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

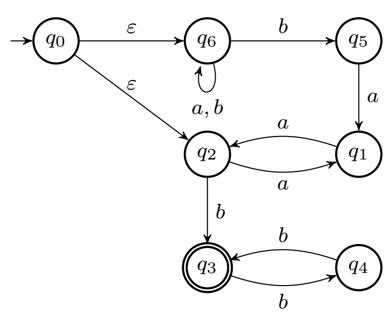
Facoltà di Ingegneria, Università degli Studi di Roma Tor Vergata

Compito scritto del 15 settembre 2021

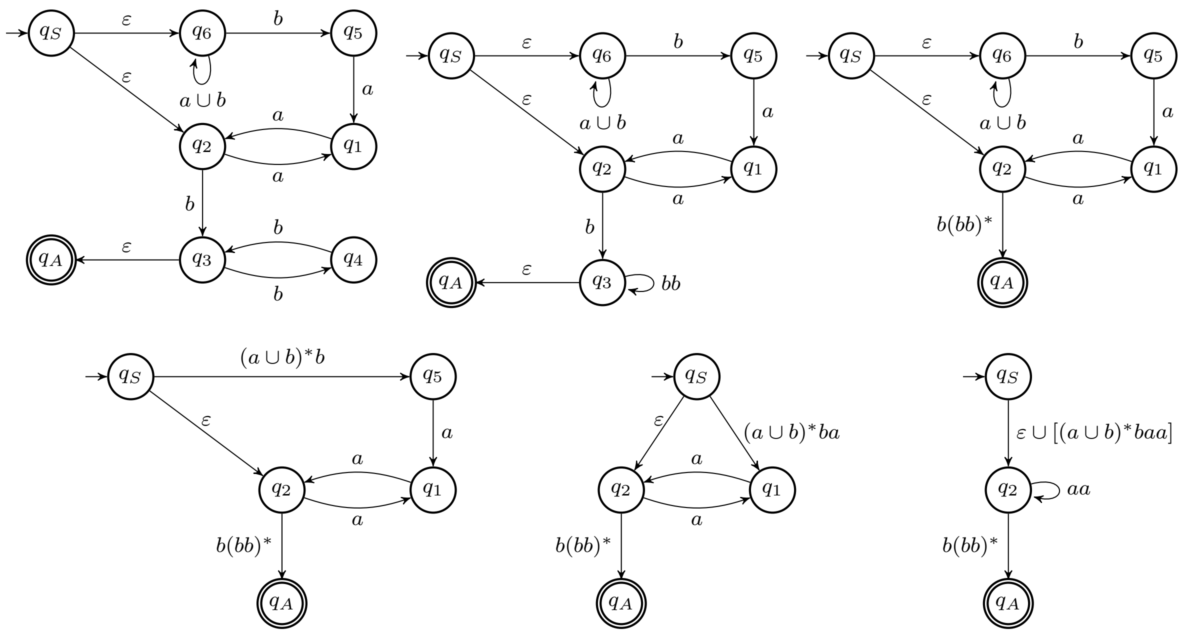
Esercizio 1 [6] Determinare una espressione regolare per il linguaggio $C = \{x \in \{a, b\}^* \mid$ l'ultima sequenza di a consecutive in x ha lunghezza pari, l'ultima sequenza di b consecutive in x ha lunghezza dispari, e l'ultimo carattere di x è $b\}$, ovvero dimostrare che tale espressione regolare non esiste. Ad esempio, $\varepsilon \notin C$, $b \in C$, $baa \notin C$ (l'ultimo carattere non è b), $abaabbb \in C$, $baaabab \notin C$.

Soluzione: Il linguaggio C è regolare, perché per riconoscere l'appartenza di una stringa in C non è necessario contare le occorrenze di simboli a e b, ma solo memorizzare la parità o disparità del numero di occorrenze nelle ultime sequenze.

