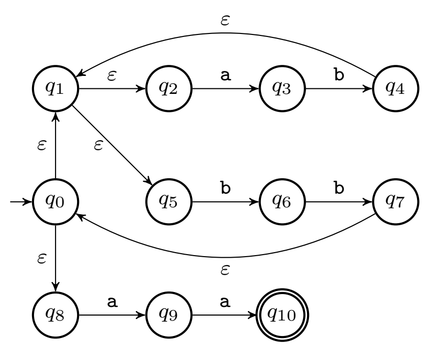
Automi e Linguaggi **(M. Cesati)**

Facoltà di Ingegneria, Università degli Studi di Roma Tor Vergata

Compito scritto del 24 gennaio 2023

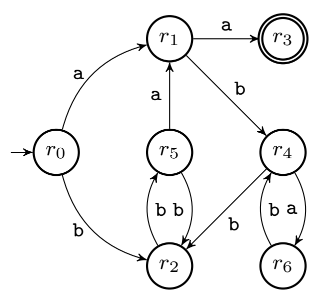
Esercizio 1 [6] Determinare un automa deterministico che riconosca il linguaggio generato dalla espressione regolare $((\texttt{ab})^*\texttt{bb})^*\texttt{aa}$.

Soluzione: L'esercizio si può risolvere in modo totalmente meccanico derivando innanzi tutto un NFA dalla espressione regolare, e successivamente trasformando lo NFA in un DFA. Applicando poche semplificazioni allo NFA derivato meccanicamente dalla espressione regolare si ottiene:

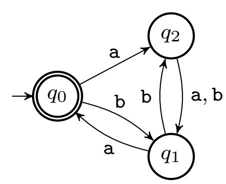


Un automa deterministico equivalente è il seguente:

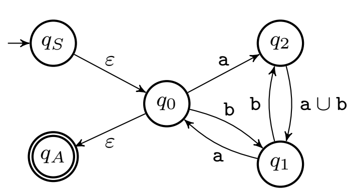
$$
\begin{aligned}
r_0 &= \{q_0, q_1, q_2, q_5, q_8\}\\
r_1 &= \{q_3, q_9\}\\
r_2 &= \{q_6\}\\
r_3 &= \{q_{10}\}\\
r_4 &= \{q_1, q_2, q_4, q_5\}\\
r_5 &= \{q_0, q_1, q_2, q_5, q_7, q_8\}\\
r_6 &= \{q_3\}
\end{aligned}
$$

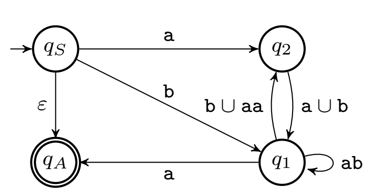


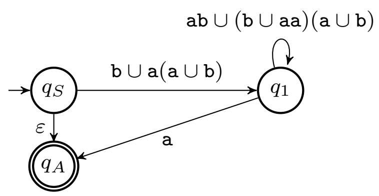
Esercizio 2 [6] Determinare una espressione regolare per il linguaggio riconosciuto dal seguente automa deterministico:

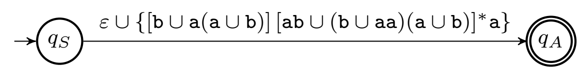


Soluzione: Trasformiamo l'automa in un GNFA aggiungendo gli stati q_S e q_A, e rimuoviamo nell'ordine i nodi q_0, q_2 e q_1:









Una espressione regolare che genera il linguaggio riconsciuto dall'automa è quindi:

$$\varepsilon \cup \{[\mathtt{b} \cup \mathtt{a}(\mathtt{a} \cup \mathtt{b})]\, [\mathtt{ab} \cup (\mathtt{b} \cup \mathtt{aa})(\mathtt{a} \cup \mathtt{b})]^* \mathtt{a}\}.$$

Cambiando l'ordine di eliminazione dei nodi si ottengono espressioni regolari diverse ma equivalenti. Ad esempio:

$$\begin{array}{rl} \{[\mathtt{b} \cup \mathtt{a}(\mathtt{a} \cup \mathtt{b})][\mathtt{b}(\mathtt{a} \cup \mathtt{b})]^* \mathtt{a}\}^* & \text{(ordine: } q_2,\, q_1,\, q_0) \\ \{\mathtt{ba} \cup (\mathtt{a} \cup \mathtt{bb})[(\mathtt{a} \cup \mathtt{b})\mathtt{b}]^* (\mathtt{a} \cup \mathtt{b})\mathtt{a}\}^* & \text{(ordine: } q_1,\, q_2,\, q_0) \end{array}$$

Esercizio 3 [6.5] Si consideri il linguaggio $A = \{\, w \in \{0,1\}^+ \mid w = w_1 w_2, |w_1| = |w_2|$, e il numero di zero in w_1 è uguale al numero di uno in $w_2 \,\}$. Ad esempio, $01 \in A$, $011 \notin A$, $0110 \in A$, $1100 \in A$, $0010 \notin A$, $0011 \in A$. Il linguaggio A è regolare? Giustificare la risposta con una dimostrazione.

Soluzione: Il linguaggio A non è regolare; per dimostrarlo, supponiamo per assurdo che lo sia, e che quindi valga per esso il pumping lemma con un opportuno valore $p > 0$.

Consideriamo la stringa $s = 0^p 101^p \in A$, di lunghezza $2p + 2$. Il pumping lemma afferma che esiste una suddivisione $s = xyz$ con $|xy| \le p$ e $|y| > 0$ tale che $xy^i z \in A$ per qualsiasi $i \ge 0$. Si osservi ora che A non contiene la stringa nulla e non contiene stringhe di lunghezza dispari. Pertanto la lunghezza $|y|$ deve essere un numero pari, altrimenti la lunghezza $|xz|$ sarebbe dispari, e quindi $xy^0 z$ non potrebbe far parte di A. Quindi possiamo scrivere $|y| = 2k$ con $k > 0$. Di conseguenza, $|xy^i z| = 2p + 2 + 2k(i - 1)$.

