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

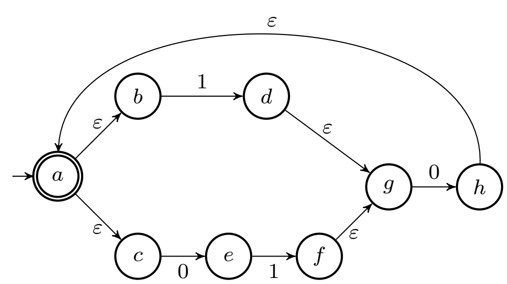
Facoltà di Ingegneria, Università degli Studi di Roma Tor Vergata

Compito scritto del 5 luglio 2019

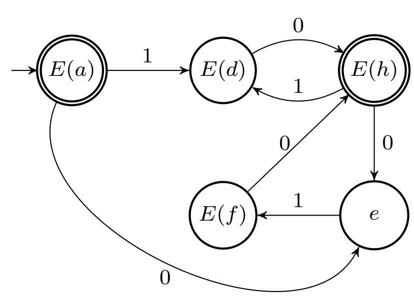
Esercizio 1 [5] Si consideri l'espressione regolare $R = ((1 \cup (01))0)^*$. Costruire un DFA che riconosce il linguaggio generato da R.

Soluzione: Il linguaggio è molto semplice e quindi l'automa deterministico può essere costruito direttamente senza problemi. Mostriamo però il procedimento meccanico che dapprima deriva da R un NFA N che riconosce il linguaggio, poi trasforma N in un equivalente DFA D.

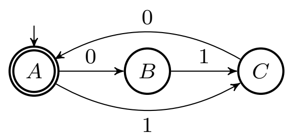
Adottando le regole di composizione degli NFA per le operazioni di concatenazione, unione, e star, ed eliminando qualche transizione ε ridondante, si costruisce lo NFA N seguente:



Calcoliamo gli insiemi chiusura di ciascuno stato rispetto alle transizioni ε: $E(a) = \{a, b, c\}$, $E(b) = \{b\}$, $E(c) = \{c\}$, $E(d) = \{d, g\}$, $E(e) = \{e\}$, $E(f) = \{f, g\}$, $E(g) = \{g\}$, $E(h) = \{h, a, b, c\}$. La procedura di conversione dall'NFA N al DFA D produce il seguente automa:



È immediato osservare che le transizioni uscenti dagli stati marcati con $E(a)$ e con $E(h)$ sono identiche, così come quelle uscenti dagli stati marcati con $E(d)$ e $E(f)$; dunque possiamo semplificare l'automa ed ottenere il DFA seguente:



Esercizio 2 [7] Sia C un linguaggio context-free e sia R un linguaggio regolare. Dimostrare che $C \cap R$ è un linguaggio context-free.

Soluzione: Cominciamo con l'osservare che considerazioni insiemistiche sulla natura di $C \cap R$ non portano da nessuna parte. Ad esempio, a nulla serve enunciare che $C = (C \cap R) \cup (C \cap \overline{R})$ e che l'unione di linguaggi CFL è CFL. Infatti, ciò non significa che un linguggio CFL esprimibile come $C = A \cup B$ implichi necessariamente che A o B siano CFL. Od ancora, il fatto che il linguaggio regolare R sia anche CFL non implica che il suo sottoinsieme $C \cap R$ sia necessariamente CFL o regolare; questa linea di ragionamento, se applicata a due linguaggi in CFL, porterebbe a concludere che la loro intersezione è in CFL, cosa manifestamente non vera.

Poiché C è CFL, esiste un PDA $P = (Q_P, \Sigma_P, \Gamma_P, \delta_P, q_0^P, F_P)$ che accetta tutti e solo gli elementi di C. Analogamente, poiché R è regolare, esiste un DFA $D = (Q_D, \Sigma_D, \delta_D, q_0^D, F_D)$ che accetta tutti e solo gli elementi di R. L'idea della dimostrazione è di esibire un altro PDA che si comporta esattamente come P ma contemporaneamente simula l'esecuzione di D, ed accetta se e solo se sia P che D terminano in stato di accettazione.

Consideriamo dunque un altro PDA $P' = (Q, \Sigma, \Gamma, \delta, q_0, F)$ ove $Q = Q_P \times Q_D$, $\Sigma = \Sigma_P \cup \Sigma_D$, $\Gamma = \Gamma_P$, $q_0 = (q_0^P, q_0^D)$ e $F = F_P \times F_D$.

