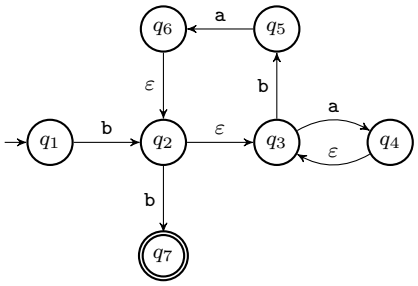
Automi e Linguaggi (M. Cesati)

Facoltà di Ingegneria, Università degli Studi di Roma Tor Vergata

Compito scritto del 8 febbraio 2023

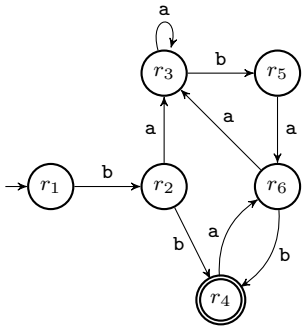
Esercizio 1 [6] Determinare un automa deterministico che riconosca il linguaggio generato dalla espressione regolare `b(a*ba)*b`.

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:

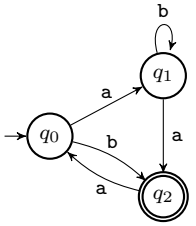


Un automa deterministico equivalente è il seguente:

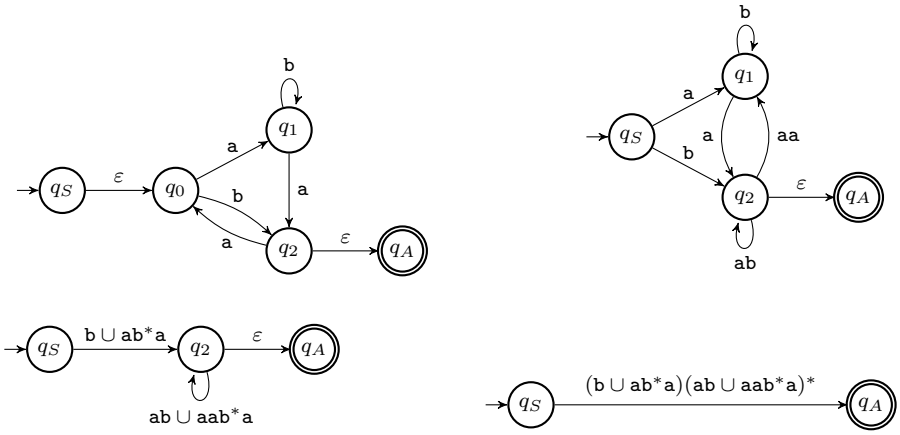
$r_1 = \{q_1\}$
$r_2 = \{q_2, q_3\}$
$r_3 = \{q_3, q_4\}$
$r_4 = \{q_5, q_7\}$
$r_5 = \{q_5\}$
$r_6 = \{q_2, q_3, q_6\}$



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_1 e q_2:



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

$$(\mathtt{b} \cup \mathtt{ab}^*\mathtt{a})(\mathtt{ab} \cup \mathtt{aab}^*\mathtt{a})^*$$

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

$$(\mathtt{ba} \cup \mathtt{ab}^*\mathtt{aa})^*(\mathtt{b} \cup \mathtt{ab}^*\mathtt{a}) \quad \text{(ordine: } q_2,\, q_1,\, q_0\text{)}$$
$$(\mathtt{b} \cup \mathtt{ab}^*\mathtt{a})[\mathtt{a}(\mathtt{b} \cup \mathtt{ab}^*\mathtt{a})]^* \quad \text{(ordine: } q_1,\, q_0,\, q_2\text{)}$$

Esercizio 3 [7] Sia C un linguaggio regolare, e sia M un DFA tale che $L(M) = C$. Dimostrare che C è l'insieme vuoto se e solo se C non include alcuna stringa di lunghezza inferiore al numero di stati interni di M.

