Proprietà dei linguaggi regolari

Proprietà dei Linguaggi regolari

- **Pumping Lemma.**
  Ogni linguaggio regolare soddisfa il pumping lemma. Se qualcuno vi presenta un falso linguaggio regolare, l'uso del pumping lemma mostrerà una contraddizione.
- **Proprietà di chiusura.**
  Come costruire automi da componenti usando delle operazioni, ad esempio dati L e M possiamo costruire un automa per $L \cap M$.
- **Proprietà di decisione.**
  Analisi computazionale di automi, cioè quanto costa controllare varie proprietà, come l'equivalenza di due automi.
- **Tecniche di minimizzazione.**
  Possiamo risparmiare costruendo automi più piccoli.

Il Pumping Lemma, informalmente

- Supponiamo che $L_{01} = \{0^n1^n : n \geq 1\}$ sia regolare.
- Allora deve essere accettato da un qualche DFA A, con, ad esempio, k stati.
- Supponiamo che A legga 0^k. Avrà le seguenti transizioni:

$$\begin{array}{ll} \epsilon & p_0 \\ 0 & p_1 \\ 00 & p_2 \\ \ldots & \ldots \\ 0^k & p_k \end{array}$$

$\Rightarrow \exists i < j : p_i = p_j$

- Chiamiamo q questo stato.

- Adesso possiamo ingannare A:
  - Se $\hat{\delta}(q, 1^i) \in F$ l'automa accetterà, sbagliando, 0^j1^i.
  - Se $\hat{\delta}(q, 1^i) \notin F$ l'automa rifiuterà, sbagliando, 0^i1^i.
- Quindi L_{01} non può essere regolare.

Teorema 4.1: Il Pumping Lemma per Linguaggi Regolari

Sia L un linguaggio regolare.
Allora $\exists n, \forall w \in L : |w| \geq n \Rightarrow w = xyz$ tale che:

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. $\forall k \geq 0,\ xy^k z \in L$

Intuitivamente

- Esiste una costante n dipendente dal linguaggio L tale che tutte le stringhe di lunghezza $\geq n$ possono essere scomposte in un dato modo
- È sempre possibile scegliere una stringa *non vuota* y da replicare, ovvero **cancellare** o **ripetere** k volte, pur rimanendo all'interno del linguaggio L

Ovvero un cammino più lungo di n deve contenere un ciclo ed è il ciclo a pompare.

Dimostrazione

- Supponiamo che L sia regolare.
- Allora L è riconosciuto da un DFA A con, ad esempio, n stati $Q = \{q_1, \dots, q_{n-1}\}$.
- Prendiamo come costante il valore n, e consideriamo una generica stringa $w \in L$ più lunga di n. Avremo quindi $w = a_1 a_2 \dots a_m \in L$ con $m \geq n$.

Dimostrazione (2)

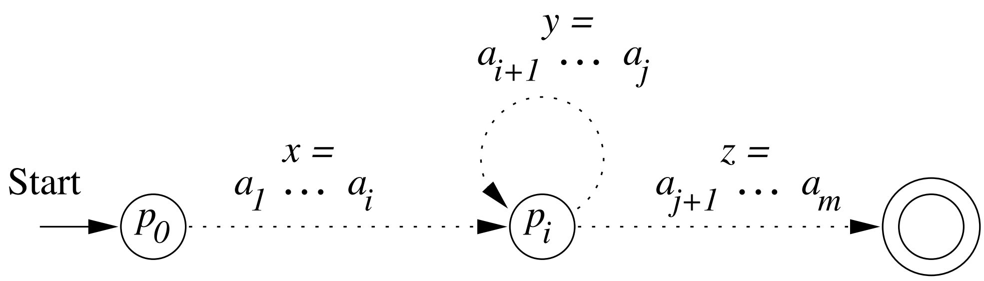
Chiamiamo p_i, per $i \in \{0, \dots, m\}$, lo stato in cui si trova l'automa A dopo avere esaminato $a_1 a_2 \dots a_i$ a partire dallo stato iniziale q_0. Formalmente, utilizzando la funzione di transizione estesa:

- $p_0 = \hat{\delta}(q_0, \epsilon) = q_0$
- $p_i = \hat{\delta}(q_0, a_1 a_2 \cdots a_i)$.
- Dato che ci sono n stati distinti, gli $n+1$ stati p_i non possono essere tutti distinti: $\Rightarrow \exists i < j : p_i = p_j$

