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

Facoltà di Ingegneria, Università degli Studi di Roma Tor Vergata

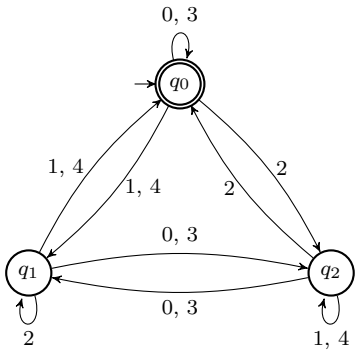
Compito scritto del 19 gennaio 2022

Esercizio 1 [5] Determinare un automa a stati finiti deterministico (DFA) per il linguaggio contenente la stringa vuota e tutti i numeri in base 5 multipli di 3.

Soluzione: L'automa a stati finiti richiesto può essere costruito considerando che l'unica informazione necessaria da memorizzare è il valore modulo 3 del numero rappresentato dalle cifre lette man mano. In altri termini, se v è il valore del numero rappresentato dalle cifre in base 5 lette fino ad un certo punto, e la successiva cifra letta è $b \in \{0, \ldots, 4\}$, allora il nuovo valore rappresentato dalle cifre lette sarà $v' = v \times 5 + b$; tuttavia, è sufficiente memorizzare negli stati dell'automa soltanto il resto della divisione per tre, in quanto $v' \bmod 3 = ((v \bmod 3) \times 5 + b) \bmod 3$. Si ha dunque la seguente tabella:

$v \bmod 3$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
0	0	1	2	0	1
1	2	0	1	2	0
2	1	2	0	1	2

È ora immediato derivare dalla tabella il seguente DFA:

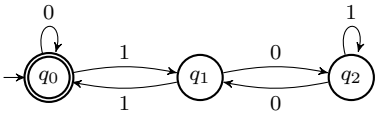


Si osservi che, come richiesto, l'automa accetta anche la stringa vuota ε.

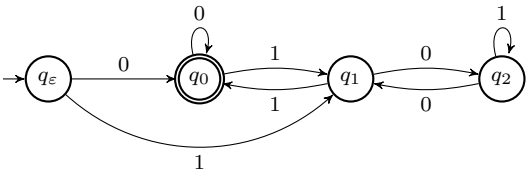
Esercizio 2 [5] Determinare una espressione regolare per il linguaggio contenente tutti i numeri in base 2 multipli di 3.

Soluzione: L'espressione regolare può essere derivata costruendo un automa a stati finiti che accetta le stringhe del linguaggio. Si osservi che il linguaggio non contiene la stringa vuota, in quanto questa non rappresenta alcun numero. Abbiamo due possibili alternative: (1) derivare l'espressione regolare da un automa che accetta esattamente tutti e soli gli elementi del linguaggio, oppure (2) derivare l'espressione di un automa che accetta anche la stringa vuota, poi modificare tale espressione regolare in modo da escludere ε.

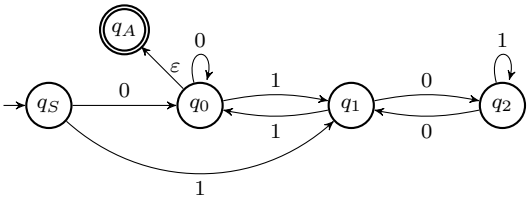
Un automa che accetta i numeri binari multipli di 3 *e* la stringa vuota deve semplicemente memorizzare i valori modulo 3 del numero man mano letto, assumendo inizialmente che la stringa vuota corrisponda al numero zero. Quindi:

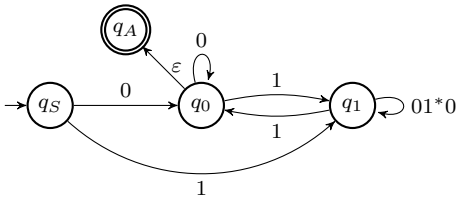


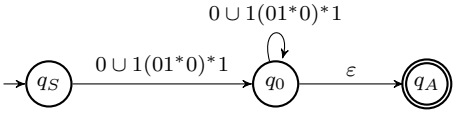
Per escludere la stringa vuota è necessario introdurre uno stato iniziale q_ε non accettante:

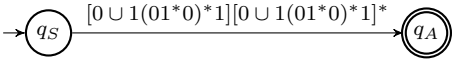


Convertendo in GNFA e rimuovendo i vari nodi si ottiene:









Quindi l'espressione regolare per i numeri binari multipli di 3 è $[0 \cup 1(01^*0)^*1]^+$. Si osservi che utilizzando il secondo approccio ed eliminando i nodi nello stesso ordine si sarebbe ottenuta l'espressione $[0 \cup 1(01^*0)^*1]^*$, la cui unica differenza consiste nel generare anche la stringa vuota ε.