Una strategia per determinare una espressione regolare per il linguaggio richiesto consiste nel determinare dapprima un NFA che riconosca il linguaggio, e successivamente derivare dal NFA una REX. Il linguaggio C può essere riconosciuto dal seguente NFA, in cui gli stati q_1, q_2, q_3 e q_4 servono a memorizzare la parità delle ultime sequenze di a e di b, mentre gli stati q_5 e q_6 sono utilizzati per "indovinare" la posizione dell'ultima sequenza di a nella stringa in ingresso. Dallo stato iniziale q_0 si può non deterministicamente saltare direttamente allo stato q_2 per considerare il caso in cui nella stringa in ingresso sia presente una unica sequenza di a (eventualmente di lunghezza zero) ed una unica sequenza finale di b.



Trasformando in GNFA e rimuovendo nell'ordine i nodi q_4, q_3, q_6, q_5, q_1 e q_2 si ottiene:



ed infine

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

Esercizio 2 [5] Si consideri la grammatica G con variabile iniziale S

$$S \to PABb \quad P \to aP \mid bP \mid \varepsilon \quad A \to aaA \mid \varepsilon \quad B \to bbB \mid \varepsilon.$$

Il linguaggio generato dalla grammatica coincide con il linguaggio C dell'esercizio precedente? Giustificare la risposta con una dimostrazione.

Soluzione: Il linguaggio generato dalla grammatica è differente dal linguaggio C dell'esercizio precedente. Per dimostrarlo è sufficiente considerare la stringa ab: essa non appartiene a C in quanto l'ultima sequenza di a ha lunghezza dispari; d'altra parte $ab \in L(G)$, come dimostrato dalla seguente derivazione:

$$S \Rightarrow P\ A\ B\ b \Rightarrow a\ P\ A\ B\ b \Rightarrow a\ A\ B\ b \Rightarrow a\ B\ b \Rightarrow a\ b.$$

Esercizio 3 [6] Si consideri la grammatica G con variabile iniziale S

$$S \to PABb \quad P \to aP \mid bP \mid \varepsilon \quad A \to aaA \mid \varepsilon \quad B \to bbB \mid \varepsilon.$$

La grammatica G è LR(1)? Giustificare la risposta con una dimostrazione.

Soluzione: La grammatica G non è LR(1). Per dimostrarlo è sufficiente costruire lo stato iniziale dell'automa DK_1:

$$\boxed{\begin{array}{ll} S \to .PABb & ab \\ P \to .aP & ab \\ P \to .bP & ab \\ P \to . & ab \end{array}}$$

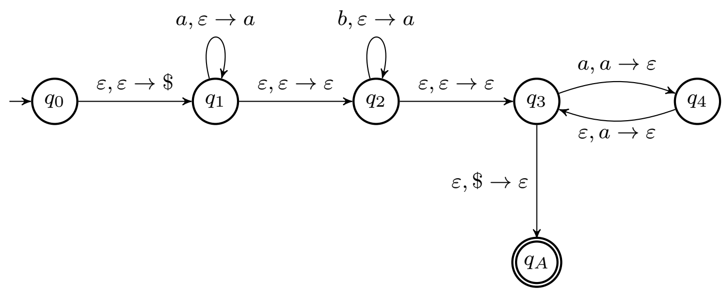
I simboli di lookahead delle regole "P" (seconda, terza e quarta nello stato dell'automa) sono a e b in quanto dalla sequenza ABb che segue P nella prima regola può essere generato come primo simbolo sia a (applicando $A \to aaA$) che b (applicando $A \to \varepsilon$). A causa della quarta regola, lo stato è accettante. D'altra parte, nella seconda regola a segue il punto ed è anche simbolo di lookahead della regola completata (stesso discorso per la terza regola). Pertanto, il DK_1-test è fallito, e quindi la grammatica non è LR(1).

Esercizio 4 [6.5] Si consideri il linguaggio $D = \left\{a^h\, b^k\, a^l \mid k = h \wedge l = 2h\right\}$. D è un linguaggio libero dal contesto (CFL)? Giustificare la risposta con una dimostrazione.

