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

Facoltà di Ingegneria, Università degli Studi di Roma Tor Vergata

Compito scritto del 21 settembre 2023

Esercizio 1 [7] Determinare un automa a stati finiti deterministico (DFA) per il linguaggio contenente tutti i numeri in base 3 multipli di 4, ovvero dimostrare che tale automa non esiste.

Soluzione: L'alfabeto di ingresso dell'automa richiesto è costituito dalle cifre '`0`', '`1`' e '`2`'. L'automa deve essenzialmente "memorizzare" tramite i propri stati interni quale sia il resto della divisione tra il numero parzialmente letto in input ed il valore quattro. Quindi sia q_j, con j compreso tra 0 e 3, lo stato interno che memorizza il resto j nella divisione per quattro. Se i simboli in ingresso terminano avendo l'automa nello stato interno q_0, allora il valore in ternario letto in ingresso è un multiplo di quattro; pertanto, q_0 è l'unico stato di accettazione.

Le transizioni da uno stato all'altro sono condizionate dal simbolo letto in ingresso. Per ogni successiva cifra letta il valore ottenuto precedentemente deve essere moltiplicato per tre (essendo appunto i numeri codificati in base 3), e poi al valore ottenuto va sommata la nuova cifra. Quindi le transizioni si ottengono nel seguente modo:

1. Nello stato q_0 il valore finora letto dall'input è un multiplo di quattro, ossia $4\,k$ per un generico $k \in \mathbb{N}$.
   - '`0`' Si ottiene il valore $3 \times (4\,k) = 4 \times (3\,k)$, quindi ancora un multiplo di quattro: si rimane in q_0.
   - '`1`' Si ottiene il valore $3 \times (4\,k) + 1 = 4 \times (3\,k) + 1$, quindi si transita in q_1.
   - '`2`' Si ottiene il valore $3 \times (4\,k) + 2 = 4 \times (3\,k) + 2$, quindi si transita in q_2.
2. Nello stato q_1 il valore finora letto dall'input è a distanza uno da un multiplo di quattro, ossia $4\,k + 1$ per un generico $k \in \mathbb{N}$.
   - '`0`' Si ottiene il valore $3 \times (4\,k + 1) = 4 \times (3\,k) + 3$, quindi si transita in q_3.
   - '`1`' Si ottiene il valore $3 \times (4\,k + 1) + 1 = 4 \times (3\,k + 1)$, quindi si transita in q_0.
   - '`2`' Si ottiene il valore $3 \times (4\,k + 1) + 2 = 4 \times (3\,k + 1) + 1$, quindi si rimane in q_1.
3. Nello stato q_2 il valore finora letto dall'input è a distanza due da un multiplo di quattro, ossia $4\,k + 2$ per un generico $k \in \mathbb{N}$.
   - '`0`' Si ottiene il valore $3 \times (4\,k + 2) = 4 \times (3\,k + 1) + 2$, quindi si rimane in q_2.
   - '`1`' Si ottiene il valore $3 \times (4\,k + 2) + 1 = 4 \times (3\,k + 1) + 3$, quindi si transita in q_3.
   - '`2`' Si ottiene il valore $3 \times (4\,k + 2) + 2 = 4 \times (3\,k + 2)$, quindi si transita in q_0.

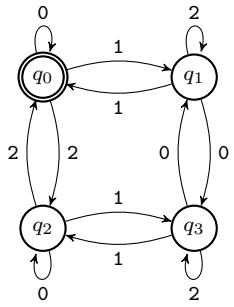
4. Nello stato q_3 il valore finora letto dall'input è a distanza tre da un multiplo di quattro, ossia $4\,k+3$ per un generico $k \in \mathbb{N}$.

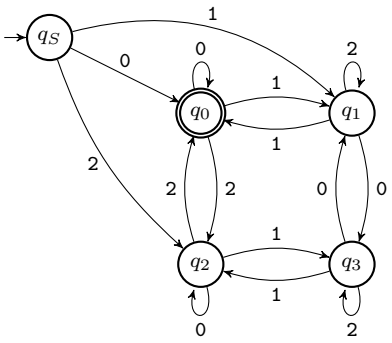
   '0' Si ottiene il valore $3 \times (4\,k+3) = 4 \times (3\,k+2) + 1$, quindi si transita in q_1.

