# Proprietà dei linguaggi regolari

| | |
|---|---|
| Materia | Fondamenti di Informatica: Linguaggi Formali |
| Data | @April 29, 2024 |

## Proprietà di chiusura rispetto alla concatenazione

Se $L_1, L_2$ sono linguaggi regolari, allora $L_1 \cdot L_2$ è regolare.

Costruisco gli automi che riconoscono le stringhe di $L_1$ e di $L_2$. Ammesso che il primo automa ha un unico stato finale, potremmo idealmente far coincidere lo stato finale di $L_1$ con lo stato iniziale di $L_2$. Tuttavia, c'è il rischio che la computazione della stringa $x_2$ nella concatenazione $x_1x_2$ venga eseguita parzialmente nell'automa che riconosce $L_1$ (ad esempio, se vi è un cammino che ritorna ad $\mathbb{A}_1$). Quindi invece di far coincidere lo stato finale di $L_1$ con lo stato iniziale di $L_2$, possiamo fare $\epsilon-$**transizioni dallo stato finale di $L_1$ allo stato iniziale di $L_2$**. Questa soluzione funziona anche per più di uno stato finale di $L_1$. Quindi otteniamo un automa che ha come stato iniziale quello di $L_1$ e come stati finali quelli di $L_2$.

NB: la $\epsilon-$transizione è una transizione in cui la testina dell'automa rimane ferma e non consuma nessun carattere.

## Proprietà di chiusura rispetto al complemento

Sia $L$ un linguaggio regolare. Allora $L^C = \Sigma^* \setminus L$ è regolare.

Se $\mathbb{A}$ è l'automa che riconosce $L$, allora avrà un insieme $F$ di stati finali. Basta che io cambio l'insieme $F$ e considero come insieme degli stati finali l'insieme $F' = Q \setminus F$. Il risultato è l'automa $\mathbb{A}'$ che riconosce le stringhe non riconosciute da $\mathbb{A}$, ossia $L^C$. Quindi:

$$\mathbb{A}^C =< Q, \Sigma, \delta, q_0, Q \setminus F >$$

Questa dimostrazione è intuitiva: dovremmo dimostrare che $\forall x \in L(\mathbb{A}) \implies x \notin L(\mathbb{A}')$.

Tuttavia, se l'automa è deterministico, la funzione $\delta$ di transizione è **totale**. Quindi dev'essere ben definita per ogni coppia stato-simbolo. Quindi qualunque sia la stringa in input, dobbiamo essere in grado di completare la computazione della stringa. L'automa che costruiamo quindi **dev'essere deterministico:** infatti se consideriamo la costruzione dell'automa non deterministico, una stessa stringa potrebbe giungere in uno stato finale e non finale, e quindi anche l'automa complementare riconoscerebbe la stringa.

### Proprietà di chiusura rispetto all'intersezione

Se $L_1, L_2$ sono due linguaggi regolari, allora $L_1 \cap L_2$ è regolare.

Possiamo usare le **leggi di De Morgan**: infatti $L_1 \cap L_2 = \overline{(\overline{L}_1 \cup \overline{L}_2)}$. E visto che il complementare di un linguaggio regolare è regolare e la stessa cosa vale per l'unione, allora anche l'intersezione sarà regolare.

In realtà si può anche dimostrare con un automa che parte da una coppia di stati iniziali $(q_0, q_0')$, e ad ogni passo giunge ad una coppia di stati $(q_i, p_i)$, dove $q_i$ è uno stato dell'automa $\mathbb{A}_1$, e $p_i$ è uno stato dell'automa $\mathbb{A}_2$. L'insieme $F$ degli stati finali sarà costituito da tutte le coppie di stati finali.

### Proprietà di chiusura rispetto alla chiusura riflessiva

Sia $L$ un linguaggio regolare. Allora $L^*$ è regolare.

Si potrebbe dire che $L^*$ è l'unione infinita di linguaggi regolari, ma in realtà **le proprietà di chiusura valgono per operazioni finite**. Quindi è necessario costruire l'automa. L'idea è semplice: basta far partire dagli stati finali $\epsilon-$ **transizioni verso lo stato iniziale**: in questo modo è come ricominciare la computazione.

Tutte le costruzioni che abbiamo fatto per dimostrare le proprietà di chiusura si chiamano **costruzioni di Thompson**.

# Pumping Lemma

Il **Pumping Lemma** dà una condizione necessaria ma non sufficiente per i linguaggi regolari, ossia una proprietà del tipo "se $L$ è regolare, allora…". Quindi non posso dimostrare che un linguaggio è regolare perché soddisfa il pumping lemma, ma permette di distinguere linguaggi regolari e non regolari, che è necessario, altrimenti dovrei dimostrare che non esiste l'automa che riconosce il linguaggio.

