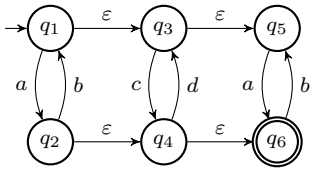
Automi e Linguaggi (M. Cesati)

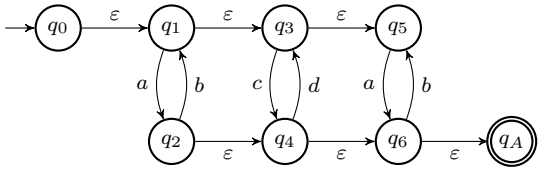
Facoltà di Ingegneria, Università degli Studi di Roma Tor Vergata

Compito scritto del 4 settembre 2019

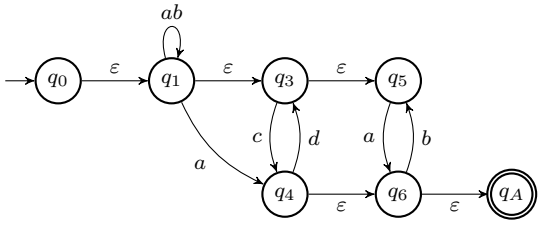
Esercizio 1 [8] Derivare l'espressione regolare che definisce il linguaggio riconosciuto dall'automa:



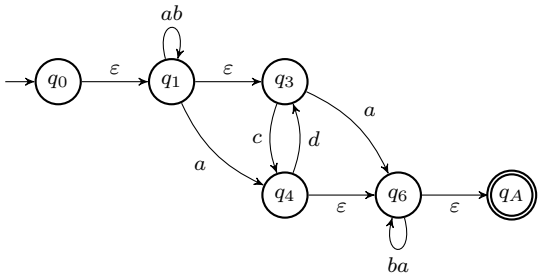
Soluzione: La derivazione di una espressione regolare comporta la trasformazione in un GNFA e la successiva eliminazione di tutti gli stati non iniziali e finali. Come primo passo consideriamo dunque un GNFA derivato dall'automa iniziale (tutti gli archi con etichetta $\emptyset$ sono omessi dal grafico):



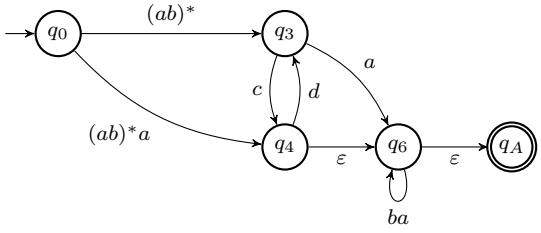
La scelta dell'ordine degli stati da eliminare è arbitraria, ma generalmente conviene iniziare dagli stati aventi un numero di archi entranti e/o uscenti piccolo. Cominciamo dallo stato q_2: esso ha archi entranti da q_1 e archi uscenti verso q_1 e q_4; pertanto il nuovo automa ha etichette aggiornate per gli archi da q_1 e verso q_1 e q_4:



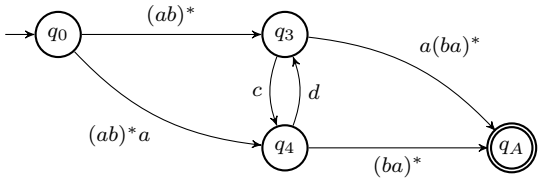
Eliminiamo ora q_5, che ha archi entranti da q_3 e q_6 e uscenti verso q_6:



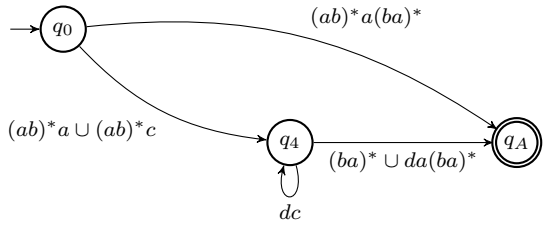
Lo stato q_1 ha un arco entrante da q_0 ed archi uscenti verso q_3 e q_4:



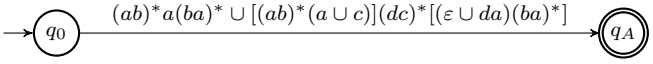
Lo stato q_6 ha archi entranti da q_3 e q_4, ed un arco uscente verso q_A:



Lo stato q_3 ha archi entranti da q_0 e q_4, ed archi uscenti verso q_4 e q_A:



Semplifichiamo l'etichetta $(ab)^*a \cup (ab)^*c$ in $(ab)^*(a \cup c)$, e l'etichetta $(ba)^* \cup da(ba)^*$ in $(\varepsilon \cup da)(ba)^*$. Rimuovendo l'ultimo stato q_4 si ottiene:



Semplificando ancora l'etichetta $(ab)^*a(ba)^* \cup [(ab)^*(a \cup c)](dc)^*[(\varepsilon \cup da)(ba)^*]$ si ottiene l'espressione regolare

$$(ab)^* \left[a \cup (a \cup c)(dc)^*(\varepsilon \cup da)\right] (ba)^*.$$

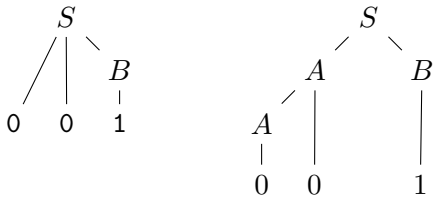
Esercizio 2 [6] Dimostrare che la grammatica context-free seguente è ambigua:

$$S \to AB \mid \texttt{00}B \qquad A \to \texttt{0} \mid A\texttt{0} \qquad B \to \texttt{1}$$