   '1' Si ottiene il valore $3 \times (4\,k+3) + 1 = 4 \times (3\,k+2) + 2$, quindi si transita in q_2.

   '2' Si ottiene il valore $3 \times (4\,k+3) + 2 = 4 \times (3\,k+2) + 3$, quindi si rimane in q_3.

In conclusione si ottiene il seguente automa (a sinistra)





Lo stato iniziale non può però essere q_0, in quanto la stringa vuota ε non dovrebbe essere accettata: essa non rappresenta un numero in base 3. Pertanto, aggiungiamo all'automa uno stato iniziale q_S il cui scopo è leggere il primo simbolo del numero in ingresso. Si ottiene perciò il DFA a destra qui sopra, che rappresenta la soluzione dell'esercizio.

Esercizio 2 [6] Si consideri il linguaggio $A = \{x\texttt{=}y\texttt{-}z \mid x, y, z \in \{0,1\}^*$ e il numero binario x è la differenza dei numeri binari y e $z\}$. Ad esempio, $\texttt{10=11-1} \in A$, $\texttt{10=11-0} \notin A$. Il linguaggio A è regolare? Giustificare la risposta con una dimostrazione.

Soluzione: Il linguaggio A non è regolare, in quanto intuitivamente per decidere se una stringa appartiene o meno al linguaggio l'automa dovrebbe saper "contare", ossia valutare il valore numerico corrispondente alle sottostringhe x, y e z. Per dimostrare formalmente che A non è regolare utilizziamo il pumping lemma. Si assuma per assurdo che A sia regolare. Esiste allora una lunghezza $p > 0$ tale che ogni stringa s di lunghezza maggiore o uguale a p ammette una suddivisione "pompabile". Sia $s = \texttt{1}^p\texttt{=1}^p\texttt{-0}$. La stringa di lunghezza $2\,p+3$ appartiene evidentemente al linguaggio, perchè il numero codificato da $\texttt{1}^p$ (ossia $2^p - 1$) è presente sia a sinistra che a destra del simbolo '=', e a destra gli viene sottratto il valore zero. Il pumping lemma afferma dunque che deve esistere una suddivisione $s = uvw$ tale che $|uv| \leq p$, $|v| > 0$, e $uv^iw \in A$ per ogni $i \geq 0$. La condizione $|uv| \leq p$ implica che la sottostringa v deve essere interamente compresa nella porzione di simboli '1' a sinistra del simbolo '='; inoltre la condizione $|v| > 0$ implica che v deve includere almeno un simbolo '1'. Consideriamo dunque il caso $i = 0$: la stringa risultante è necessariamente nella forma

$\texttt{1}^q\texttt{=1}^p\texttt{-0}$, con $q < p$. Pertanto la uguaglianza numerica ($2^q - 1 = 2^p - 1 - 0$) non può essere vera, e quindi $uv^0w = uw \notin A$. Ciò contraddice l'asserto del pumping lemma. Pertanto A non è un linguaggio regolare.

Esercizio 3 [7] Derivare un automa a pila (PDA) per il linguaggio $A = \{x\texttt{\#}y \mid x, y \in \{\texttt{0},\texttt{1}\}^*$ e x è la codifica binaria $\langle|y|\rangle$ della lunghezza di $y\}$, ovvero dimostrare che non è possibile derivare un tale PDA.

Soluzione: In effetti non è possibile costruire un PDA per il linguaggio A. È tuttavia estremamente difficile dimostrare questo fatto ragionando direttamente sul PDA, perché si dovrebbe dimostrare che *qualsiasi* algoritmo concepibile (tra gli infiniti possibili) fallirebbe nel riconoscere gli elementi di A. Possiamo invece ottenere il risultato ricordando che ogni linguaggio riconosciuto da un PDA è CFL e riconoscendo che A non è CFL. Per dimostrare l'ultimo asserto utilizziamo il "pumping lemma" per i CFL.