Soluzione: Denotiamo con p il numero di stati interni dell'automa M. La dimostrazione della condizione necessaria è banale. Infatti se $C = \emptyset$, allora non esiste alcuna stringa $w \in C$, ed in particolare non esiste alcuna stringa $w \in C$ con $|w| < p$. Per dimostrare la condizione sufficiente, assumiamo che C non includa alcuna stringa di lunghezza inferiore a p, e supponiamo per assurdo che C non sia vuoto. Ciò implica che esiste un elemento $w \in C$ avente lunghezza minima tra quelli in C con necessariamente $|w| = n \geq p$. Poiché $w \in C$, la computazione $M(w)$ è accettante: sia $q_0, \ldots q_n$ la sequenza di stati interni assunti dall'automa man mano che vengono letti i simboli di $w = w_1 w_2 \cdots w_n$. Poiché $n \geq p$, per il principio "pigeonhole" deve esistere almeno uno stato $\overline{q}$ che si ripete nei primi $p + 1$ stati nella sequenza:

$$q_0, q_1, \ldots q_i = \overline{q}, \ldots q_j = \overline{q}, q_{j+1} \ldots q_n$$

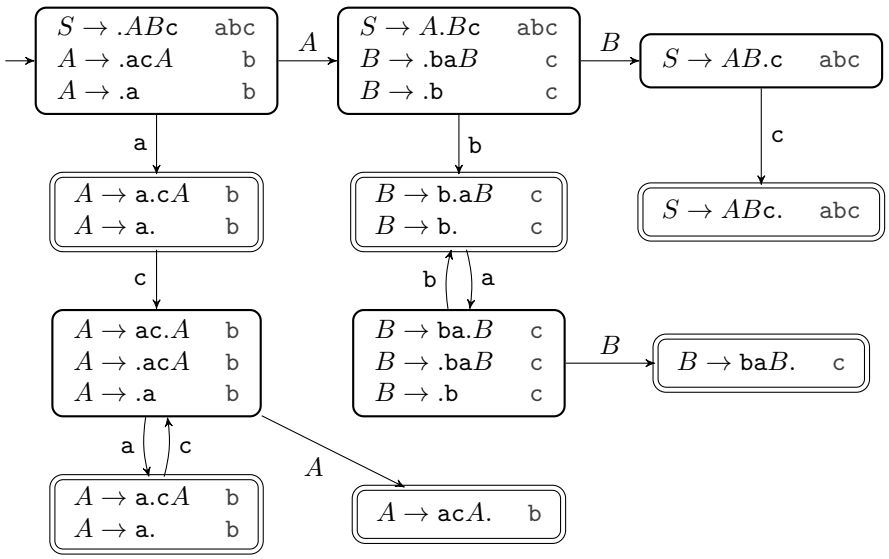
con $0 \leq i < j \leq p$. Consideriamo dunque la stringa $w' = w_1 \cdots w_i w_{j+1} \cdots w_n$: nella computazione $M(w')$ l'automa deterministico entra nello stato q_i dopo aver letto w_i, quindi legge w_{j+1} e transita in q_{j+1}; da questo momento in poi l'automa deterministico segue le stesse transizioni utilizzate per la parte finale della stringa w, quindi termina nello stato di accettazione q_n. Perciò $w' \in C$ e $|w'| < |w|$. Ma questa è una contraddizione, perché abbiamo supposto che w sia un elemento di C di lunghezza minima, e tale contraddizione può derivare unicamente dall'aver supposto che $C \neq \emptyset$. Resta dunque dimostrato che $C = \emptyset$.

Esercizio 4 [6] Si consideri la grammatica libera dal contesto con variabile iniziale S e regole

$$S \to ABc \qquad A \to acA \mid a \qquad B \to baB \mid b$$

La grammatica è deterministica? È LR(1)? Giustificare le risposte.

Soluzione: Per determinare se la grammatica è deterministica o LR(1) eseguiamo il DK_1-test:



Poiché in nessun stato accettante sono incluse regole consistenti tra loro, il DK_1-test indica che la grammatica è LR(1). D'altra parte, se consideriamo l'automa senza i simboli di lookahead, è evidente che sono presenti stati accettanti che includono regole con il dot che precede un simbolo terminale; pertanto la grammatica analizzata non è deterministica.

