Automi e Linguaggi Formali – 10/4/2024
Simulazione Prima prova intermedia – Soluzione

1. (12 punti) Diciamo che una stringa x è un *prefisso* della stringa y se esiste una stringa z tale che $xz = y$, e che è un *prefisso proprio* di y se vale anche $x \neq y$. Dimostra che se $L \subseteq \Sigma^*$ è un linguaggio regolare allora anche il linguaggio

$$NOPREFIX(L) = \{w \in L \mid \text{ nessun prefisso proprio di } w \text{ appartiene ad } L\}$$

è un linguaggio regolare.

Soluzione: Se L è un linguaggio regolare, allora sappiamo che esiste un DFA $A = (Q, \Sigma, \delta, q_0, F)$ che riconosce L. Costruiamo un DFA A' che accetta il linguaggio $NOPREFIX(L)$, aggiungendo uno stato "pozzo" agli stati di A, ossia uno stato non finale q_s tale che la funzione di transizione obbliga l'automa a rimanere per sempre in q_s una volta che lo si raggiunge. Lo stato iniziale e gli stati finali rimangono invariati. La funzione di transizione di A' si comporta come quella di A per gli stati non finali, mentre va verso lo stato pozzo per qualsiasi simbolo dell'alfabeto a partire dagli stati finali. In questo modo le computazioni accettanti di A' sono sempre sequenze di stati dove solo l'ultimo stato è finale, mentre tutti quelli intermedi sono non finali. Di conseguenza le parole che A' accetta sono accettate anche da A, mentre tutti i prefissi propri sono parole rifiutate da A, come richiesto dalla definizione del linguaggio $NOPREFIX(L)$.

Formalmente, $A' = (Q', \Sigma, \delta', q_0', F')$ è definito come segue.

- $Q' = Q \cup \{q_s\}$, con $q_s \notin Q$.
- L'alfabeto Σ rimane lo stesso.
- $\delta'(q, a) = \begin{cases} \delta(q, a) & \text{se } q \notin F \\ q_s & \text{altrimenti} \end{cases}$
- $q_0' = q_0$. Lo stato iniziale non cambia.
- $F' = F$. Gli stati finali rimangono invariati.

Per dimostrare che A' riconosce il linguaggio $NOPREFIX(L)$, dobbiamo considerare due casi.

- Se $w \in NOPREFIX(L)$, allora sappiamo che $w \in L$, mentre nessun prefisso proprio di w appartiene ad L. Di conseguenza esiste una computazione di A che accetta la parola:

  $$s_0 \xrightarrow{w_1} s_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} s_n$$

  con $s_0 = q_0$ e $s_n \in F$. Siccome tutti i prefissi propri di w sono rifiutati da A, allora gli stati $s_0, \dots, s_{n-1}$ sono tutti non finali. Per la definizione di A', la computazione che abbiamo considerato è anche una computazione accettante per A', e di conseguenza, $w \in L(A')$.
- Viceversa, se w è accettata dal nuovo automa A', allora esiste una computazione accettante che ha la forma

  $$s_0 \xrightarrow{w_1} s_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} s_n$$

  con $s_0 = q_0$, $s_n \in F$ e dove tutti gli stati intermedi $s_0, \dots, s_{n-1}$ sono non finali. Di conseguenza, la computazione è una computazione accettante anche per A, quindi $w \in L$. Siccome tutti gli stati intermedi della computazione sono non finali, allora A rifiuta tutti i prefissi propri di w, e quindi $w \in NOPREFIX(L)$.

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.