Supponiamo per assurdo che A sia CFL. A causa del "pumping lemma", deve pertanto esistere una lunghezza $p > 0$ tale che, per ogni elemento s di lunghezza maggiore o uguale a p, s può essere suddiviso come $s = uvxyz$ in modo tale che $|vy| > 0$, $|vxy| \leq p$, e per ogni $i \geq 0$, $uv^ixy^iz \in A$. Consideriamo la stringa $s = \texttt{1}^p\texttt{\#0}^{2^p-1}$: è immediato verificare che $|s| > p$ e $s \in A$. Dimostriamo che per ogni possibile suddivisione di s esiste un valore di $i \geq 0$ per il quale la stringa "pompata" *non* fa parte di A. In effetti una dimostrazione abbastanza semplice può essere basata sul fatto che ogni simbolo `1` aggiunto a sinistra comporta almeno il raddoppio (in effetti la triplicazione) della lunghezza della stringa a destra; dunque il numero di simboli da aggiungere dovrebbe essere esponenziale in p, mentre la lunghezza della porzione y non può superare p. La stessa idea fondamentale viene sfruttata nella seguente dimostrazione formale in cui si utilizza esclusivamente il "pompaggio verso il basso".

Cominciamo con l'osservare che ogni suddivisione in cui $\texttt{'\#'} \notin x$ non può soddisfare il lemma. Infatti se $\texttt{'\#'} \notin x$, e considerando che $|vy| > 0$, allora o vxy deve essere tutto a sinistra del simbolo `#` oppure deve essere tutto alla sua destra (altrimenti il simbolo `#` sarebbe rimosso o duplicato, e la stringa non potrebbe far parte di A in quanto malformata). In entrambi i casi pompare verso il basso produce una stringa in cui la codifica binaria a sinistra non coincide con il numero di simboli a destra.

Assumiamo dunque che $\texttt{'\#'} \in x$, e pertanto $v \in \texttt{1}^*$ e $y \in \texttt{0}^*$. Analizziamo i seguenti casi basati sulla lunghezza della sottostringa v:

- $|v| = 0$: in questo caso, poiché $|vy| > 0$, deve valere $|y| > 0$. Pertanto pompando verso il basso, ossia considerando $i = 0$, si ottiene una stringa uxz in cui la codifica binaria a sinistra è identica ma il numero di elementi a destra è diminuito; tale stringa non può appartenere ad A.

- $|v| = 1$: consideriamo il valore $i = 0$, ossia la stringa $1^{p-1}\#0^l$ ottenuta pompando verso il basso, ove il valore l dipende dalla dimensione di y. Perché tale stringa possa far parte di A deve valere $l = 2^{p-1} - 1$, e quindi $|y| = 2^p - 1 - (2^{p-1} - 1) = 2^{p-1}$. Ma allora $|vxy| = 1 + 2^{p-1} + |x| \geq 2 + 2^{p-1}$; ricordando che $|vxy| \leq p$, si ottiene $2^{p-1} \leq p - 2$. In effetti non esiste alcun valore positivo per p che possa soddisfare tale disuguaglianza, e quindi non esiste suddivisione che possa conservare l'appartenza ad A pompando verso il basso.

- $|v| > 1$: si applica lo stesso ragionamento del caso precedente. Sia $j = |v|$, e consideriamo il valore $i = 0$. Necessariamente la lunghezza di y deve essere pari a $l = 2^p - 1 - (2^{p-j} - 1) = \left(1 - 2^{-j}\right) 2^p$. Pertanto: $p \geq |vxy| \geq j + 1 + \left(1 - 2^{-j}\right) 2^p$, e poiché $(1 - 2^{-j}) > 1/2$ per $j > 1$:

  $$2^{p-1} < 2^p \left(1 - 2^{-j}\right) \leq p - j - 1 < p - 1.$$

  Poiché questa disuguaglianza non ha alcuna soluzione per $p > 0$, non esiste alcuna suddivisione che possa conservare l'appartenza ad A pompando verso il basso.