Esercizio 3 [6] Si consideri il linguaggio $\mathcal{L} = \{x\, y^n\, x\, y\, x^n\, y \mid x, y \in \{0,1\}, x \neq y, n > 0\}$. $\mathcal{L}$ è un linguaggio libero dal contesto? Giustificare la risposta con una dimostrazione.

Soluzione: Il linguaggio $\mathcal{L}$ è libero dal contesto (CFL). Per dimostrarlo si può, in alternativa, esibire un PDA che riconosca gli elementi di $\mathcal{L}$ oppure esibire una grammatica G tale che $L(G) = \mathcal{L}$.

Consideriamo la seguente grammatica G:

$$S \to 0A1 \mid 1B0 \quad A \to 1A0 \mid 1010 \quad B \to 0B1 \mid 0101.$$

Dimostriamo che $L(G) = \mathcal{L}$, ossia che $\mathcal{L} \subseteq L(G)$ e $L(G) \subseteq \mathcal{L}$.

Sia $w \in \mathcal{L}$, e dimostriamo per induzione che $w \in L(G)$. I più piccoli elementi di $\mathcal{L}$ si ottengono per $n = 1$, e dunque sono 010101 e 101010. Entrambi possono essere derivati da S:

$$S \Rightarrow 0\,A\,1 \Rightarrow 0\,1010\,1 \qquad S \Rightarrow 1\,B\,0 \Rightarrow 1\,0101\,0$$

Dunque $\{010101, 101010\} \subseteq L(G)$. Supponiamo ora che l'ipotesi induttiva valga per tutti gli elementi di $\mathcal{L}$ fino al valore $n = N$, e dimostriamo che essa vale anche per $n = N + 1$. Sia dunque $w \in \mathcal{L}$ della forma $xy^{N+1}xyx^{N+1}y$. Per semplificare la spiegazione, supponiamo anche che $x = 0$ e $y = 1$. Poiché vale l'ipotesi induttiva, $01^N010^N1 \in L(G)$, dunque $S \Rightarrow^* 01^N010^N1$. Ora la prima regola applicata deve essere necessariamente $S \to 0A1$, altrimenti non potrebbe essere possibile generare il primo simbolo 0 della stringa. Poiché inoltre A può generare solo stringhe terminali o stringhe contenenti A, l'ultima regola applicata deve essere stata $A \to 1010$. La sottostringa 1010 occorre in un solo punto, dunque si ha:

$$S \Rightarrow 0\,A\,1 \Rightarrow^* 0\,1^{N-1}\,A\,0^{N-1}\,1 \Rightarrow 0\,1^{N-1}\,1010\,0^{N-1}\,1$$

Consideriamo ora la derivazione da S identica a questa, ma in cui l'ultima regola applicata è sostituita dalle due regole, in successione, $A \to 1A0$ e $A \to 1010$:

$$S \Rightarrow 0\,A\,1 \Rightarrow^* 0\,1^{N-1}\,A\,0^{N-1}\,1 \Rightarrow 0\,1^{N-1}\,1\,A\,0\,0^{N-1}\,1 \Rightarrow 0\,1^{N-1}\,1\,1010\,0\,0^{N-1}\,1$$

Pertanto, $S \Rightarrow^* 01^{N+1}010^{N+1}1$. A causa della simmetria delle regole della grammatica, l'identico ragionamento si applica nel caso in cui $x = 1$ e $y = 0$, utilizzando le regole coinvolgenti il simbolo B. Pertanto resta dimostrato che la stringa $w = xy^{N+1}xyx^{N+1}y \in L(G)$. Per induzione dunque si ha che $\mathcal{L} \subseteq L(G)$.

