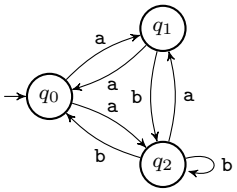
Automi e Linguaggi (M. Cesati)

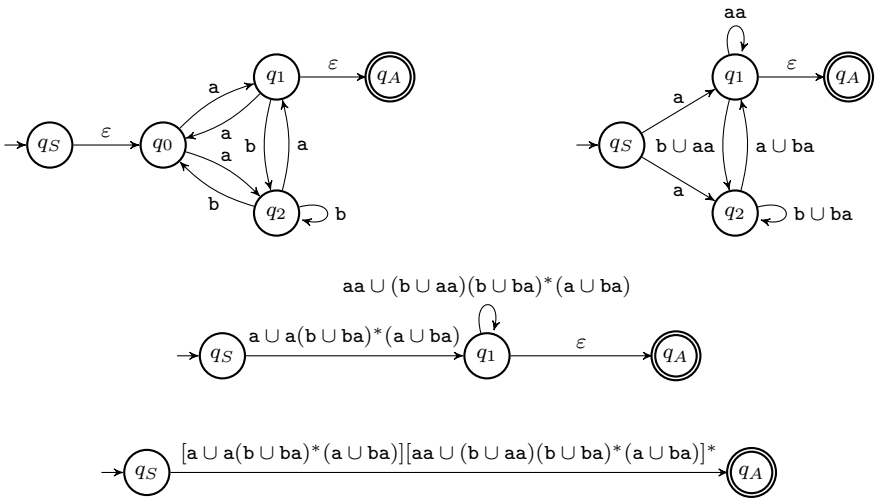
Facoltà di Ingegneria, Università degli Studi di Roma Tor Vergata

Compito scritto del 13 febbraio 2024

Esercizio 1 [6.5] Determinare una espressione regolare (REX) per il linguaggio riconosciuto dal seguente NFA:



Soluzione: Determiniamo un GNFA equivalente al NFA dato e rimuoviamo, nell'ordine, i nodi q_0, q_2 e q_1:



Una REX che genera il linguaggio riconosciuto dal NFA è dunque

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

Esercizio 2 [6] Sia A il linguaggio riconosciuto dalla REX $\mathtt{1}(\mathtt{0} \cup \mathtt{1})^*\mathtt{1}$. Sia

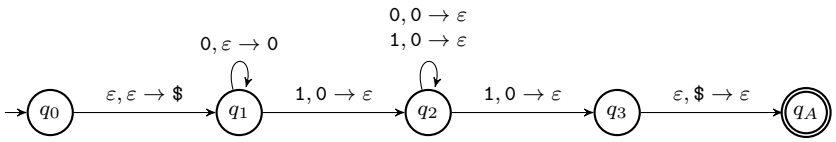
$$B = \{w \in \{\mathtt{0},\mathtt{1}\}^* \mid w = \mathtt{0}^n v,\ v \in A,\ |v| = n,\ n > 0\}.$$

Dimostrare che il linguaggio B non è regolare.

Soluzione: Dimostriamo che B non è regolare. Supponiamo per assurdo che lo sia, e dunque che si possa applicare il Pumping Lemma. Sia $p > 0$ la "pumping length" per B, e consideriamo la stringa $s = 0^p1^p$. È immediato verificare che $s \in B$ e che $|s| \geq p$. Deve quindi esistere una suddivisione $s = xyz$ con $|y| > 0$, $|xy| \leq p$ e $xy^iz \in B$ per ogni $i \geq 0$. Qualunque sia tale suddivisione, y contiene solo zeri (poiché $|xy| \leq p$) ed almeno uno zero (poiché $|y| > 0$). Consideriamo quindi il caso $i = 0$: la stringa $xy^0z = xz$ contiene meno zeri di s mentre il numero di '1' è invariato, quindi ha la forma 0^q1^p con $q < p$: pertanto $xz \notin B$. Perciò assumere che B sia regolare porta ad una contraddizione.

Esercizio 3 [6.5] Si consideri il linguaggio B come nell'esercizio precedente. Si determini un automa a pila (PDA) P tale che $L(P) = B$, oppure dimostrare che tale PDA non esiste.

