1 Parte Prima — 10 Points

- Si consideri l'espressione regolare $1\Sigma^* = 1(0 \cup 1)^*$. Convertire tale espressione regolare in una grammatica acontestuale. Dimostrare la correttezza.
- Enunciare e dimostrare il *pumping lemma* per linguaggi regolari. Fornire un esempio di utilizzo.

$1(1 \cup 0)^*$ genera tutte le stringhe che iniziano con 1.

$$G: \begin{cases} S \to 1A \\ A \to 1 \\ A \to 0 \\ A \to AA \mid \varepsilon \end{cases}$$

Chiaramente G puo' generare solo stringhe del tipo $1A$, ma A e' coinvolta nelle 3 regole canoniche per la generazione di una qualsiasi stringa.

$$\Rightarrow \quad G' = \begin{cases} A \to AA \mid \varepsilon \\ A \to 0 \\ A \to 1 \end{cases} \quad \Rightarrow L(G') = \Sigma^*$$

Teo (Pumping Lemma): Sia $L \in REG$ e sia $w \in L$, $\exists p \le |w|$ tale che:

per una scomposizione $w = xyz$ tale che

- $|y| \neq 0$ • $|xy| \le p$ • $xy^i z \in L$

Dim: Sia p il numero di stati del DFA che accetta L, sia $w = w_1 w_2 \cdots w_n$ e sia $R = \{r_1, r_2 \cdots r_{n+1}\}$ la successione di stati: $\forall i \;\; \delta(r_i, w_i) = r_{i+1}$

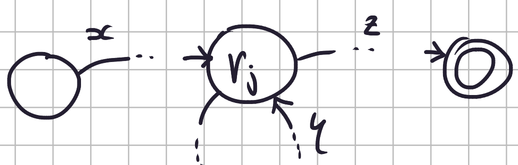
oss $n \ge p \Rightarrow n+1 > p \Rightarrow$ in R c'e' uno stato ripetuto nei primi $p+1$ stati di R

$r_1 \cdots, r_j \cdots r_\ell \cdots r_{n+1}$ e $r_j = r_\ell$, pongo la scomposizione $w = xyz$ con

$\ell \le p+1$

$$\begin{aligned} x &= w_1 \cdots w_{j-1} \\ y &= w_j \cdots w_{\ell-1} \\ z &= w_\ell \cdots w_n \end{aligned} \quad \Rightarrow |xy| = \ell - 1 \le p \qquad \text{inoltre } j \neq \ell \Rightarrow |y| \neq 0$$

y porta da r_j a r_ℓ quindi $xy^i z \in L$ perche'



Esempio di applicazione

$w \in L = \{0^n 1^n \mid n \in \mathbb{N}\}$

Sia $p = n \Rightarrow w = xyz$ ci sono diversi modi di scomporre

$w = 0^n 1^{n-q} 1^q \Rightarrow 0^n 1^{(n-q)i} 1^q \notin L$

$w = 0^q 0^{n-q} 1^n \Rightarrow 0^q 0^{(n-q)i} 1^n \notin L$

$w = 0^q (0^{n-q} 1^{n-q}) 1^q \Rightarrow 0^q (0^{n-q} 1^{n-q})^i 1^q = \underbrace{00 \cdots 0}_{q} \, 0101 \cdots 01 \, \underbrace{11 \cdots 1}_{q} \notin L$

2 Parte Seconda

10 Points

- Uno stato di una macchina di Turing è detto inutile se la macchina non entra mai in quello stato per ogni possibile input. Considerare il problema di determinare se una macchina di Turing ha uno stato inutile. Dimostrare che il linguaggio associato a questo problema non è decidibile.
- Dimostrare che esistono linguaggi che non sono decidibile e neppure Turing-riconoscibili.

$U_S = \{\langle M \rangle \mid M$ e' una TM e M ha uno stato inutile$\}$

Definisco R t.c. $\langle M, \omega \rangle \in A_{TM} \iff R(\langle M, \omega \rangle) \in U_S$

- Su input $\langle M, \omega \rangle$
- Definisce M' t.c. che
  - M' ha uno stato di accettazione q_a
  - Su input x, rifiuta sempre se $x \neq \omega$
  - M' va su $q_{acc} \iff M(\omega)$ rifiuta
- L'output e' $\langle M' \rangle$

R e' definita come segue:

Se $\langle M, \omega \rangle \in A_{TM} \Rightarrow M(\omega)$ accetta $\Rightarrow M'$ non raggiunge $q_a \Rightarrow M'$ ha uno stato inutile $\Rightarrow \langle M' \rangle \in U_S$

Se $\langle M, \omega \rangle \notin A_{TM} \Rightarrow M(\omega)$ rifiuta $\Rightarrow M'$ raggiunge $q_a \Rightarrow M'$ non ha uno stato inutile $\Rightarrow \langle M' \rangle \notin U_S$

$A_{TM} \leq_m U_S \Rightarrow U_S$ non e' decidibile.