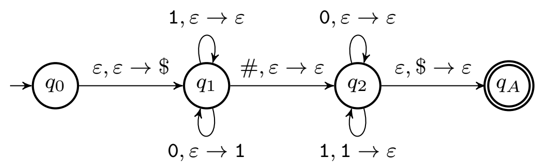
Consideriamo ora il valore particolare $i = 0$: la corrispondente stringa $xy^0 z$ ha lunghezza $|xz| = 2p + 2 - 2k$. Se $xz = w_1 w_2$, con $|w_1| = |w_2|$, allora $|w_1| = |w_2| = p + 1 - k$. Perciò necessariamente $w_2 = 1^{p+1-k}$, in quanto $k \ge 1$. D'altra parte, w_1 contiene esattamente $p - 2k + 1$ zero, e dunque dovremmo imporre

$$p - 2k + 1 = p + 1 - k,$$

che implica $k = 0$. Ma ciò è impossibile in quanto $|y| = 2k > 0$. La contraddizione deriva dall'aver supposto che per A valga il pumping lemma, e dunque A non può essere regolare.

Esercizio 4 [6.5] Si consideri il linguaggio $B = \{\, w_1 \# w_2 \mid w_1, w_2 \in \{0,1\}^*$, ed il numero di zero in w_1 è uguale al numero di uno in $w_2 \,\}$. Si osservi che w_1 e w_2 possono avere differente lunghezza. Ad esempio, $0\#1 \in B$, $0\#11 \notin B$, $11\#00 \in B$, $00\#10 \notin B$, $0\#011 \notin B$. Il linguaggio B è libero dal contesto (CFL)? Giustificare la risposta con una dimostrazione.

Soluzione: Per dimostrare che B è CFL è sufficiente esibire un PDA che riconosca tutti e soli gli elementi di B:



Solo la lettura del simbolo '#' fa transitare l'automa dallo stato q_1 allo stato q_2, Lo stato q_1 memorizza sullo stack un simbolo '1' per ogni simbolo '0' letto in ingresso, e corrispondentemente lo stato q_2 rimuove un simbolo dallo stack per ogni simbolo '1' letto in ingresso. È possibile transitare da q_2 allo stato di accettazione q_A solo se lo stack è vuoto, e l'automa accetterà la stringa solo se si entra in q_A avendo letto tutti i caratteri della stringa in ingresso.

Esercizio 5 [6] Sia $C = \{\langle M \rangle \mid M \text{ è una TM tale che } L(M) \text{ è decidibile}\}$. C è decidibile? Giustificare la risposta con una dimostrazione.

Soluzione: Il linguaggio C non è decidibile, e per dimostrarlo è sufficiente verificare che le ipotesi del Teorema di Rice sono verificate. Si consideri come proprietà P della TM il riconoscere un linguaggio decidibile. Tale proprietà è non banale: infatti una TM M_1 che accetta tutte le stringhe riconosce Σ^*, che è decidibile; d'altra parte, sappiamo che $\mathcal{A}_{\text{TM}}$ è ricorsivamente enumerabile ma non decidibile; dunque una TM M_0 che riconosce tale linguaggio non soddisfa la proprietà P. Inoltre, P è in effetti una proprietà del linguaggio riconosciuto dalla TM: infatti se due diverse TM riconoscono lo stesso linguaggio, per entrambe vale che il linguaggio è decidibile oppure no, dunque entrambe le TM soddisfano la priorità P oppure no. Poiché tutte le ipotesi del Teorema di Rice sono soddisfatte possiamo concludere immediatamente che il linguaggio C contenente codifiche di TM che soddisfano la proprietà P è indecidibile.

Esercizio 6 [9] Si consideri il linguaggio costituito dalle codifiche delle formule booleane che hanno almeno due assegnazioni di verità che soddisfano la formula. Dimostrare che tale linguaggio è NP-completo.

Soluzione: Questo problema è generalmente chiamato Double SAT, ed è una generalizzazione molto semplice del problema SAT.

Si dimostra facilmente che Double SAT è in NP. Infatti, data una qualunque istanza $\langle \Phi \rangle$ che codifica una formula booleana, un certificato per l'esistenza di una soluzione è costituito da due liste di valori di verità. Il verificatore controlla che ciascuna lista contenga esattamente un valore (vero o falso) per ciascuna variabile di Φ, e che entrambe le assegnazioni di verità soddisfino la formula Φ.

Per dimostrare che Double SAT è NP-hard consideriamo la seguente riduzione dal problema SAT: considerata una generica istanza $\langle \Phi \rangle$ di SAT, sia Φ' la formula booleana ottenuta da Φ aggiungendo una nuova variabile v e ponendo

$$\Phi' = \Phi \wedge (v \vee \overline{v}).$$

È immediato verificare che se la formula Φ è soddisfacibile allora esistono almeno due assegnazioni di verità che soddisfano Φ': la assegnazione che soddisfa Φ estesa con $v = T$ e la stessa assegnazione estesa con $v = F$. Viceversa, se la formula Φ non è soddisfacibile, allora nemmeno Φ' può essere soddisfacibile, perché la variabile v non appare nella formula Φ e quindi non può contribuire a rendere quella parte della formula Φ' soddisfacibile.

Da tutto ciò si può concludere che DOUBLE SAT è NP-completo.