Soluzione: La struttura degli elementi del linguaggio B è molto semplice: le stringhe sono costituite da un numero $n > 0$ di '0', seguiti da una stringa che inizia e termina con '1' e lunga esattamente n simboli. Il progetto del PDA è quindi consequenziale: si comincia scrivendo il simbolo di "fine stack", poi si accumulano sullo stack gli '0' iniziali. Appena si legge '1' in input si comincia a leggere la seconda parte della stringa rimuovendo per ogni carattere letto un simbolo dallo stack. Per terminare, si indovina non deterministicamente la posizione dell'ultimo '1' della stringa, e si accetta se lo stack non contiene più alcun '0'.



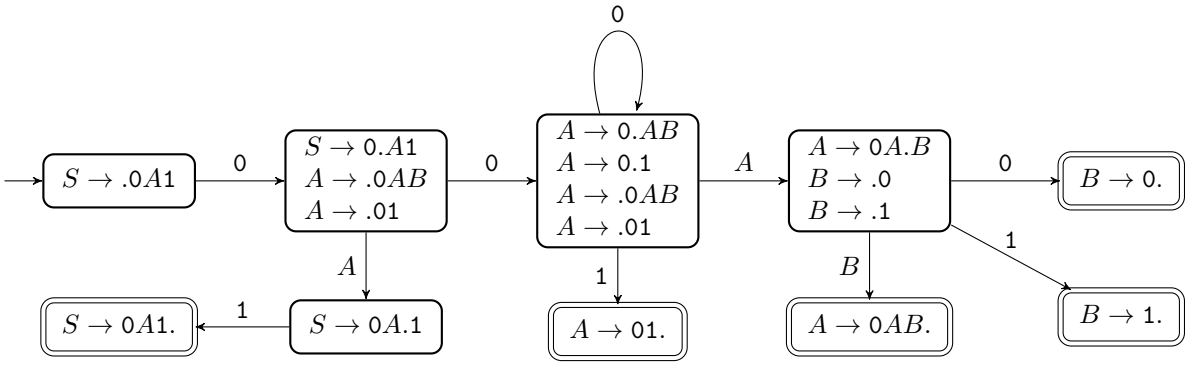
Se la scelta non deterministica effettuata nello stato q_2 risulta sbagliata, il ramo di computazione deterministica non può accettare in quanto non può essere letto alcun altro simbolo della stringa di ingresso. Per lo stesso motivo, se la stringa non termina con '1' l'automa non può accettare. Se gli zeri iniziali sono in numero maggiore della restante parte della stringa, non è possibile rimuovere tutti gli '0' accumulati sullo stack e quindi non si può arrivare allo stato di accettazione. Se infine il numero di zeri iniziali è inferiore alla restante parte della stringa, non è possibile consumare tutti i simboli in ingresso perché non esiste transizione da q_2 che legge un simbolo in ingresso senza contemporaneamente leggere uno '0' dallo stack. Pertanto questo PDA riconosce esattamente il linguaggio B.

Esercizio 4 [6] Si consideri la grammatica G con variabile iniziale S avente le regole:

$$S \to 0A1 \quad A \to 0AB \,|\, 01 \quad B \to 0 \,|\, 1$$

Determinare se G è una grammatica deterministica.

Soluzione: Per verificare se la grammatica G è deterministica utilizziamo il DK-test.



Poiché nessuno stato di accettazione include due regole tra loro consistenti (due regole terminate oppure una regola terminata ed un'altra regola non terminata in cui il dot precede un simbolo terminale), il DK-test ha successo e quindi la grammatica G è deterministica.

Esercizio 5 [7] Sia A un linguaggio Turing-riconoscibile (ossia ricorsivamente enumerabile) e B un linguaggio decidibile. Sia $X \setminus Y = \{x \in X \mid x \notin Y\}$. Quali tra i quattro linguaggi $A \setminus B$, $A \setminus B^c$, $A^c \setminus B$, $A^c \setminus B^c$ è Turing riconoscibile? Giustificare le risposte con dimostrazioni o controesempi.

Soluzione: Supponiamo, senza perdita di generalità, che A e B siano linguaggi definiti su di un comune insieme alfabeto Σ. Osserviamo subito che, poiché B è per ipotesi decidibile, allora anche il suo complemento B^c deve essere decidibile. Pertanto, i quattro casi del testo dell'esercizio si riducono ai seguenti due, in quanto la risposta per $A \setminus B$ vale anche a $A \setminus B^c$, e la risposta per $A^c \setminus B$ vale anche per $A^c \setminus B^c$.