Soluzione: Una grammatica si definisce ambigua se esistono due derivazioni "leftmost" (o equivalentemente "rightmost") della stessa stringa terminale. Consideriamo dunque la stringa terminale `001`: essa può essere prodotta con le seguenti due diverse derivazioni "leftmost":

$$S \to \texttt{00}B \to \texttt{001}$$
$$S \to AB \to A\texttt{0}B \to \texttt{00}B \to \texttt{001}$$

Equivalentemente, la grammatica è certamente ambigua perché la stringa terminale `001` può essere prodotta con due differenti alberi sintattici ("parse tree"):



Esercizio 3 [8] Dimostrare che il seguente linguaggio non è context-free:

$$A = \{w \in \{a, b, c\}^* \mid w \text{ ha un ugual numero di } a, b \text{ e } c\}$$

Soluzione: Una facile dimostrazione indiretta può essere basata su due risultati già stabiliti in precedenza. Il primo risultato è che l'insieme $\{0^n 1^n 2^n \mid n \geq 0\}$ non è un CFL (dimostrato durante le lezioni del corso), quindi l'insieme ad esso isomorfo $B = \{a^n b^n c^n \mid n \geq 0\}$ non è un CFL. Il secondo risultato è l'asserto di un esercizio di una sessione d'esami precedente, ossia che l'intersezione di un CFL ed un linguaggio regolare è un CFL. È facile verificare che $B = A \cap R$, ove R è il linguaggio generato dall'espressione regolare $a^*b^*c^*$. Pertanto, se per assurdo A fosse CFL, allora anche B dovrebbe esserlo, una contraddizione.

Per una dimostrazione diretta utilizziamo il pumping lemma. Supponiamo per assurdo che A sia un CFL, e sia p la corrispondente lunghezza garantita dal lemma. Scegliamo come stringa $s = a^p b^p c^p$; poiché vale il pumping lemma, deve esistere una suddivisione $s = uvxyz$ ove $|vxy| \leq p$, $|vy| > 0$, e $uv^i xy^i z \in A$ per ogni $i \geq 0$. Qualunque suddivisione per la quale vy non contenga un ugual numero di a, b e c non può preservare l'appartenza a A. D'altra parte, qualunque suddivisione tale che vy contenga sia a che b che c deve essere lunga almeno $p + 2$ simboli (un a, tutti i b ed un c). Pertanto risulterebbe $|vxy| > p$, mentre il pumping lemma garantisce l'esistenza di una suddivisione con $|vxy| \leq p$. La contraddizione deriva dall'aver supposto che A sia un CFL.

Esercizio 4 [9] Si consideri il linguaggio $U = \{\langle M \rangle \mid M \text{ è una TM e } n \in L(M)\}$ ove n è la codifica in binario del tuo numero di matricola universitaria. Il linguaggio U è decidibile? La risposta deve includere una dimostrazione.

Soluzione: Poiché il linguaggio è costituito da codifiche di macchine di Turing, è plausibile che ad esso possa applicarsi il teorema di Rice, che afferma che qualunque proprietà non banale relativa ai linguaggi riconosciuti da TM è indecidibile.

Per verificare se le ipotesi del teorema di Rice sono soddisfatte, consideriamo se la proprietà caratterizzante l'insieme U è banale o meno: nella definizione dell'insieme il numero di matricola corrisponde semplicemente ad una stringa di bit ben definita e costante. Quindi la proprietà non è banale, in quanto almeno un insieme (l'insieme vuoto) non include questa stringa binaria, mentre almeno un altro insieme ($\{0,1\}^*$) la include. La seconda ipotesi del teorema di Rice è che la proprietà caratterizzante U deve riferirsi al linguaggio riconosciuto dalle macchine di Turing, e non alle TM stesse. Ciò è evidente dalla definizione stessa di U; in altri termini, è evidente che se $M \in U$ allora per ogni altra TM M' tale che $L(M) = L(M')$, $n \in L(M')$, e dunque $M' \in U$.

Avendo quindi verificato le ipotesi del teorema di Rice, possiamo concludere direttamente con il suo asserto, ossia che il linguaggio U non è decidibile.

Una dimostrazione alternativa dell'indecidibilità di U può essere basata su di una riduzione da un problema indecidibile come A_{TM}. Supponiamo per assurdo che U sia decidibile, quindi esista una TM M_U che decida U. Sia inoltre la TM N un elemento di U, ovvero $n \in L(N)$. Data una istanza di A_{TM} (M, w), possiamo costruire la seguente TM P:

P="On input $\langle M, w \rangle$, where M is a TM and w is its input:

1. Construct a TM M_w that accepts on input x if and only if M accepts w and N accepts x.

2. Run M_U on M_w. If it accepts, accept; otherwise, reject."

Supponiamo che $w \in L(M)$: in questo caso $L(M_w) = L(N)$, perché M accetta w, quindi $n \in L(M_w)$ e pertanto $M_w \in U$; la TM P accetta (M, w) in quanto M_U accetta M_w. Nel caso invece che $w \notin L(M)$, $L(M_w) = \emptyset$, quindi $n \notin L(M_w)$; la TM P rifiuta (M, w) poiché M_U rifiuta M_w. L'esistenza di una TM M_U che decide U implica dunque l'esistenza di una TM P che decide A_{TM}; poiché A_{TM} è indecidibile, possiamo concludere che anche U deve essere indecidibile.

Esercizio 5 [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.