Esistono linguaggi non riconoscibili:

Ogni TM e' codificabile in una stringa binaria $\langle M \rangle$ di lunghezza finita. $TM \subseteq \Sigma^*$ ma Σ^* e' in biezione con $\mathbb{N}$ secondo $f: \mathbb{N} \to \Sigma^*$ t.c. $f(k) =$ k-esimo elemento in Σ^* secondo l'ordine lessicografico. $\Rightarrow$ Le TM sono numerabili.

L'insieme di tutti i linguaggi $\mathcal{L} = \mathcal{P}(\Sigma^*)$ e' in biezione con l'insieme delle stringhe di lunghezza infinita $\mathcal{B} \Rightarrow \mathcal{L}$ non e' numerabile $\Rightarrow$ esistono linguaggi non associati a nessuna TM. Esempio: $\overline{A_{TM}}$

3 Parte Terza

10 Points

- Una macchina di Turing fortemente non-deterministica ha tre possibili stati finali: *accept*, *reject*, e *notsure*. Tale macchina decide un linguaggio L come segue: Se $x \in L$, tutti i possibili rami di computazione con input x portano ad uno stato *accept* oppure *notsure*, con almeno un ramo che termina in *accept*. D'altra parte, se $x \notin L$, tutti i possibili rami di computazione con input x portano ad uno stato *reject* oppure *notsure*, con almeno un ramo che termina in *reject*. Dimostrare che L è decidibile in tempo polinomiale da una macchina di Turing fortemente non-deterministica se e solo se $L \in NP \cap coNP$.
- Enunciare e dimostrare il teorema di gerarchia di tempo.

TM fort. non det.

Se L e' dec. in tempo $O(n^k)$ da F, allora

- Se $w \in L \Rightarrow F(w)$ ha almeno un ramo accettante
- se $w \notin L \Rightarrow F(w)$ ha almeno un ramo rifiutante

Quindi una NTM N puo' eseguire F e sicuramente almeno un ramo accetta o rifiuta $\Rightarrow$ N decide L.

Chiaramente si puó decidere se $w \notin L \Rightarrow w \in \bar{L} \Rightarrow L \in NP \cap coNP$

Se $L \in NP \cap coNP$ allora

$\exists N_1$ che decide se $w \in L$ in $O(n^k)$
$\exists N_2$ che decide se $w \in \bar{L}$ in $O(n^k)$

$\Rightarrow$ definisco F che esegue N_1 e N_2

F esegue N_1 e N_2
N_1 accetta $\Rightarrow F$ accetta
N_2 accetta $\Rightarrow F$ rifiuta

Se $w \in L$ N_1 accetta allora F ha almeno 1 ramo accettante

Se $w \notin L$ N_2 accetta allora F ha almeno 1 ramo in cui rifiuta

$\Rightarrow F$ decide L in $O(n^k)$.

Teo gerarchia di tempo

Se $t_1(n) \ll t_2(n)$ $\exists L \in Dtime(t_2(n))$ t.c. $L \notin Dtime(t_1(n))$

Dim: Definisco una TM D t.c.

- Su input $\langle M \rangle$
- esegue $M(\langle M \rangle)$ per $t_{1.5}(n)$ step
- Se termina prima, fa l'opposto

$t_1(n) \ll t_{1.5}(n) \ll t_2(n)$

Prop : $L(D) \notin Dtime(t_1(n))$

$\forall x \; Q(x) = D(x)$

Per assurdo Q decide $L(D)$ in $O(t_1(n))$

Eseguo $D(\langle Q \rangle)$

$\Rightarrow D$ esegue $Q(\langle Q \rangle)$

$Q(\langle Q \rangle)$ termina in $t_1(n) < t_{1.5}(n)$ passi

D quindi fa l'opposto

$\Downarrow$

$D(\langle Q \rangle) \neq Q(\langle Q \rangle) \Rightarrow Q$ non decide $L(D)$. ∎