Dimostrazione (3)

Ora $w = xyz$, dove

1. $x = a_1 a_2 \cdots a_i$ (x porta a p_i la prima volta)
2. $y = a_{i+1} a_{i+2} \cdots a_j$ (y porta da p_i a p_i, dato che p_i e p_j coincidono)
3. $z = a_{j+1} a_{j+2} \ldots a_m$ (z conclude w)



Il pumping lemma
Proprietà di chiusura
Proprietà di decisione

Dimostrazione (4)

Notiamo che

- x può essere vuota (per $i = 0$) e anche z può essere vuota (per $j = n = m$). Invece
- $y \neq \epsilon$: la stringa y non è vuota, dato che $i < j$
- $|xy| \leq n$ dato che gli stati $p_0, \ldots, p_{j-1}$ sono tutti distinti (basta considerare il minimo indice che si ripete)

Data la forma dell'automa, è chiaro che, eseguendo $k \geq 0$ cicli in p_i, l'automa accetta ogni stringa xy^kz.

- per $k = 0$, l'automa passa dallo stato iniziale $q_0 = p_0$ a $p_i = p_j$ su input x. Allora passa da p_i allo stato accettante con input z. Quindi accetta xz.
- per $k > 0$, A va da q_0 a p_i su x, cicla su p_i per k volte su input y^k e passa allo stato accettante per z e quindi accetta xy^kz

Quindi per $k \geq 0$, abbiamo che $xy^kz \in L(A)$

Esempio

- Sia $L_{01} = \{0^n 1^n\}$ 01 il linguaggio delle stringhe formate da un certo numero di 0, seguiti dallo stesso numero di 1.
- Supponiamo che L_{01} sia regolare. Allora $w = 0^n 1^n \in L$ la stringa per n
- Per il pumping lemma, $w = xyz$, $|xy| \leq n$, $y \neq \epsilon$ e $xy^k z \in L_{01}$
  - $y = 0^h 1^j$: è chiaro che ripetendo la stringa k volte, gli 0 e gli 1 vengono mescolati; quindi la stringa ottenuta non sta nel linguaggio
  - $y = 0^h$: è formata solo da 0 (o analogamente $y = 1^h$ è formata solo da 1),

$$w = \underbrace{000\ldots}_{x}\underbrace{\ldots 000}_{y}\underbrace{0111\ldots 11}_{z}$$

Se consideriamo xz ha meno 0 che 1 e non sta nel linguaggio

Esempio

- Sia L_{eq} il linguaggio delle stringhe con ugual numero di zeri e di uni.
- Supponiamo che L_{eq} sia regolare. Allora $w = 0^n 1^n \in L_{eq}$.
- Per il pumping lemma, $w = xyz$, $|xy| \leq n$, $y \neq \epsilon$ e $xy^k z \in L_{eq}$

$$w = \underbrace{000\cdots}_{x}\underbrace{\cdots 0}_{y}\underbrace{0111\cdots 11}_{z}$$

- In particolare, $xz \in L_{eq}$, ma xz ha meno zeri di uni.

Esempio

Supponiamo che $L_{pr} = \{1^p : p \text{ è primo}\}$ sia regolare.
Sia n dato dal pumping lemma.
Scegliamo un numero primo $p \geq n+2$.

$$w = \overbrace{\underbrace{111\cdots\cdots}_{x}\underbrace{1\,1111}_{\substack{y\\|y|=m}}\underbrace{\cdots 11}_{z}}^{p}$$

Ora $xy^{p-m}z \in L_{pr}$
$|xy^{p-m}z| = |xz| + (p-m)|y| = p - m + (p-m)m = (1+m)(p-m)$
che non è primo a meno che uno dei fattori non sia 1.

- $y \neq \epsilon \Rightarrow 1 + m > 1$
- $m = |y| \leq |xy| \leq n, \quad p \geq n+2$

$\Rightarrow p - m \geq n + 2 - n = 2.$

Nota Bene

Il pumping lemma fornisce soltanto una condizione necessaria affinché un linguaggio sia regolare.