Dimostriamo ora che ogni stringa terminale generata dalla grammatica è inclusa in $\mathcal{L}$. Supponiamo che la prima regola applicata da S sia $S \to 0A1$. A causa della forma delle regole per espandere A, qualunque stringa terminale derivata da $0A1$ deve essere costituita da un certo numero $N \geq 0$ di applicazioni della regola $A \to 1A0$, seguite al termine dall'applicazione della regola $A \to 1010$. Si ha perciò:

$$S \Rightarrow 0A1 \Rightarrow^* 0\,1^N\,A\,0^N\,1 \Rightarrow 0\,1^N\,1010\,0^N\,1$$

Tale stringa terminale appartiene ad $\mathcal{L}$ (ponendo nella definizione del linguaggio $x = 0$, $y = 1$ e $n = N + 1 > 0$). Per la simmetria della grammatica e della definizione del linguaggio, lo stesso ragionamento si applica quando la prima regola applicata da S è $S \to 1B0$. Pertanto $L(G) \subseteq \mathcal{L}$.

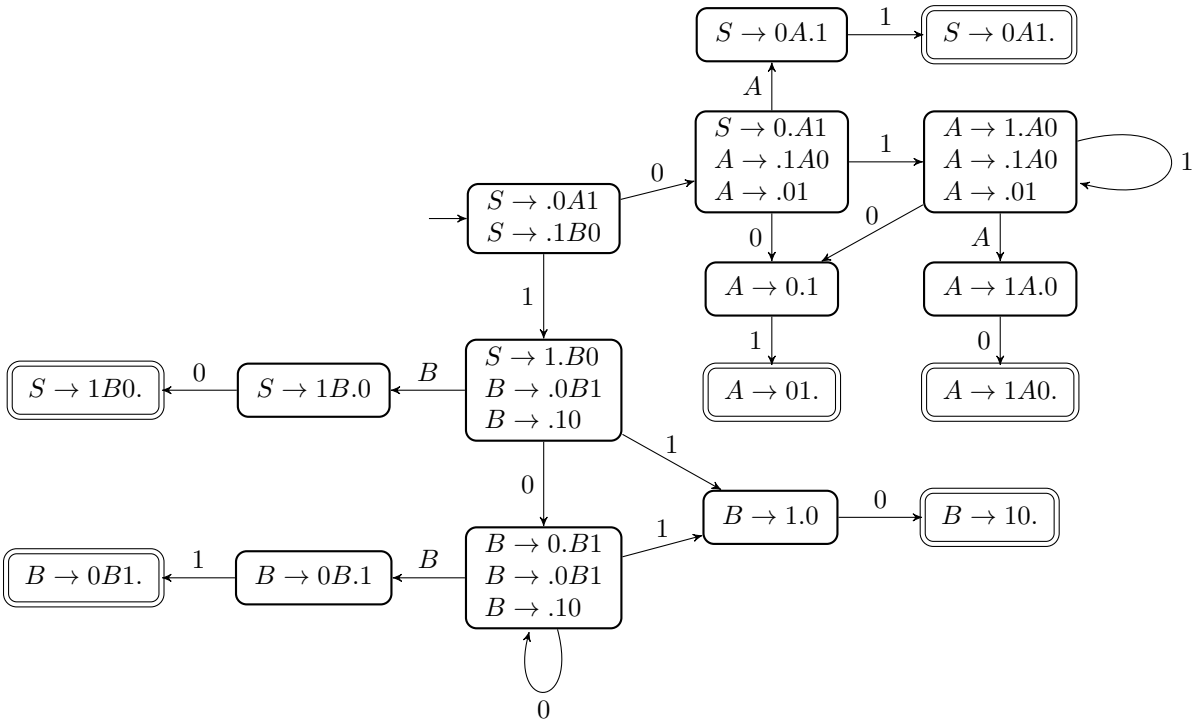
Concludiamo che $\mathcal{L} = L(G)$, e quindi che $\mathcal{L}$ è CFL.

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

$$S \to 0A1 \mid 1B0 \quad A \to 1A0 \mid 01 \quad B \to 0B1 \mid 10.$$

La grammatica è deterministica? Giustificare la risposta con una dimostrazione.

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



Poiché tutti gli stati finali dell'automa DK contengono una sola regola, la grammatica è deterministica.