Soluzione: Il linguaggio D non è CFL, come si può dimostrare applicando il "pumping lemma" per CFL. Supponiamo per assurdo che D sia CFL; varrebbe dunque per esso il lemma, e quindi dovrebbe esistere una lunghezza $p > 0$ tale che ogni stringa in D di lunghezza non inferiore a p può essere suddivisa in modo da poter essere "pompata" verso l'alto o verso il basso. Consideriamo dunque la stringa $s = a^p b^p a^{2p} \in D$, ed una sua suddivisione $s = uvxyz$ tale che $uv^ixy^iz \in D$ per ogni $i \geq 0$, $|vy| > 0$ e $|vxy| \leq p$. Affinché la stringa pompata appartenga ancora a D, vxy deve includere almeno una a a sinistra, tutte le b ed almeno una a a destra: infatti se pompando si modifica la lunghezza di una qualunque delle tre sequenze a^p, b^p e a^{2p}, allora anche le altre due sequenze devono essere opportunamente modificate affinché la stringa pompata appartenga a D. Perciò $|vxy| > |ab^pa| = p + 2$, il che è in contraddizione con la condizione $|vxy| \leq p$. La contraddizione è dovuta all'aver supposto che D è CFL. Resta dunque dimostrato che D non è CFL.

Esercizio 5 [6.5] Si consideri il linguaggio $E = \left\{a^h\, b^k\, a^l \mid h + k = 2\,l\right\}$. E è un linguaggio libero dal contesto (CFL)? Giustificare la risposta con una dimostrazione.

Soluzione: Il linguaggio E è CFL. Determinare una grammatica libera dal contesto che genera il linguaggio E non è facile; è molto più semplice invece esibire un PDA che riconosce il linguaggio E:



L'automa utilizza il non determinismo per "indovinare" la posizione delle tre sequenze che costituiscono ciascuna stringa in E. Le prime due sequenze vengono salvate sullo stack (utilizzando solo il simbolo a) negli stati q_1 e q_2. Poi per ciascun simbolo della terza sequenza vengono rimossi due elementi dallo stack (stati q_3 e q_4). Si osservi che l'automa non può accettare se la stringa in ingresso è malformata (ossia non in $a^*b^*a^*$). Se inoltre $h + k < 2l$ allora non si potranno leggere tutti i simboli della terza sequenza nello stato q_3, oppure non si potrà transitare dallo stato q_4 allo stato q_3, quindi l'automa non potrà accettare. Se invece $h + k > 2l$, dopo aver terminato di leggere i simboli della terza sequenza nello stato q_3 non si potrà transitare nello stato di accettazione poiché sulla cima dello stack sarà presente a e non \$. In conclusione, l'automa accetta una stringa in ingresso x se e solo se $x \in E$.

Esercizio 6 [10] Un grafo diretto G è detto *fortemente connesso* (*strongly connected*) se per ogni coppia di nodi $u, v \in V(G)$, esiste un percorso diretto in G da u a v. Dimostrare che il linguaggio STRONGLY CONNECTED contenente le codifiche dei grafi diretti fortemente connessi è NL-completo.

Soluzione: Dimostriamo innanzi tutto che il problema STRONGLY CONNECTED è in NL. Per semplificare questa dimostrazione sfruttiamo il fatto che NL=coNL, ossia $X \in$ NL se e solo se $\overline{X} \in$ NL. Il complemento di STRONGLY CONNECTED è costituito dalle codifiche di grafi diretti in cui esiste almeno una coppia di nodi u, v tale che *non* esiste alcun percorso orientato da u a v. Sfruttiamo inoltre il fatto che il problema PATH è NL-completo, e dunque che il suo complemento $\overline{\text{PATH}}$ è in NL: pertanto, esiste una TM nondeterministica N che su input $\langle G, u, v\rangle$ esegue con spazio di lavoro logaritmico e accetta se e solo se *non* esiste un percorso orientato in G dal nodo $u \in V(G)$ al nodo $v \in V(G)$. Consideriamo dunque la seguente TM non deterministica M:

M= "On input $\langle G\rangle$, where G is a directed graph:
1. Nondeterministically guess a pair of nodes $u, v \in V(G)$
2. Nondeterministically guess a deterministic branch of $N(\langle G, u, v\rangle)$
3. If the deterministic branch of $N(\langle G, u, v\rangle)$ accepts then accept
4. Otherwise, reject"