Proprietà di chiusura dei linguaggi regolari

Siano L e M due linguaggi regolari. Allora i seguenti linguaggi sono regolari:

- *Unione:* $L \cup M$
- *Intersezione:* $L \cap M$
- *Complemento:* $\overline{N}$
- *Differenza:* $L \setminus M$
- *Inversione:* $L^R = \{w^R : w \in L\}$
- *Chiusura:* L^*.
- *Concatenazione:* $L.M$

Il pumping lemma
Proprietà di chiusura
Proprietà di decisione

Chiusura rispetto a unione e complemento

Teorema 4.4. Per ogni coppia di linguaggi regolari L e M, $L \cup M$ è regolare.
Dimostrazione. Sia $L = L(E)$ e $M = L(F)$. Allora $L(E + F) = L \cup M$ per definizione.

Teorema 4.5. Se L è un linguaggio regolare su Σ, allora che $\overline{L} = \Sigma^* \setminus L$ è regolare.
Dimostrazione. Sia L riconosciuto da un DFA

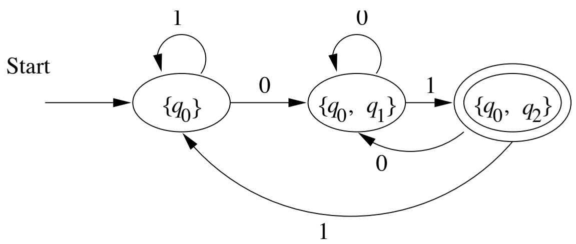
$$A = (Q, \Sigma, \delta, q_0, F).$$

Sia $B = (Q, \Sigma, \delta, q_0, Q \setminus F)$. Allora $L(B) = \overline{L}$.

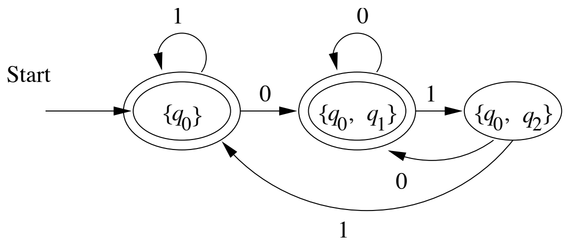
Proprietà dei linguaggi regolari

Esempio

Sia L riconosciuto dal DFA qui sotto:



Allora $\overline{L}$ è riconosciuto da:



Domanda: Quali sono le espressioni regolari per L e $\overline{L}$?

Chiusura rispetto all'intersezione

Teorema 4.8. Se L e M sono regolari, allora anche $L \cap M$ è regolare.

Dimostrazione 1. Per la legge di De Morgan, $L \cap M = \overline{\overline{L} \cup \overline{M}}$. Sappiamo già che i linguaggi regolari sono chiusi sotto il complemento e l'unione.

Chiusura rispetto all'intersezione: un'altra dimostrazione

Se L e M sono regolari, allora anche $L \cap M$ è regolare.
Dimostrazione 2. Sia L il linguaggio di

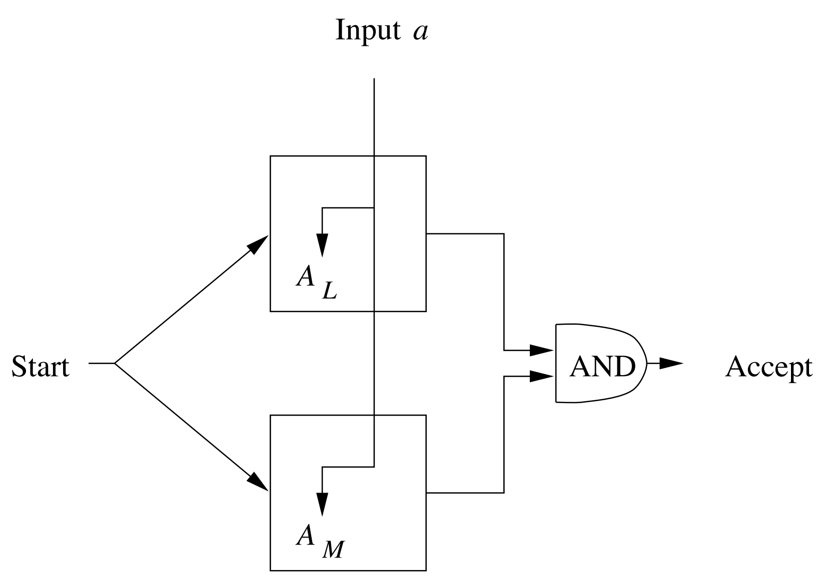
$$A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$$

e M il linguaggio di

$$A_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$$

Supponiamo senza perdita di generalità che entrambi gli automi siano deterministici.
Costruiremo un automa che simula A_L e A_M in parallelo, e accetta se e solo se sia A_L che A_M accettano.