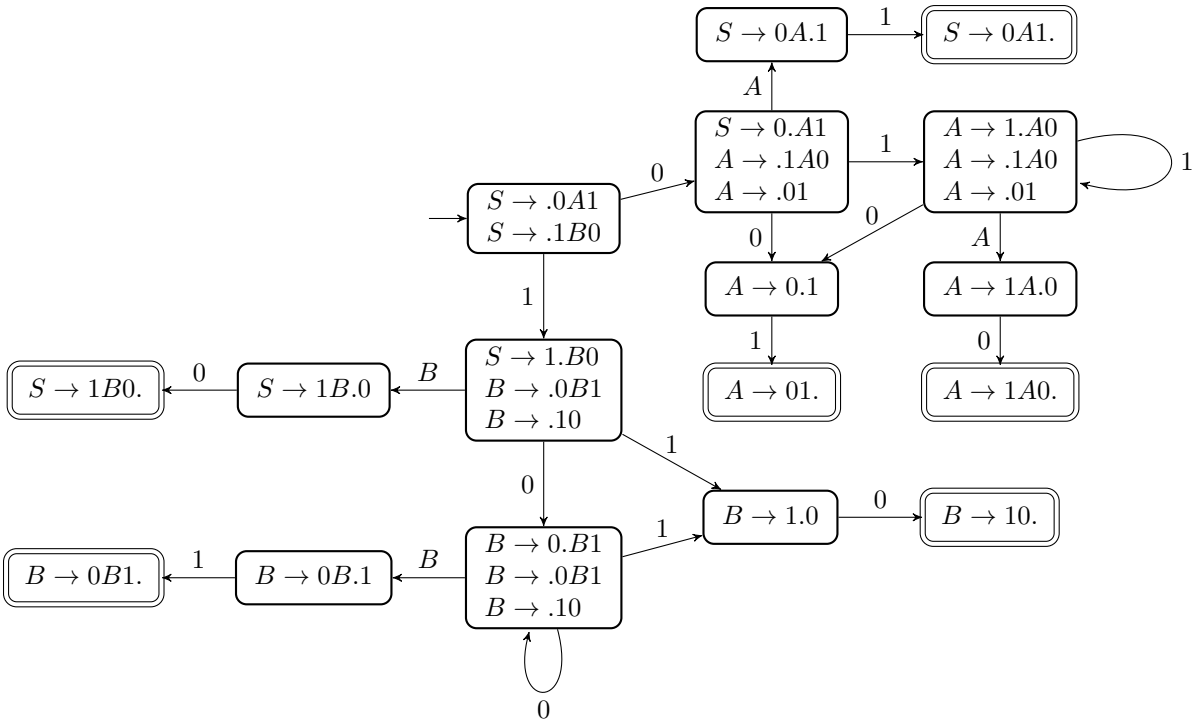
Poiché ogni possibile suddivisione di s risulta non appartente ad A quando pompata verso il basso, le conclusioni del "pumping lemma" non valgono. La contraddizione è dovuta alla supposizione che A sia CFL. Concludiamo pertanto che non esiste alcun PDA che possa riconoscere gli elementi del linguaggio A.

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 completata, la grammatica è deterministica: infatti ovviamente negli stati accettanti non esistono né diverse regole completate né regole non completate in cui il dot precede un simbolo terminale.

Esercizio 5 [7] Dimostrare che la concatenazione di linguaggi Turing-riconoscibili (ossia ricorsivamente enumerabili) è un linguaggio Turing-riconoscibile.

Soluzione: Siano A e B linguaggi ricorsivamente enumerabili; dobbiamo dimostrare che $AB = \{xy \,|\, x \in A, y \in B\}$ è ricorsivamente enumerabile. 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.

Consideriamo la seguente NTM M:

M= "On input w:

1. Guess a subdivision x, y of w: $w = xy$
2. Emulate the TM M_A on input x
3. Emulate the TM M_B on input y
4. If both $M_A(x)$ and $M_B(x)$ accepted, then accept
5. Otherwise, reject"

Supponiamo che $M(w)$ accetti; di conseguenza esiste una computazione accettante che "indovina" una opportuna suddivisione $xy = w$, ed emula $M_A(x)$ e $M_B(y)$. Poiché $M(w)$ termina accettando, le due emulazioni devono terminare ed entrambe le computazioni devono accettare. Pertanto $x \in A$ e $y \in B$, e quindi $w \in A\,B$. D'altra parte, supponiamo che $w \in A\,B$; dunque esiste una suddivisione $w = xy$ tale che $x \in A$ e $y \in B$. Questa suddivisione corrisponde ad un ramo dell'albero di computazioni di $M(w)$; inoltre poiché M_A riconosce A, $M_A(x)$ termina accettando; analogamente $M_B(y)$ termina accettando. Perciò $M(w)$ accetta.