Esercizio 5 [8] Siano A e B linguaggi Turing-riconoscibili (ricorsivamente enumerabili). Dimostrare che sia $A\#B = \{x \# y \mid x \in A,\, y \in B\}$ (ove $\#$ è un simbolo non incluso in A o B) che $A\,B = \{x\,y \mid x \in A,\, y \in B\}$ sono Turing-riconoscibili.

Soluzione: Poiché sia A che B sono ricorsivamente enumerabili, esistono due TM M_A e M_B che riconoscono le stringhe di A e B, rispettivamente. È possibile costruire due diverse TM, basate su M_A e M_B, la prima deterministica per riconoscere se una stringa è in $A\#B$ e la seconda nondeterministica per riconoscere se una stringa in ingresso è in $A\,B$ (si osservi che il nondeterminismo, utilizzato per semplicità per "indovinare" una suddivisione della stringa in ingresso in due sottostringhe x e y, non costituisce un problema, perché per ogni TM non deterministica esiste una TM deterministica equivalente).

Per sottolineare come i due linguaggi $A\#B$ e $A\,B$ siano affini, svolgiamo l'esercizio utilizzando una tecnica di dimostrazione alternativa. Poiché sia A che B sono ricorsivamente enumerabili, esistono due TM E_A e E_B che stampano in uscita tutte e sole le stringhe dei linguaggi A e B, rispettivamente. Consideriamo dunque la seguente TM E [ovvero E'] che enumera gli elementi di $A\#B$ [ovvero gli elementi di $A\,B$]:

E [E'] = "On any input:
1. Ignore the input
2. for $k = 1$ to ∞:
3. Emulate the TM E_A for k steps
4. For any string x printed by E_A:
5. Emulate the TM E_B for k steps
6. For any string y printed by E_B:
7. Print the string $x\#y$ [the string $x\,y$]"

Si osservi che gli enumeratori in linea di principio non terminano mai, quindi non è corretto inserire nell'algoritmo una emulazione senza limiti sul numero di passi. In questo caso tipicamente l'enumeratore stamperebbe *soltanto* le stringhe $\overline{x}\#y$ [ovvero $\overline{x}\,y$], ove $y \in B$ ma $\overline{x}$ è la *prima* stringa stampata da E_A.

È immediato verificare che ogni stringa stampata dall'enumeratore fa parte del linguaggio, in quanto composta da sottostringhe stampate da E_A e E_B. D'altra parte, supponiamo che $w \in A\#B$ [ovvero $w \in A\,B$]. Dunque w contiene due sottostringhe $x \in A$ e $y \in B$. L'enumeratore E_A stamperà certamente x dopo k_A di passi, mentre l'enumeratore E_B stamperà y dopo k_B passi. Perciò la stringa w verrà stampata dall'emulatore E [E'] nella iterazione corrispondente a $k = \max\{k_A, k_B\}$. Dunque E enumera $A\#B$, E' enumera $A\,B$, e pertanto entrambi i linguaggi sono ricorsivamente enumerabili.

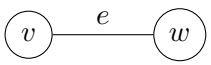
Esercizio 6 [10] Il problema DOMINATING SET è il seguente: dato un grafo non diretto $G = (V, E)$, ed un numero intero k, determinare se esiste un sottoinsieme di nodi $V' \subseteq V$ di cardinalità k tale che ogni nodo del grafo o appartiene a V' oppure è adiacente ad un nodo in V' (o entrambe le cose). Dimostrare che DOMINATING SET è NP-completo.

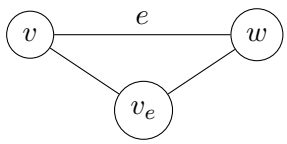
Soluzione: Per prima cosa dimostriamo che DOMINATING SET (DS) appartiene a NP. Il problema è polinomialmente verificabile perché ogni istanza che fa parte del linguaggio ha come certificato il sottoinsieme di nodi del grafo che costituisce il dominating set: è certamente di dimensione non superiore al numero di nodi del grafo, e verificare che ogni nodo del grafo fa parte od è adiacente al sottoinsieme può essere facilmente realizzato da un algoritmo che esegue in tempo polinomiale nella dimensione dell'istanza del problema:

M= "On input $\langle G = (V, E), k, V' \rangle$, where G is a graph, $V' \subseteq V$:
1. if $|V'| > k$: reject
2. for each node v in V:
3. if $v \in V'$ then continue with next node in step 1
4. for each edge $e \in E$:
5. if e links v to a node in V', continue with next node in step 1
6. Reject, because node v is not dominated by V'
7. Accept, because all nodes in V are dominated by V'"

Il numero totale di passi eseguiti dall'algoritmo è $\mathrm{O}(n\,m\,n) = \mathrm{O}(n^4)$, ove $n = |V(G)|$ e $m = |E(G)| = \mathrm{O}(n^2)$.