Se A_L va dallo stato p allo stato s leggendo a, e A_M va dallo stato q allo stato t leggendo a, allora $A_{L\cap M}$ andrà dallo stato (p,q) allo stato (s,t) leggendo a.



Formalmente

$$A_{L\cap M} = (Q_L \times Q_M, \Sigma, \delta_{L\cap M}, (q_L, q_M), F_L \times F_M),$$

dove

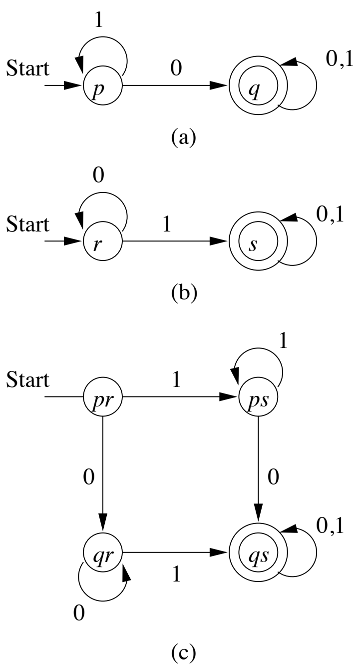
$$\delta_{L\cap M}((p,q),a) = (\delta_L(p,a), \delta_M(q,a))$$

Si può mostrare per induzione su $|w|$ che

$$\hat{\delta}_{L\cap M}((q_L,q_M),w) = \left(\hat{\delta}_L(q_L,w), \hat{\delta}_M(q_M,w)\right)$$

Esempio

$(c) = (a) \times (b)$



Chiusura rispetto alla differenza

Teorema 4.10 Se L e M sono linguaggi regolari, allora anche $L \setminus M$ è regolare.
Dimostrazione. Osserviamo che $L \setminus M = L \cap \overline{M}$. Sappiamo già che i linguaggi regolari sono chiusi sotto il complemento e l'intersezione.

Il pumping lemma
Proprietà di chiusura
Proprietà di decisione

Chiusura rispetto al "reverse"

Teorema 4.11 Se L è un linguaggio regolare, allora anche L^R è regolare.
Dimostrazione 1: Sia L riconosciuto da un FA A. Modifichiamo A per renderlo un FA per L^R:

1. Giriamo tutti gli archi.
2. Rendiamo il vecchio stato iniziale l'unico stato finale.
3. Creiamo un nuovo stato iniziale p_0, con $\delta(p_0, \epsilon) = F$ (i vecchi stati finali).

Proprietà dei linguaggi regolari

Chiusura rispetto al "reverse": un'altra dimostrazione

Se L è un linguaggio regolare, allora anche L^R è regolare.
Dimostrazione 2: Sia L descritto da un'espressione regolare E. Costruiremo un'espressione regolare E^R, tale che $L(E^R) = (L(E))^R$.
Procediamo per induzione strutturale su E.
Base: Se E è ϵ, $\emptyset$, o $\mathbf{a}$, allora $E^R = E$.
Induzione:

1. $E = F + G$. Allora $E^R = F^R + G^R$
2. $E = F.G$. Allora $E^R = G^R.F^R$
3. $E = F^*$. Allora $E^R = (F^R)^*$

Proprietà di decisione

1. Convertire tra diverse rappresentazioni dei linguaggi regolari.
2. È $L = \emptyset$?
3. È $w \in L$?
4. Due descrizioni definiscono lo stesso linguaggio?

Da NFA a DFA

- Supponiamo che un ϵ-NFA abbia n stati.
- Per calcolare ECLOSE(p) seguiamo al più n^2 archi. Lo facciamo per n stati, quindi in totale sono n^3 passi.
- Il DFA ha 2^n stati, per ogni stato S e ogni $a \in \Sigma$ calcoliamo $\delta_D(S, a)$ in n^3 passi. In totale abbiamo $O(n^3 2^n)$ passi.
- Se calcoliamo δ solo per gli stati raggiungibili, dobbiamo calcolare $\delta_D(S, a)$ solo s volte, dove s è il numero di stati raggiungibili. In totale: $O(n^3 s)$ passi.