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

Soluzione: Il linguaggio B non è decidibile, e per dimostrarlo è sufficiente controllare che le ipotesi del Teorema di Rice siano vere. Si consideri come proprietà P della TM il riconoscere un linguaggio regolare. Tale proprietà è non banale: infatti una TM M_1 che accetta tutte le stringhe riconosce Σ^*, che è regolare; d'altra parte, è banale costruire una TM M_0 che riconosce il linguaggio $\{a^n b^n \mid n \geq 0\}$, che sappiamo essere non regolare; quindi M_0 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 regolare 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 B contenente codifiche di TM che soddisfano la proprietà P è indecidibile.

Esercizio 6 [9] Il problema Not-All-Equal SAT (NAESAT) è costituito dalle istanze di SAT in cui esiste una assegnazione di verità alle variabili tale per cui in ogni clausola esiste

almeno un letterale vero ed un letterale falso (ossia, nessuna clausola ha i valori dei letterali tutti veri o tutti falsi). Dimostrare che NAESAT è NP-completo.

Soluzione: Per prima cosa dimostriamo l'appartenza di NAESAT in NP. Il problema è polinomialmente verificabile perché ogni istanza che fa parte del linguaggio ha come certificato l'assegnazione di verità alle variabili che garantisce la presenza in ciascuna clausola di un letterale falso e di un letterale vero. Tale certificato è certamente di dimensione non superiore alla dimensione dell'istanza, e verificare che l'assegnazione soddisfa i requisiti del problema è implementabile in modo immediato, in modo del tutto analogo alla verifica usata per SAT che una assegnazione di verità renda almeno un letterale vero in ogni clausola.

Per dimostrare la NP-hardness di NAESAT costruiamo una riduzione dal problema NP-completo 3SAT. Data una istanza di 3SAT

$$\Phi = \bigwedge_i \bigvee_{j=1}^{3} l_{i,j} \,,$$

la riduzione polinomiale costruisce la formula CNF

$$\Psi = \bigwedge_i \left(s \vee \bigvee_{j=1}^{3} l_{i,j} \right),$$

ove s è una variabile che non appare in alcuno dei letterali $l_{i,j}$.

Supponiamo che Φ sia una "istanza sì" di 3SAT, quindi che esista un assegnazione di valori di verità alle variabili di Φ che rende almeno un letterale entro ogni clausola vero. La stessa assegnazione di valori, estesa con l'assegnazione del valore "falso" alla variable s, assicura che nella formula Ψ esista almeno un letterale vero ed un letterale falso in ogni clausola. Pertanto, Ψ derivata da Φ è una "istanza sì" di NAESAT.

Viceversa, supponiamo che una formula Ψ costruita con lo stesso procedimento sia una "istanza sì" di NAESAT, pertanto che esista una assegnazione di verità alle variabili di Ψ tale che ogni clausola di Ψ include almeno un letterale vero ed un letterale falso. Distinguiamo due casi: se l'assegnazione di verità ha posto la variabile s a "falso", allora la stessa assegnazione di verità (senza s) soddisfa la formula Φ da cui Ψ è stata derivata, poiché le rimanenti variabili garantiscono la presenza di almeno un letterale vero in ogni clausola di Φ. Se invece l'assegnazione di verità ha posto la variabile s a "vero", allora si consideri l'assegnazione di verità complementare, in cui tutte le variabili ricevono la negazione del valore precedentemente assegnato. È immediato verificare che la nuova assegnazione di verità continua a garantire la presenza in ogni clausola di Ψ di un letterale vero e di un letterale falso, perché il valore di ciascun letterale viene negato rispetto alla precedente assegnazione. Quindi anche la assegnazione complementare certifica che Ψ appartiene a NAESAT. Nella assegnazione complementare la variabile s ha il valore "falso", dunque si ricade nel caso precedente: la formula Φ è una "istanza sì" di 3SAT.

In conclusione, ogni "istanza sì" di 3SAT corrisponde ad una "istanza sì" di NAESAT, ed ogni "istanza no" di 3SAT corrisponde ad una "istanza no" di NAESAT. La trasformazione dalle istanze di 3SAT a quelle di NAESAT è evidentemente costruibile in tempo polinomiale e costituisce una riduzione tra problemi. Quindi NAESAT è NP-hard, e pertanto è NP-completo, come volevasi dimostrare.