È evidente che M può implementare il passo 1 utilizzando una quantità di spazio di lavoro logaritmica nella dimensione dell'input; infatti ha necessità di memorizzare due soli vertici del grafo G, e quindi utilizza $O(\log |V(G)|)$ celle di lavoro. Nel passo 2, M sceglie una computazione deterministica della TM non deterministica N su input $\langle G, u, v\rangle$; poiché N opera in spazio logaritmico, la lunghezza massima di tale computazione deterministica è lineare nella dimensione dell'input, e quindi M può tenere traccia dei passi della computazione deterministica utilizzando una quantità logaritmica di celle di lavoro (oltre a quelle utilizzate da N). Infine, i passi 3 e 4 richiedono una quantità costante di celle di lavoro. In conclusione, M è una NTM che accetterà se e solo se esistono $u, v \in V(G)$ tali che esiste un ramo di computazione deterministica accettante di $N(\langle G, u, v\rangle)$, ossia se e solo se esistono $u, v \in V(G)$ tali che $(G, u, v) \in \overline{\text{PATH}}$, ossia se e solo se $\langle G\rangle \in \overline{\text{STRONGLY CONNECTED}}$. Quindi STRONGLY CONNECTED $\in$ coNL = NL.

Dimostriamo ora che STRONGLY CONNECTED è NL-hard esibendo una riduzione in spazio logaritmico da PATH a STRONGLY CONNECTED. In particolare, consideriamo il seguente trasduttore logaritmico T:

T= "On input $\langle G, s, t\rangle$, where G is a directed graph and $s, t \in V(G)$:
1. For each element $x \in V(G)$:
    2. Copy x on the output tape as an element in $V(G')$
    3. Copy (x, s) on the output tape as an element in $E(G')$
    4. Copy (t, x) on the output tape as an element in $E(G')$
5. For each element $(x, y) \in E(G)$:
    6. Copy (x, y) on the output tape as an element in $E(G')$"

Il trasduttore T trasforma l'istanza di PATH $\langle G, s, t\rangle$ in una istanza di STRONGLY CONNECTED $\langle G'\rangle$, ove G' è derivato da G aggiungendo tutti gli archi diretti da t a tutti i nodi di G, e tutti gli archi diretti da ogni nodo di G al nodo s (in sintesi, $V(G') = V(G)$ e $E(G') = E(G) \cup \{(x, s) \mid x \in V(G)\} \cup \{(t, x) \mid x \in V(G)\}$). È facile verificare che T può derivare la codifica di G' utilizzando spazio logaritmico nella dimensione di G: in effetti, deve implementare un ciclo che itera su tutti i nodi del grafo G, ed un altro ciclo che itera su tutti gli archi di G; quindi necessita di un numero di celle di lavoro logaritmico in $|\langle G\rangle|$.

Supponiamo che $\langle G, s, t\rangle \in$ PATH. Pertanto esiste un percorso diretto da s a t nel grafo G. Consideriamo due qualunque nodi $u, v \in V(G') = V(G)$. Per costruzione, esiste in G' un arco da u a s ed un arco da t a v; dunque utilizzando il percorso da s a t che già era in G, esiste un percorso in G' da u a v. Pertanto, $\langle G'\rangle \in$ STRONGLY CONNECTED.

Viceversa, supponiamo che $\langle G'\rangle \in$ STRONGLY CONNECTED. Pertanto, per ogni possibile coppia $u, v \in V(G') = V(G)$, esiste un percorso da u a v. Di conseguenza, esiste in G' un percorso orientato da $u = s$ a $v = t$. Supponiamo che tale percorso utilizzi un arco e in $E(G')$ ma non in $E(G)$: per costruzione tale arco e deve essere entrante in s oppure uscente da t. Tutto il sottopercorso ad anello che porta da s a rientrare in s attraverso l'arco e, ovvero ad uscire da t attraverso l'arco e per poi rientrare in t, può essere rimosso: il risultato è ancora un percorso in $E(G')$ da s a t che non contiene e. Dunque concludiamo che esiste un percorso da s a t che fa uso esclusivamente di archi in $E(G)$, e pertanto $\langle G, s, t \rangle \in$ Path.

In conclusione, Strongly Connected è in NL ed è NL-hard, dunque è NL-completo, come volevasi dimostrare.