Se $L$ è un linguaggio regolare, esiste un numero $n$ tale che se esiste $x \in L$ tale che la sua lunghezza è superiore ad $n$, la stringa $x$ può essere divisa in tre parti $uvw$, tale che $v$ può essere ripetuta quante volte voglio e ottenere sempre una

stringa appartenente al linguaggio. Formalmente l'enunciato è il seguente:
Sia
$L$ un linguaggio regolare. Allora $\exists n > 0 : z \in L, |n| \geq n \implies$ possiamo scrivere $z = uvw$, con $u, v, w \in \Sigma^*$, con $|uv| \leq n, v \neq \epsilon$ e $uv^iw \in L, \forall i \geq 0$.
La condizione $v \neq \epsilon$ è una condizione importante perché **ogni stringa può essere divisa in $uw$, e quindi il Lemma non permette di distinguere linguaggi regolari da non regolari**. Inoltre, la condizione $|uv| \leq n$ sottolinea che posso prendere una sezione arbitrariamente piccola della stringa.

Informalmente, potremmo dire che un automa che riconosce il linguaggio $L$, se ho una stringa $x$ di $L$ e un numero $n$ tale che $|x| > n$, e considero $n = |Q|$, allora nella computazione della stringa dovrò per forza ritornare in uno degli stati precedenti. Pertanto esisterà un ciclo all'interno del grafo di computazione della stringa. Questo ciclo consiste nella parte $v$ centrale non vuota della stringa.

## Dimostrazione formale

Sia $\mathbb{A} =< Q, \Sigma, \delta, q_0, F >$ un automa a stati finiti deterministico che riconosce $L$. Sia $|Q| = n$. Vogliamo dimostrare che la costante $n$ è il numero che soddisfa la condizione dell'enunciato. Sia $z \in L, |z| = k \geq n$. Sia $z_h$ il **prefisso di $z$ di lunghezza** $h$, con $0 \leq h \leq k$. Sia $q_{i_0}, ..., q_{i_k}$ la successione degli stati di $\mathbb{A}$ raggiunti durante la computazione su $z$, ossia $q_{i_0} = q_0$, e quindi $q_{i_h} = \overline{\delta}(q_0, z_h)$.
La successione è costituita da $k+1$ stati ($k$ stati più lo stato $q_0$). Ma $k \geq n$, quindi $k + 1 > n$, quindi almeno uno stato $q \in Q$ deve ripetersi. Esistono quindi due indici $s, t$ tali che $0 < s < t < k$ e $q_{i_s} = q_{i_t}$. Ovvero, $\overline{\delta}(q_0, z_s) = \overline{\delta}(q_0, z_t)$. Tuttavia, possiamo semplicemente scrivere $t \leq n$, in quanto leggendo i primi $n$ simboli della stringa avrò già una sequenza di $k + 1 > n$ stati.
Sia
$u = z_s$, $uv = z_t$, $uvw = z$. Poiché $t \leq n$, $|uv| = |z_t| = t \leq n$, quindi soddisfa la prima condizione dell'enunciato. Inoltre, sappiamo che $|u| = |z_s| = s < t = |z_t| = |uv|$, quindi $v \neq \epsilon$. Per dimostrare che $uv^iw \in L \ \ \forall i \geq 0$. Lo dimostreremo per induzione su $i$.
Il caso base è
$i = 0$, quindi $uv^iw = uw \in L$. Ma ciò accade soltanto se $\overline{\delta}(q_0, uw) \in F$. Una proprietà che possiamo usare è quella di spezzare la stringa in due parti, e quindi considerare:

$$\overline{\delta}(\overline{\delta}(q_0, u), w)$$

che è ovvio per definizione di $\overline{\delta}$. Quindi calcoliamo $\overline{\delta}(q_0, u) = \overline{\delta}(q_0, z_s) = q_{i_s} = q_{i_t} = \overline{\delta}(q_0, z_t) = \overline{\delta}(q_0, uv)$. Quindi quello che stiamo calcolando è:

$$\overline{\delta}(q_0, uvw)$$

ma $uvw = z$, e sappiamo che $z \in L$, pertanto il caso base è dimostrato.

Il passo induttivo è $P(i) \implies P(i+1)$. Dobbiamo quindi dimostrare che:

$$\overline{\delta}(q_0, uv^{i+1}w) \in F$$

che possiamo scrivere come:

$$\overline{\delta}(q_0, uv^ivw) = \overline{\delta}(\overline{\delta}(q_0, uv^i), vw) = \overline{\delta}(\overline{\delta}(q_0, z_t)v^iw) = \overline{\delta}(\overline{\delta}(q_0, z_s)v^iw) = \overline{\delta}(q_0, uv^iw) \in F$$

E il teorema è dimostrato.