Per ogni stato $q = (q^P, q^D) \in Q = Q_P \times Q_D$ e simboli $x \in \Sigma_\varepsilon$, $y \in \Gamma_\varepsilon$, sia

$$\delta(q,x,y) = \begin{cases} \left\{ ((q'^P, q^D), y') \mid (q'^P, y') \in \delta_P(q^P, \varepsilon, y) \right\} & \text{se } x = \varepsilon \\ \left\{ ((q'^P, q'^D), y') \mid (q'^P, y') \in \delta_P(q^P, x, y) \text{ e } \delta_D(q^D, x) = q'^D \right\} & \text{se } x \in \Sigma_P \cap \Sigma_D \\ \emptyset & \text{altrimenti.} \end{cases}$$

Si osservi che se P' legge un simbolo di input che non appartiene all'alfabeto di R allora non può accettare l'intera stringa perché non può applicare alcuna transizione. P' può anche effettuare delle transizioni senza consumare simboli di input, ed in questo caso lo stato di D registrato negli stati di P' non cambia. Infine, se il simbolo di input x appartiene sia a Σ_P che Σ_D, allora può essere applicata una qualunque delle transizioni definite da P e contemporaneamente la transizione definita da D.

Supponiamo che $w \in C \cap R$; allora esiste una sequenza di transizioni di P che porta $P(w)$ in uno stato in F_P, ed esiste una sequenza di transizioni di D che porta $D(w)$ in uno stato di F_D. Pertanto per costruzione esiste una sequenza di transizioni di P' che porta $P'(w)$ in uno stato di $F = F_P \times F_D$. Viceversa, se P' accetta una stringa w, allora per costruzione di P' sia $P(w)$ che $D(w)$ accettano, e dunque $w \in C \cap R$.

Poiché esiste un PDA che riconosce $C \cap R$, concludiamo che esso è CFL.

Esercizio 3 [6] Dimostrare che il linguaggio

$$C = \left\{ w \in \{\mathtt{1}, \mathtt{+}, \mathtt{=}\}^* \mid w = \mathtt{1}^i\mathtt{+}\mathtt{1}^j\mathtt{=}\mathtt{1}^{i+j}, i, j > 0 \right\}$$

è DCFL esibendo un opportuno PDA deterministico.

Soluzione: L'automa a pila deterministico può svolgere efficacemente il riconoscimento delle stringhe di C utilizzando lo stack per contare il numero di `1` degli addendi che precedono il simbolo `=` e confrontandolo con il numero di `1` che seguono il simbolo `=`.

Formalmente, la descrizione ad alto livello del DPDA N che decide C è la seguente:

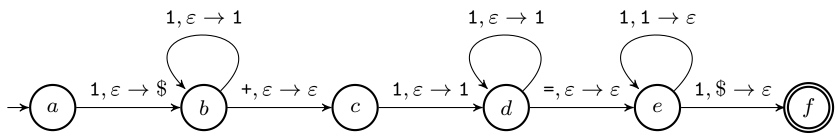
N= "On input w, where $w = w_1 w_2 \cdots w_m \in \{\mathtt{1}, \mathtt{+}, \mathtt{=}\}^*$:

1. if $w_1 \neq \mathtt{1}$ then reject
2. while iterating over the symbols $w_1, w_2, \ldots, w_m$ of the input:
   3. if $w_i \neq \mathtt{1}$ then break the loop and go to step 5
   4. push `1` onto the stack

5 if $w_i \neq \mathtt{+}$ or $w_{i+1} \neq \mathtt{1}$ then reject

6. while iterating over the remaining symbols $w_{i+1}, \ldots, w_m$ of the input:
   7. if $w_j \neq \mathtt{1}$ then break the loop and go to step 9
   8. push `1` onto the stack
9. if $w_j \neq \mathtt{=}$ then reject
10. while iterating over the remaining symbols $w_{j+1}, \ldots, w_m$ of the input:
    11. if $w_k \neq \mathtt{1}$ then reject
    12. if the stack is empty then reject
    13. pop `1` from the stack
14. if the stack is empty then accept, otherwise reject."

