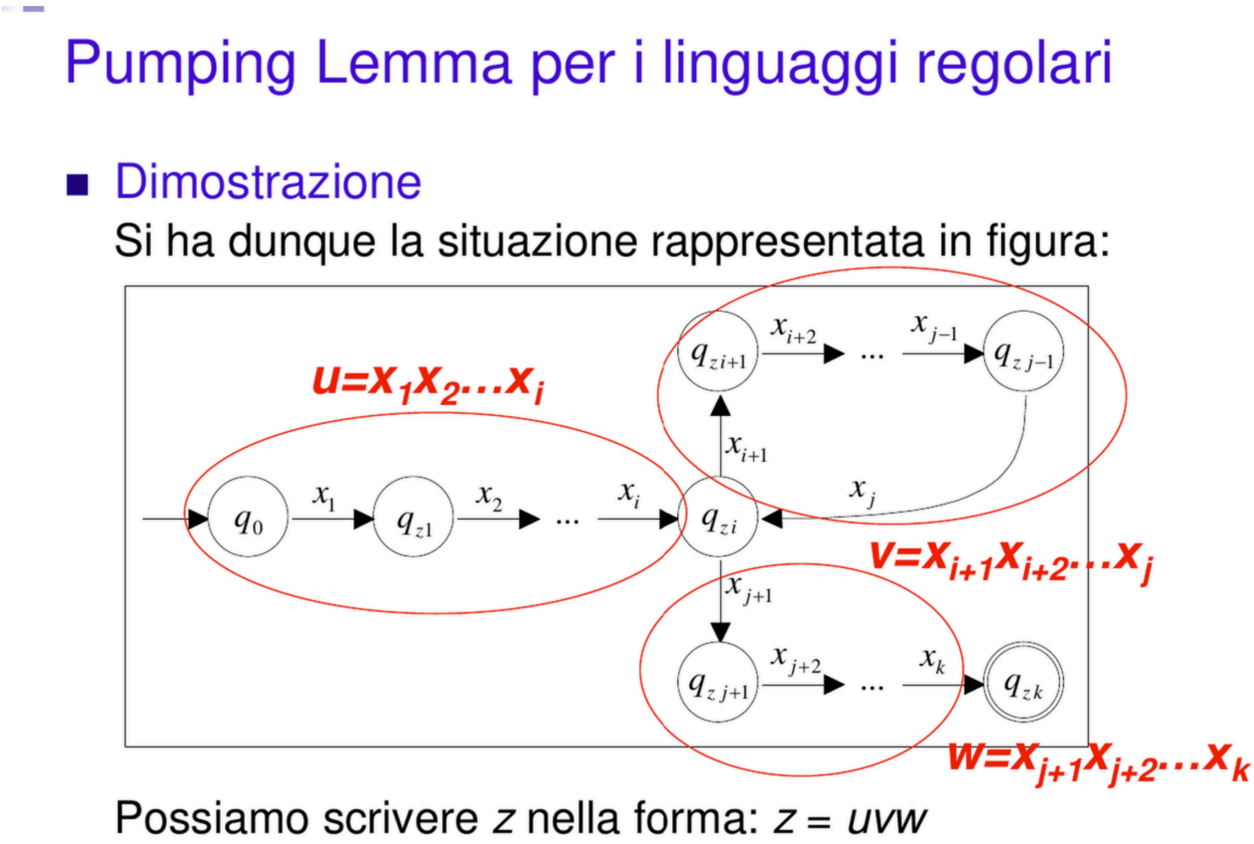
  Sia $L = T(M)$ un linguaggio regolare con
  $M = (Q, \delta, q_0, F)$ un automa accettore a stati finiti.
  Allora $\exists n = |Q|$ t.c. $\forall z \in L,\ |z| \geq n : z = uvw$ e:
  - $|uv| \leq n$
  - $v \neq \lambda$
  - $uv^iw \in L,\ \forall i,\ i \geq 0$

- Dimostrazione

  Sia $z = x_1 x_2 \ldots x_k$, $z \in T(M)$.
  Possiamo rappresentare il comportamento dell'automa M, con ingresso z, come segue:



  Se si ha: $|z| \geq n$, nella figura devono comparire almeno $n+1$ stati ma, poiché M ha solo n stati distinti ($|Q| = n$) almeno una stato tra $q_0, q_{z1}, q_{z2}, \ldots, q_{zk}$ deve comparire due volte.
  Supponiamo che si abbia $q_{zi} = q_{zj}$, $i < j$.



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:

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