Proprietà dei linguaggi regolari

Corso di Fondamenti di Informatica - modulo 1
Corso di Laurea in Informatica
Università di Roma "Tor Vergata"
a.a. 2020-2021

Giorgio Gambosi

1 Proprietà di chiusura

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 $\diamond$:

- se $\diamond$ è unaria, $\diamond L$ è un linguaggio regolare per ogni L regolare
- se $\diamond$ è binaria, $L_1 \diamond L_2$ 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 $\mathcal{A}_1 = \langle \Sigma_1, Q_1, \delta_{N_1}, q_{0_1}, F_1 \rangle$ e $\mathcal{A}_2 = \langle \Sigma_2, Q_2, \delta_{N_2}, q_{0_2}, F_2 \rangle$, due ASFD che accettano L_1 e L_2. Costruiamo da $\mathcal{A}_1$ e $\mathcal{A}_2$ un automa $\mathcal{A} = \{\Sigma, Q, \delta_N, q_0, F\}$ che riconosce il linguaggio $L_1 \cup L_2$.

Chiusura dei linguaggi regolari: unione

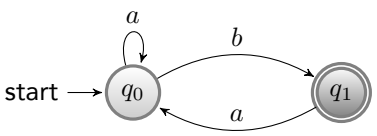
Costruzione:

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 $\mathcal{A}_1$, $\mathcal{A}_2$ riconosce anche la stringa vuota.
4. La funzione di transizione δ_N è definita come:

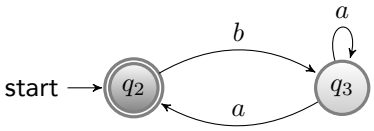
$$\begin{aligned} \delta_N(q,a) &= \delta_{N_1}(q,a) \text{ se } q \in Q_1, a \in \Sigma_1 \\ \delta_N(q,a) &= \delta_{N_2}(q,a) \text{ se } q \in Q_2, a \in \Sigma_2 \\ \delta_N(q_0,a) &= \delta_{N_1}(q_{0_1},a) \cup \delta_{N_2}(q_{0_2},a), a \in \Sigma. \end{aligned}$$

Esempio

$\mathcal{A}_1$

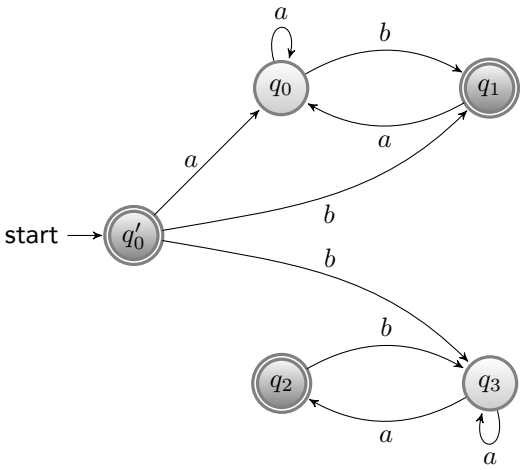


$\mathcal{A}_2$



Esempio

$\mathcal{A}$



Chiusura dei linguaggi regolari: complemento

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

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

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

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

Infatti, ogni stringa che porta l'automa $\mathcal{A}$ in uno stato finale porta l'automa $\overline{\mathcal{A}}$ in uno stato non finale e, viceversa, ogni stringa che porta $\overline{\mathcal{A}}$ in uno stato finale porta $\mathcal{A}$ 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 $\mathcal{A}_1 = \langle \Sigma_1, Q_1, \delta_{N_1}, q_{0_1}, F_1 \rangle$ e $\mathcal{A}_2 = \langle \Sigma_2, Q_2, \delta_{N_2}, q_{0_2}, F_2 \rangle$, due ASFD che accettano L_1 e L_2. Costruiamo da $\mathcal{A}_1$ e $\mathcal{A}_2$ un automa $\mathcal{A} = \{\Sigma, Q, \delta_N, q_0, F\}$ che riconosce il linguaggio $L_1 \circ L_2$.