In alternativa, e più semplicemente, possiamo descrivere il PDA tramite il grafo dei suoi stati e delle sue transizioni. Ad esempio, il seguente automa N a 6 stati riconosce C:



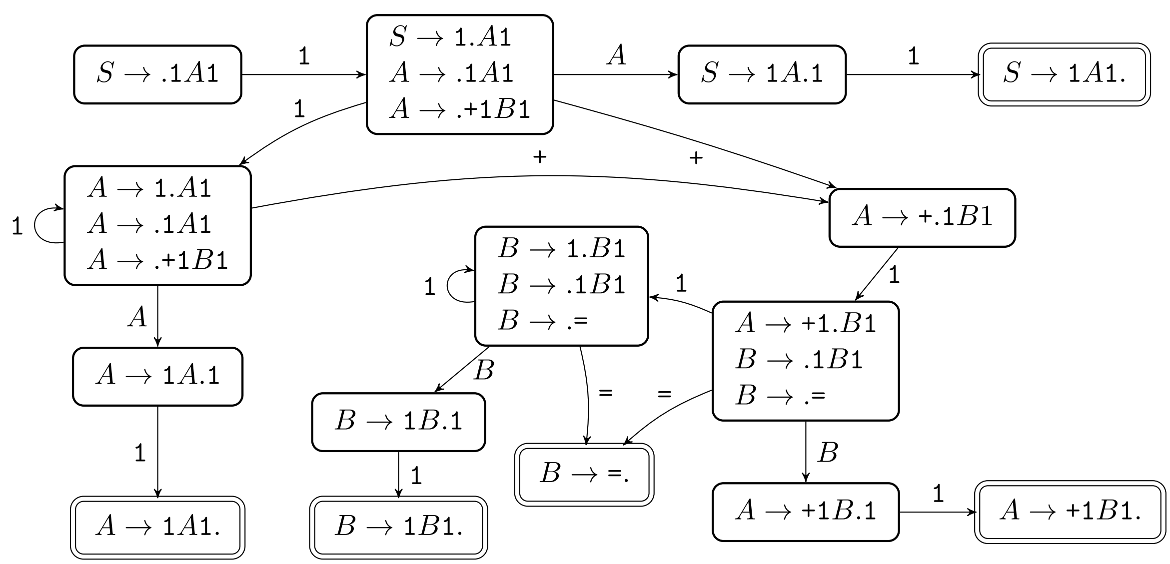
In entrambe le descrizioni di N è immediato verificare che N è un automa a pila deterministico che accetta la stringa in input se e solo se essa è della forma $\texttt{1}^+\texttt{+1}^+\texttt{=1}^+$ ed inoltre il numero di `1` che precedono il simbolo `=` è uguale al numero di `1` che lo seguono. Pertanto, C è DCFL.

Esercizio 4 [11] Determinare una grammatica context-free deterministica per il linguaggio C dell'esercizio precedente, eseguendo su di essa il DK-test.

Soluzione: Una grammatica context-free che genera il linguaggio C è la seguente:

$$
\begin{aligned}
S &\to \texttt{1}A\texttt{1} \\
A &\to \texttt{1}A\texttt{1} \;\mid\; \texttt{+1}B\texttt{1} \\
B &\to \texttt{1}B\texttt{1} \;\mid\; \texttt{=}
\end{aligned}
$$

Dimostriamo per prima cosa che questa grammatica è deterministica applicando il DK-test. Si ottiene il seguente diagramma:



Poiché tutti gli stati terminali contengono una singola regola completata, il test ha successo e quindi la grammatica è deterministica.

Dimostriamo ora che la grammatica genera tutti e soli gli elementi di C. Infatti, sia $w \in C$; allora esistono $i, j > 0$ tali che $\texttt{1}^i\texttt{+}\texttt{1}^j\texttt{=}\texttt{1}^{i+j}$. Possiamo ridurre la stringa w in questo modo:

$$\texttt{1}^i\texttt{+}\texttt{1}^j\texttt{=}\texttt{1}^j\texttt{1}^i = \texttt{1}^i\texttt{+}\texttt{1}\,\overbrace{\texttt{1}^{j-1}\texttt{=}\texttt{1}^{j-1}}^{B\cdots}\,\texttt{1}\texttt{1}^i \overset{*}{\mapsto} \texttt{1}^i\,\overbrace{\texttt{+}\texttt{1}B\texttt{1}}^{A}\,\texttt{1}^i \mapsto \texttt{1}\,\overbrace{\texttt{1}^{i-1}A\texttt{1}^{i-1}}^{A\cdots}\,\texttt{1} \overset{*}{\mapsto} \overbrace{\texttt{1}A\texttt{1}}^{S} \mapsto S.$$

Viceversa, sia w una stringa terminale generata dalla grammatica deterministica ($S \Rightarrow w$), e supponiamo per assurdo che $w \notin C$. Allora deve verificarsi uno dei seguenti casi:

1. w non inizia con $\texttt{1}$: si ha una contraddizione perché la variabile iniziale S può espandere solo in $\texttt{1}A\texttt{1}$.

