Automi e Linguaggi Formali – 20/4/2023
Simulazione Prima prova intermedia – Soluzione

1. (12 punti) Se L è un linguaggio e a un simbolo, allora L/a, il *quoziente* di L e a, è l'insieme delle stringhe

$$L/a = \{w \mid wa \in L\}.$$

Per esempio, se $L = \{0, 001, 100\}$, allora $L/0 = \{\varepsilon, 10\}$. Dimostra che se L è regolare allora anche L/a è regolare.

Soluzione: Se L è un linguaggio regolare, allora sappiamo che esiste un DFA $A = (Q, \Sigma, \delta, q_0, F)$ che riconosce L. Data una parola $wa \in L$ dove $w = w_1 \dots w_n$, la computazione di A su aw è una sequenza di stati $r_0 r_1 \dots r_{n+1}$ tali che:

- $r_0 = q_0$;
- $\delta(r_{i-1}, w_i) = r_i$ per ogni $i = 1, \dots, n$;
- $\delta(r_n, a) = r_{n+1}$;
- $r_{n+1} \in F$.

Data questa osservazione possiamo costruire un automa A' che accetta il linguaggio L/a cambiando gli stati finali di A. Formalmente, $A' = (Q, \Sigma, \delta, q_0, F')$ dove Q, Σ, δ e q_0 sono gli stessi di A e l'insieme degli stati finali contiene tutti gli stati che raggiungono uno stato finale di A dopo aver consumato a:

$$F' = \{q \mid \delta(q, a) \in F\}.$$

In questo modo abbiamo che per ogni $wa \in L$ la sequenza di stati $r_0 \dots r_n$ descritta sopra è una computazione di A' che accetta w (perché r_n diventa uno stato finale di A'), ed abbiamo dimostrato che se $wa \in L$ allora $w \in L(A')$. Viceversa, se $w \in L(A')$ allora esiste una computazione $s_0 \dots s_n$ di A' tale che:

- $s_0 = q_0$;
- $\delta(s_{i-1}, w_i) = s_i$ per ogni $i = 1, \dots, n$;
- $s_n \in F'$.

Di conseguenza, $s_{n+1} = \delta(s_n, a) \in F$ e la sequenza di stati $s_1 \dots s_{n+1}$ è una computazione di A su wa, ed abbiamo dimostrato che se $w \in L(A')$ allora $wa \in L$. Quindi possiamo concludere che il linguaggio di A' è precisamente L/a, come richiesto.

2. (12 punti) Considera il linguaggio

$$L_2 = \{1^n w \mid w \text{ è una stringa di 0 e 1 di lunghezza } n\}.$$

Dimostra che L_2 non è regolare.

Soluzione: Usiamo il Pumping Lemma per dimostrare che il linguaggio non è regolare. Supponiamo per assurdo che L_2 sia regolare:

- sia k la lunghezza data dal Pumping Lemma;
- consideriamo la parola $w = 1^k 0^k$, che appartiene ad L_2 ed è di lunghezza maggiore di k;
- sia $w = xyz$ una suddivisione di w tale che $y \neq \varepsilon$ e $|xy| \leq k$;
- poiché $|xy| \leq k$, allora x e y sono entrambe contenute nella sequenza iniziale di 1. Inoltre, siccome $y \neq \varepsilon$, abbiamo che $x = 1^q$ e $y = 1^p$ per qualche $q \geq 0$ e $p > 0$. z contiene la parte rimanente della stringa: $z = 1^{k-q-p}0^k$. Consideriamo l'esponente $i = 0$: la parola xy^0z ha la forma

  $$xy^0z = xz = 1^q 1^{k-q-p} 0^k = 1^{k-p} 0^k$$

  Poiché $p > 0$, la sequenza iniziale di 1 è più corta della sequenza finale di 0, e quindi la parola iterata xy^0z non può essere scritta nella forma $1^n w$ con $n = |w|$ perché non contiene abbastanza 1 nella parte iniziale.

Abbiamo trovato un assurdo quindi L_2 non può essere regolare.