Chiusura dei linguaggi regolari: concatenazione

Costruzione:

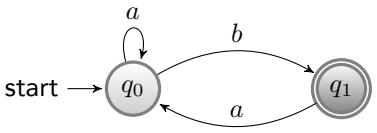
1. $\Sigma = \Sigma_1 \cup \Sigma_2$;
2. $Q = Q_1 \cup Q_2$;
3. $F = \begin{cases} F_2 & \text{se } \varepsilon \notin L(\mathcal{A}_2), \\ F_1 \cup F_2 & \text{altrimenti;} \end{cases}$
4. $q_0 = q_{01}$;

5. δ_N è definita come:

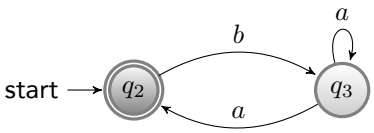
$$\begin{aligned}\delta_N(q,a) &= \delta_1(q,a), \forall q \in Q_1 - F_1, a \in \Sigma_1\\ \delta_N(q,a) &= \delta_1(q,a) \cup \delta_2(q_{02},a), \forall q \in F_1, a \in \Sigma\\ \delta_N(q,a) &= \delta_2(q,a), \forall q \in Q_2, a \in \Sigma_2.\end{aligned}$$

Esempio

$\mathcal{A}_1$

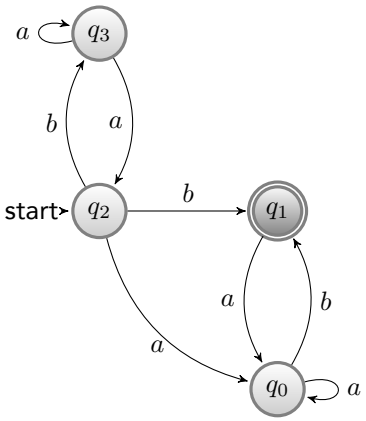


$\mathcal{A}_2$



Esempio

$\mathcal{A}$



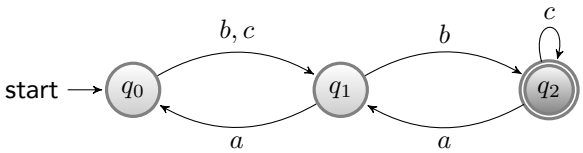
Chiusura dei linguaggi regolari: iterazione

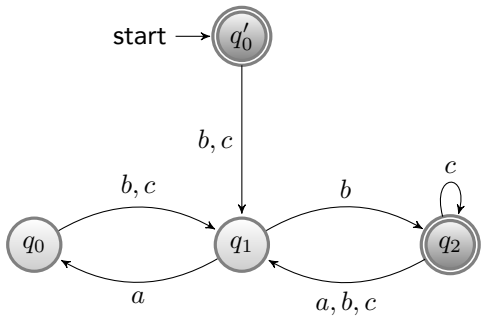
Dato un linguaggo regolare L, la sua iterazione L^* è un linguaggio regolare.

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





2 Decidibilità di predicati

Decidibilità di predicati su LR

Teorema 1. *È possibile decidere se il linguaggio accettato da un dato automa a stati finiti è vuoto.*

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 $\mathcal{A}$ con $n = |Q|$ stati e $s = |\Sigma|$ simboli di input, per decidere se $L(\mathcal{A}) = \Lambda$ basta verificare se $\mathcal{A}$ 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 $\mathcal{A}$, $L(\mathcal{A})$ 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 2. *È possibile decidere se il linguaggio accettato da un dato automa a stati finiti è finito.*

Decidibilità di predicati su LR

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

Decidibilità di predicati su LR

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

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

- assumiamo, senza perdere generalità, che z sia una stringa di lunghezza minima tra tutte stringhe in $L(\mathcal{A})$ 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, $uw \in L(\mathcal{A})$ e $|uw| < |uvw|$: data l'ipotesi che z abbia lunghezza minima tra tutte le stringhe di $L(\mathcal{A})$ 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 $L(\mathcal{A})$ di lunghezza minima tra tutte quelle lunghe almeno n non può essere lunga di $2n$ o più, e quindi la proprietà è vera

In conclusione:

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

Dato un automa $\mathcal{A}$ con n stati e s simboli di input, per decidere se $L(\mathcal{A})$ è infinito basta verificare se $\mathcal{A}$ 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.

Osservazione: dato il grafo di transizione di $\mathcal{A}$, $L(\mathcal{A})$ 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.

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

Teorema 3. *Il problema dell'equivalenza di due linguaggi regolari è decidibile.*

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.