2. w non contiene $\texttt{=}$: si ha una contraddizione perché l'unico modo per ottenere una stringa terminale dalla grammatica è eliminare la variabile B con la regola $B \to \texttt{=}$. Ogni altra regola introduce un'altra variabile non terminale.

3. w non contiene $\texttt{+}$: si ha una contraddizione perché l'unico modo per ottenere una stringa terminale dalla grammatica è eliminare la variabile B con la regola $B \to \texttt{=}$, e l'unico modo per ottenere la variabile B è tramite la regola $A \to \texttt{+1B1}$.

4. in w c'è più di una occorrenza di $\texttt{+}$: si ha una contraddizione perché l'unica regola che genera $\texttt{+}$ può essere applicata una sola volta: S genera una sola A nella stringa, ogni A genera una sola A oppure una sola B (ed in questo caso genera anche $\texttt{+}$), e le regole di B non possono generare $\texttt{+}$.

5. in w c'è più di una occorrenza di $\texttt{=}$: si ha una contraddizione perché l'unica regola che genera $\texttt{=}$ può essere applicata una sola volta: S genera una sola A nella stringa, ogni A genera una sola A oppure una sola B, e la regola di B che genera $\texttt{=}$ produce necessariamente la stringa terminale.

6. in w, $\texttt{=}$ precede $\texttt{+}$: si ha una contraddizione perché l'unico modo per ottenere una stringa terminale è tramite l'applicazione di $B \to \texttt{=}$ preceduta dall'applicazione di $A \to \texttt{+1}B\texttt{1}$. Quindi $\texttt{+}$ deve precedere $\texttt{=}$.

7. in w, $\texttt{=}$ segue immediatamente $\texttt{+}$: si ha una contraddizione perché la regola $A \to \texttt{+1}B\texttt{1}$ che genera $\texttt{+}$ pone immediatamente dopo il simbolo $\texttt{1}$.

8. w termina con $\texttt{=}$: si ha una contraddizione perché la regola che genera $\texttt{=}$ è $B \to \texttt{=}$, e B può essere generata solo dalle regole $A \to \texttt{+1}B\texttt{1}$ oppure $B \to \texttt{1}B\texttt{1}$: in entrambi i casi la stringa termina con $\texttt{1}$, non $\texttt{=}$.

9. $w = \texttt{1}^i\texttt{+}\texttt{1}^j\texttt{=}\texttt{1}^h$ con $h > i + j$: si ha una contraddizione, perché riducendo la stringa si otterrebbe:

$$\texttt{1}^i\texttt{+}\texttt{1}^j\texttt{=}\texttt{1}^h \mapsto \texttt{1}^i\texttt{+}\texttt{1}^jB\texttt{1}^h \overset{*}{\mapsto} \texttt{1}^i\texttt{+1}B\texttt{1}\texttt{1}^{h-j} \mapsto \texttt{1}^iA\texttt{1}^{h-j} \overset{*}{\mapsto} \texttt{1}A\texttt{1}\texttt{1}^{h-(i+j)} \mapsto S\texttt{1}^{h-(i+j)}$$

ed ovviamente non esisterebbe modo di generare la sequenza di $\mathtt{1}$ a destra della variabile iniziale S.

10. $w = \mathtt{1}^i\mathtt{+}\mathtt{1}^j\mathtt{=}\mathtt{1}^h$ con $h < i + j$: si ha una contraddizione, perché riducendo la stringa si otterrebbe:
$$\mathtt{1}^i\mathtt{+}\mathtt{1}^j\mathtt{=}\mathtt{1}^h \mapsto \mathtt{1}^i\mathtt{+}\mathtt{1}^j B\mathtt{1}^h \overset{*}{\mapsto} \mathtt{1}^i\mathtt{+}\mathtt{1}\mathtt{1}^{j-h}B\mathtt{1}$$
e poiché $j - h > 0$ non ci sarebbe modo di ridurre ulteriormente applicando la regola A.

Poiché ogni motivo di esclusione di w da C porta ad una contraddizione, se ne conclude che $w \in C$ per ogni $S \Rightarrow w$.