Da DFA a NFA

Dobbiamo solo mettere le parentesi graffe attorno agli stati.
Totale: $O(n)$ passi.

Da FA a espressione regolare

Dobbiamo calcolare n^3 cose di grandezza fino a 4^n. Totale: $O(n^3 4^n)$.
L'FA può essere un NFA. Se prima vogliamo convertire l'NFA in un DFA, il tempo totale sarà doppiamente esponenziale.

Da espressioni regolari a FA

Possiamo costruire un albero per l'espressione in n passi.
Possiamo costruire l'automa in n passi.
Eliminare le ϵ-transizioni ha bisogno di $O(n^3)$ passi.
Se si vuole un DFA, potremmo aver bisogno di un numero esponenziale di passi.

Controllare se un linguaggio è vuoto

- $L(A) \neq \emptyset$ per FA A se e solo se uno stato finale è raggiungibile dallo stato iniziale in in A. Totale: $O(n^2)$ passi.
- Oppure, possiamo guardare un'espressione regolare E e vedere se $L(E) = \emptyset$, considerando tutti i casi:
  - $E = F + G$. Allora $L(E)$ è vuoto se e solo se sia $L(F)$ che $L(G)$ sono vuoti.
  - $E = F.G$. Allora $L(E)$ è vuoto se e solo se o $L(F)$ o $L(G)$ sono vuoti.
  - $E = F^*$. Allora $L(E)$ non è mai vuoto, perché $\epsilon \in L(E)$.
  - $E = \epsilon$. Allora $L(E)$ non è vuoto.
  - $E = \mathbf{a}$. Allora $L(E)$ non è vuoto.
  - $E = \emptyset$. Allora $L(E)$ è vuoto.

Controllare l'appartenenza ad un linguaggio

- Per controllare se $w \in L(A)$ per DFA A, simuliamo A su w. Se $|w| = n$, questo prende $O(n)$ passi.
- Se A è un NFA e ha s stati, simulare A su w prende $O(ns^2)$ passi.
- Se A è un ϵ-NFA e ha s stati, simulare A su w prende $O(ns^3)$ passi.
- Se $L = L(E)$, per l'espressione regolare E di lunghezza s, prima convertiamo E in un ϵ-NFA con $2s$ stati. Poi simuliamo w su questo automa, in $O(ns^3)$ passi.

Pumping Lemma per dimostrare la non regolarità

Non soddisfare il Pumping Lemma significa invertire l'implicazione, utilizzando il fatto che A implica B equivale a $\overline{B}$ implica $\overline{A}$.
Utilizzando un po' di manipolazione algebrica degli operatori logici possiamo passare quindi dalla formula:

$$\text{L reg.} \Rightarrow \left(\exists n \forall w \in L |w| \geq n \Rightarrow \left(\exists x, y, z \; t.c. \begin{cases} w = xyz \\ |xy| \leq n \\ y \neq \epsilon \end{cases} \wedge \; \forall k : xy^k z \in L \right) \right)$$

alla formula

$$\left(\forall n \exists w \in L |w| \geq n \wedge \left(\forall x, y, z \; t.c. \begin{cases} w = xyz \\ |xy| \leq n \\ y \neq \epsilon \end{cases} \Rightarrow \; \exists k : xy^k z \notin L \right) \right) \Rightarrow \overline{\text{L reg.}}$$

PL: una condizione necessaria per la regolarità

Ci sono linguaggi che soddisfano il pumping lemma senza essere tuttavia regolari, ad esempio:

$$L = \{ab^j c^j | j \geq 0\} \cup \{ a^i b^j c^k | i, j, k \geq 0 \wedge i \neq 1\}$$

Per dimostrare la non regolarità dobbiamo usare altri sistemi.

Pumping Lemma per la non regolarità: qualche esercizio

Dimostrare che i seguenti linguaggi non sono regolari.

- L'insieme delle stringhe di parentesi bilanciate.
- $L = \{0^n 1^m | n \leq m\}$
- $L = \{0^n 1^m 2^n | n, m \text{ interi}\}$
- $L = \{0^{n^2} | n \text{ intero}\}$
- $L = \{ww | w \in \{0, 1\}^*\}$
- $L = \{ww^R | w \in \{0, 1\}^*\}$
- $L = \{0^i 1^j | mcd(i, j) = 1\}$