(a) $A \setminus B$: questo linguaggio è Turing-riconoscibile. Infatti, sia M_A la DTM che riconosce il linguaggio A, e sia M_B la DTM che decide il linguaggio B. Da esse si può derivare la seguente DTM:

$M =$"On input x, where x is a string in Σ^*:
1. run M_A on input x
2. if $M_A(x)$ rejected, then reject
3. run M_B on input x
4. if $M_B(x)$ rejected, then accept
5. reject"

Supponiamo che $x \in A \setminus B$; poiché M_A riconosce A, $M_A(x)$ termina in un numero finito di passi accettando. Pertanto M arriva ad eseguire il passo 3. Poiché M_B decide B, $M_B(x)$ termina in un numero finito di passi e rifiuta, in quanto $x \notin B$. Perciò $M(x)$ accetta al passo 4. D'altra parte, se $x \notin A \setminus B$, allora $M(x)$ potrebbe non terminare, in quanto non si

ha alcuna garanzia che il passo 1 termini in un numero finito di passi. Quindi M riconosce (ma non decide) il linguaggio $A \setminus B$.

(b) $A^c \setminus B$: questo linguaggio non è necessariamente Turing-riconoscibile. Per dimostrarlo è sufficiente considerare $A = \mathcal{A}_{\text{TM}}$ (il linguaggio Turing-riconoscibile ma non decidibile di accettazione delle TM) e $B = \emptyset$. Il linguaggio $A^c \setminus B = \mathcal{A}_{\text{TM}}^c \setminus \emptyset = \mathcal{A}_{\text{TM}}^c$ non è Turing-riconoscibile. Infatti, se un linguaggio ed il suo complemento sono Turing-riconoscibili, allora il linguaggio stesso è decidibile; però sappiamo che $\mathcal{A}_{\text{TM}}$ non è decidibile.

Esercizio 6 [8] Il problema PCP (Post Correspondence Problem) non è decidibile. Si consideri la restrizione F-PCP di PCP: le sue istanze-sì sono le istanze-sì di PCP che hanno un match con una sequenza di tessere tutte differenti tra loro. F-PCP è decidibile? Giustificare la risposta con una dimostrazione.

Soluzione: Una generica istanza del problema PCP è una collezione di tessere

$$P = \left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \ldots, \left[\frac{t_n}{b_n}\right] \right\}$$

con $t_i, b_i \in \Sigma^*$ per ogni i. Una istanza-sì è una collezione P tale che esiste un "match", ossia una sequenza finita di tessere di P (eventualmente ripetute) per cui la concatenazione delle stringhe superiori delle tessere produce la stessa stringa della concatenazione delle stringhe inferiori. Sappiamo che il problema PCP è indecidibile, sostanzialmente poiché è possibile costruire una istanza di PCP che simula esattamente la computazione di una macchina di Turing.

Nel problema PCP è cruciale che per la costruzione del match si possano utilizzare tessere identiche (ossia ripetute). La riduzione tra $\mathcal{A}_{\text{TM}}$ e PCP usa in effetti in modo essenziale tessere identiche nella costruzione del match. Se consideriamo la restrizione F-PCP di PCP in cui le istanze-sì sono caratterizzate da match in cui non appaiono tessere ripetute, la riduzione canonica da $\mathcal{A}_{\text{TM}}$ non funziona più; in effetti, è facile mostrare che F-PCP è un problema decidibile.

Consideriamo la seguente DTM T:

$T =$"On input P, where P is a collection of PCP's dominos:
1. for $k = 1$ to n:
    2. for each disposition π of class k of the elements in P:
        3. if π is a PCP match, then accept
4. reject"

La decidibilità di F-PCP risiede nel fatto che il numero di possibili match costituiti da tessere tutte differenti è finito e dipende ovviamente dalla cardinalità n dell'insieme di tessere P nella

istanza del problema. Più precisamente, poiché esistono $D_{n,k} = \frac{n!}{(n-k)!}$ modi di permutare k elementi tratti da un insieme di n elementi (senza ripetizioni), il numero di esecuzioni del passo 3 è $\sum_{k=1}^{n} D_{n,k}$, che ovviamente è un numero finito. A propria volta, il passo 3 consiste nel verificare se una data permutazione π è un match valido per PCP, ossia se la concatenazione delle k stringhe superiori dei domino in π coincide con la concatenazione delle k stringhe inferiori.

Pertanto T decide ogni istanza di F-PCP, e quindi questo problema è decidibile.