Nota: per questo esercizio scegliere un esponente $i > 1$ non è corretto, perché se p è pari allora la parola xy^iz appartiene al linguaggio L_2, in quanto può essere scritta come $1^{k+ip}0^k = 1^k 1^{ip/2} 1^{ip/2} 0^k = 1^{k+ip/2} w$ con $w = 1^{ip/2}0^k$ parola di lunghezza $k + ip/2$.

3. (12 punti) Una CFG è detta *lineare a destra* se il corpo di ogni regola ha al massimo una variabile, e la variabile si trova all'estremità di destra. In altre parole, tutte le regole di una grammatica lineare a destra sono nella forma $A \to wB$ o $A \to w$, dove A e B sono variabili e w è una stringa di zero o più simboli terminali.

Dimostra che ogni grammatica lineare a destra genera un linguaggio regolare. *Suggerimento:* costruisci un ε-NFA che simula le derivazioni della grammatica.

Soluzione: Data una grammatica lineare a destra $G = (V, \Sigma, R, S)$, possiamo notare che le derivazioni di G hanno una struttura particolare. Siccome le regole sono nella forma $A \to wB$ o $A \to w$, dove A e B sono variabili e w è una stringa di zero o più simboli terminali, ogni derivazione di una parola della grammatica sarà formata da n applicazioni di una regola del tipo $A \to wB$ seguita da un'applicazione finale di una regola $A \to w$. A partire da questa osservazione possiamo costruire un ε-NFA $A = (Q, \Sigma, \delta, q_0, F)$ che simula le derivazioni della grammatica. Per rendere più chiara la costruzione usiamo una notazione abbreviata per la funzione di transizione, che ci fornisce un modo per far consumare un'intera stringa all'automa in un singolo passo. Possiamo simulare queste transizioni estese introducendo stati aggiuntivi per consumare la stringa un simbolo alla volta. Siano p e q stati dell'ε-NFA e sia $w = w_1 \dots w_n$ una stringa sull'alfabeto Σ. Per fare in modo che l'automa vada da p a q consumando l'intera stringa w introducendo nuovi stati intermedi $r_1 \dots r_{n-1}$ e definendo la funzione di transizione come segue:

$$\begin{aligned} \delta(p, w_1) &\text{ contiene } r_1, \\ \delta(r_1, w_2) &= \{r_2\}, \\ &\dots \\ \delta(r_{n-1}, w_n) &= \{q\}. \end{aligned}$$

Useremo la notazione $q \in \delta(p, w)$ per denotare quando l'automa può andare da p a q consumando la stringa di input w. Gli stati dell'automa sono $Q = V \cup \{q_f\} \cup E$ dove V è l'insieme delle variabili della grammatica, q_f è l'unico stato finale dell'automa e E è l'insieme di stati necessari per realizzare le transizioni estese appena descritte. Lo stato iniziale è S, variabile iniziale della grammatica. La funzione di transizione è definita come segue:

- $B \in \delta(A, w)$ per ogni regola $A \to wB$ della grammatica;
- $q_f \in \delta(A, w)$ per ogni regola $A \to w$ della grammatica.

L'automa A che abbiamo costruito è in grado di riconoscere lo stesso linguaggio della grammatica G perché simula le derivazioni di G nel modo descritto di seguito. L'automa inizia la computazione dallo stato che corrisponde alla variabile iniziale di G, e ripete i seguenti passi:

(a) se lo stato corrente corrente corrisponde alla variabile A, sceglie non deterministicamente una delle regole per A;
(b) se la regola scelta è $A \to wB$, prova a consumare la stringa w dall'input. Se ci riesce va nello stato B, altrimenti la computazione si blocca su questo ramo;
(c) se la regola scelta è $A \to w$, prova a consumare la stringa w dall'input. Se ci riesce va nello stato finale q_f e accetta se ha consumato tutto l'input. La computazione si blocca su questo ramo se l'automa non riesce a consumare w o se non ha consumato tutto l'input quando arriva in q_f.