Consideriamo ora una riduzione polinomiale da VERTEX COVER (VC) a DS. Sia $(G = (V, E), k)$ una istanza di VC. Costruiamo un nuovo grafo $G' = (V', E')$ in questo modo: $V' = V \cup V_E$ è costituito dai nodi di G e da un nodo v_e per ciascun arco $e \in E$; in totale quindi $|V'| = n + m = |V| + |E|$. $E' = E \cup E_V$ è costituito dagli archi di G e, per ciascun nodo $v_e \in V_E$, da due archi (v_e, v) e (v_e, w) ove $e = (v, w)$; in totale quindi $|E'| = 3m = 3\,|E|$. Possiamo dimostrare che $(G, k) \in$ VC se e solo se $(G', k + s) \in$ DS, ove s è il numero di nodi $S \subseteq V$ "isolati" (senza archi incidenti).





Supponiamo che $(G, k) \in$ VC; dunque esiste $U \subseteq V$ tale che $|U| = k$ ed ogni arco di G è incidente ad almeno un nodo di U. Consideriamo il sottoinsieme di nodi $U' = U \cup S$ in G' (esiste perché tutti i nodi di G sono anche nodi di G'), e dimostriamo che è un dominating set di dimensione $k + s$. Infatti, sia $x \in V(G') = V \cup V_E$: se $x \in V$, allora o $x \in S$, e dunque $x \in U'$, oppure esiste un arco $e \in E$ incidente su x. Poiché U è un ricoprimento degli archi in G, esiste un nodo $y \in U$ tale che $e = (x, y)$; pertanto, il nodo x è dominato dal nodo $y \in U'$. Se invece $x \in V_E$, allora $x = v_e$ per un certo arco $e \in E$: dunque, U deve contenere un nodo y che ricopre e; allora, per costruzione di E_V, $(y, v_e) \in E_V$, e quindi x è dominato da $y \in U'$.

Per la direzione opposta, supponiamo che $(G', k + s) \in$ DS, e quindi esiste un sottoinsieme U', con $|U'| = k + s$, che domina ogni nodo di G'. Risulta evidente che $S \subseteq U'$, perché l'unico

modo per dominare nodi isolati è inserirli nel sottoinsieme dominante. Un'altra osservazione è che se U' contiene un qualunque nodo $v_e \in V_E$, è possibile sostituire in U' il nodo v_e con uno qualunque dei due nodi v, w tali che $e = (v, w)$. Infatti per costruzione di G' il nodo v_e può dominare solo se stesso, v e w; ma v (o equivalentemente w) domina almeno se stesso, w e v_e, quindi sostituendo v_e con v si ottiene un dominating set di dimensione pari od inferiore a quella di U' che domina almeno lo stesso insieme di nodi di G'. Consideriamo dunque il dominating set U'' in cui ogni nodo v_e è stato sostituito da un nodo in V come appena descritto, e sia $U = U'' \setminus S$. Il sottoinsieme U ha cardinalità $\leq k$, è un sottoinsieme di nodi di G, ed è un ricoprimento degli archi in G. Infatti, sia $e \in E$, con $e = (v, w)$. Poiché il nodo v_e deve essere dominato da qualche nodo in U'', per costruzione v, w, od entrambi appartengono al sottoinsieme dominante U''. Naturalmente $v, w \notin S$, quindi v, w oppure entrambi, appartengono ad U. Pertanto, l'arco e risulta ricoperto da un elemento di U.

Abbiamo dunque dimostrato che la trasformazione da (G, k) a $(G', k + s)$ è una riduzione tra problemi. È inoltre evidente che tale trasformazione può essere costruita in tempo polinomiale. Pertanto, VC $\leq_m$ DS, e quindi DS è NP-hard in quanto VC è NP-completo. Ciò conclude la dimostrazione di NP-completezza di DS.