Esercizio 5 [11] Il problema Set Cover è il seguente: dato un insieme U di elementi, una collezione $\mathcal{C} = \{S_1, \ldots, S_m\}$ di sottoinsiemi di U ($S_i \subseteq U$ per ogni i), ed un intero $k > 0$, determinare se è possibile selezionare k sottoinsiemi $S_{i_1}, \ldots S_{i_k}$ nella collezione $\mathcal{C}$ tali che $\cup_{j=1}^{k} S_{i_j} = U$. Dimostrare che Set Cover è NP-completo descrivendo una riduzione polinomiale da Vertex Cover.

Soluzione: Per prima cosa dimostriamo che Set Cover (SC) appartiene a NP. Il problema è polinomialmente verificabile perché ogni istanza $\langle U, S_1, \ldots, S_m, k\rangle$ che fa parte del linguaggio ha come certificato la collezione di sottoinsiemi che ricopre U: è certamente di dimensione non superiore alla collezione di tutti i sottoinsiemi in $\mathcal{C} = \{S_1, \ldots, S_m\}$, e verificare che ogni elemento di U fa parte di uno dei sottoinsiemi può essere facilmente realizzato da un algoritmo che esegue in tempo polinomiale nella dimensione dell'istanza del problema:

M= " On input $\langle U, \mathcal{C}, k, S'_1, \ldots, S'_h\rangle$, where $\mathcal{C}$ is a collection of subsets $S_i \subseteq U$:
1. if $h > k$ or $S'_j \notin \mathcal{C}$ for any j then reject
2. for each element $x \in U$:
    3. for each S'_j ($1 \le j \le h$):
        4. if $x \in S'_j$ then continue with next element in step 2
    5. Reject, because element x is not covered by any subset S'_j
6. Accept, because all elements in U are covered by the subsets S'_j "

Il numero di passi totali eseguito dall'algoritmo è in $\mathrm{O}(|U| \cdot h \cdot \max_{j=1}^{h} \left|S'_j\right|) = \mathrm{O}(|U|^2 \cdot h)$, ossia è polinomiale nella dimensione dell'istanza $\mathrm{O}(h \cdot |U|)$.

Consideriamo ora una riduzione polinomiale da Vertex Cover (VC) a SC. Sia $(G = (V, E), k)$ una istanza di VC. La corrispondente istanza di SC ha come insieme U un elemento x_e per ciascun arco e di G ($U = \{x_e \mid e \in E(G)\}$), e come collezione $\mathcal{C}$ un sottoinsieme

S_v per ciascun nodo v di G contenente tutti gli elementi x_e tali che l'arco e è incidente su v: $\mathcal{C} = \{S_v \mid v \in V(G)\}$ con $S_v = \{x_e \in U \mid e = (v, w) \in E(G)\}$.

Supponiamo che $(G, k) \in$ VC; dunque esiste $V' \subseteq V$ tale che $|V'| \leq k$ e V' copre tutti gli archi di G. Consideriamo dunque come sottoinsiemi ricoprenti per SC quelli corrispondenti agli elementi di V'. Sia x_e un qualunque elemento di U: poiché $e \in E(G)$, esiste un nodo v in V' tale che $e = (v, w)$. Ma allora $x_e \in S_v$ e S_v è tra i sottoinsiemi selezionati. Dunque esistono al più k sottoinsiemi tra quelli in $\mathcal{C}$ che coprono U.

Per la direzione opposta, supponiamo che esista un sottoinsieme di al più k sottoinsiemi in $\mathcal{C}$ che coprono U, e siano S'_{v_1}, S'_{v_2}, ..., S'_{v_k}. Sia $V' = \{v_1, \ldots, v_k\}$. Per costruzione dell'istanza di SC, $V' \subseteq V(G)$ ed ovviamente $|V'| \leq k$. Dimostriamo che V' è un vertex cover. Sia dunque $e \in E(G)$ un qualunque arco di G; il corrispondente elemento $x_e \in U$ deve far parte di almeno uno dei sottoinsiemi S'_{v_i}. Per costruzione di S'_{v_i} allora $e = (v_i, w)$, e dunque il nodo $v_i \in V'$ copre l'arco e.

Abbiamo dunque dimostrato che la trasformazione da (G, k) a $(U, S_1, \ldots, S_m, k)$ è una riduzione tra problemi. È inoltre evidente che tale trasformazione può essere costruita in tempo polinomiale. Pertanto, VC $\leq_m$ SC, e quindi SC è NP-hard, e di conseguenza NP-completo.