Si osservi che il non-determinismo di M non costituisce un problema, perché per ogni TM non deterministica esiste una TM deterministica equivalente. Perciò il linguaggio $A\,B$ è ricorsivamente enumerabile.

Esercizio 6 [7] Sia $\mathcal{L} = \{\langle M\rangle \mid M$ è una macchina di Turing deterministica tale che esiste almeno una stringa di input x per la quale $M(x)$ non termina$\}$. Dimostrare che $\mathcal{L}$ non è decidibile.

Soluzione: Osserviamo innanzi tutto che non è possibile ricorrere al Teorema di Rice per risolvere l'esercizio, poiché la proprietà che caratterizza il linguaggio $\mathcal{L}$ non è propria del linguaggio delle macchine di Turing che appartengono a $\mathcal{L}$. Infatti, sia M una macchina di Turing che termina su ogni input e rifiuta almeno una stringa $\overline{x}$. È facile derivare da M una TM M' che per ogni input x dapprima controlla se $x = \overline{x}$, ed in tal caso entra in un ciclo senza fine; altrimenti, se $x \neq \overline{x}$, $M'(x)$ simula $M(x)$. Pertanto $L(M) = L(M')$, ma $\langle M\rangle \notin \mathcal{L}$ mentre $\langle M'\rangle \in \mathcal{L}$.

Per dimostrare che $\mathcal{L}$ è non decidibile è sufficiente mostrare una riduzione da un problema indecidibile; è naturale prendere in considerazione $\mathcal{A}_{\mathrm{TM}}$. Possiamo in effetti mostrare che $\mathcal{A}_{\mathrm{TM}} \leq_{\mathrm{T}} \mathcal{L}$ tramite la seguente TM con oracolo $\mathcal{L}$ che decide $\mathcal{A}_{\mathrm{TM}}$:

$M^{\mathcal{L}}$= "On input $\langle T, x\rangle$, where T is a TM and x is a string:

1. Build from $\langle T, x\rangle$ the following TM:

   M'= "On input y, where y is a string:
   (a) If $y \neq x$, then halt
   (b) Run T on input x"
   (c) Halt"

2. Ask the oracle whether $\langle M'\rangle \in \mathcal{L}$
3. If the answer is YES, reject
4. Otherwise, if the answer is NO, run T on input x
5. If $T(x)$ accepts, then accept; otherwise, reject"

Sia $\langle T, x\rangle \in \mathcal{A}_{\mathrm{TM}}$; dunque $M'(y)$ termina sempre (al passo (a) se $y \neq x$, al passo (c) se $y = x$). Perciò $\langle M'\rangle \notin \mathcal{L}$, e quando la computazione $M^{\mathcal{L}}(\langle T, x\rangle)$ interroga l'oracolo, questo risponderà

NO. Dunque $M^{\mathcal{L}}(\langle T, x\rangle)$ simula $T(x)$ ed accetta poiché $T(x)$ accetta. Supponiamo invece che $\langle T, x\rangle \notin \mathcal{A}_{\mathrm{TM}}$. Esistono due casi: $T(x)$ rifiuta e $T(x)$ entra in un ciclo senza fine senza terminare. Nel caso in cui $T(x)$ rifiuta, l'oracolo risponde NO perché $T(x)$ termina, quindi non esiste alcun input sul quale M' non termina. $M^{\mathcal{L}}(\langle T, x\rangle)$ arriva dunque a simulare $T(x)$ e, quando quest'ultima computazione termina rifiutando, rifiuta. Se infine $T(x)$ non termina, $M'(x)$ non termina, dunque $\langle M'\rangle \in \mathcal{L}$ e l'oracolo risponde YES; perciò $M^{\mathcal{L}}(\langle T, x\rangle)$ rifiuta al passo 3. Riassumendo, $M^{\mathcal{L}}$ decide $\mathcal{A}_{\mathrm{TM}}$, e quindi $\mathcal{A}_{\mathrm{TM}} \leq_{\mathrm{T}} \mathcal{L}$; poiché $\mathcal{A}_{\mathrm{TM}}$ è indecidibile, resta dimostrato che anche $\mathcal{L}$ è indecidibile.