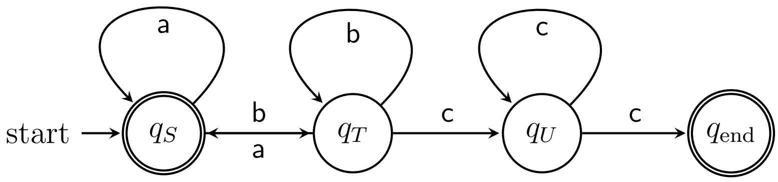
Computational Complexity
Homework 2024-25

Simone Bianco, 1986936

Sapienza Università di Roma, Italy

January 14, 2025

Esercizio 1.1



Esercizio 1.2

Enunciato: Se L è un linguaggio regolare allora esiste un $p \in \mathbb{N}$ tale che $\forall w \in L$ con $|w| \geq p$ esistono tre stringhe $x, y, z \in \Sigma^*$ per cui $w = xyz$ e:

1. $\forall i \in \mathbb{N}$ vale che $xy^iz \in L$
2. $|y| > 0$
3. $|xy| \leq p$

Dimostrazione:

Dato $L \in \mathsf{REG}$, sia $D := (Q, \Sigma, \delta, q_0, F)$ il DFA tale che $L = L(D)$

Sia $p := |Q|$. Data la stringa $w := w_1 \dots w_n \in L$ dove $w_1, \dots, w_n \in \Sigma$ e dove $n \geq p$, consideriamo la sequenza di stati $r_1, \dots, r_{n+1}$ tramite cui w viene accettata da D:

$$\forall k \in [1, n] \;\; \delta(r_k, w_k) = r_{k+1}$$

Notiamo quindi che $|r_1, \dots, r_{n+1}| = n + 1$, ossia che il numero di stati attraversati sia $n + 1$. Inoltre, in quanto $n \geq p$, ne segue automaticamente che $n + 1 \geq p + 1$. Tuttavia, poiché $p := |Q|$ e $n + 1 \geq p + 1$, ne segue necessariamente che $\exists i, j \mid 1 \leq i < j \leq p + 1 \land r_i = r_j$, ossia che tra i primi $p + 1$ stati della sequenza vi sia almeno uno stato ripetuto

A questo punto, consideriamo le seguenti sottostringhe di w:

- $x = w_1 \dots w_{i-1}$, tramite cui si ha che $\delta^*(r_1, x) = r_i$
- $y = w_i \dots w_{j-1}$, tramite cui si ha che $\delta^*(r_i, y) = r_j = r_i$
- $z = w_j \dots w_n$, tramite cui si ha che $\delta^*(r_j, z) = r_{n+1}$

Poiché $\delta^*(r_i, y) = r_i$, ossia y porta sempre r_i in se stesso, ne segue automaticamente che

$$\forall k \in \mathbb{N} \;\; \delta^*(r_i, y^k) = r_i \implies \delta(r_1, xy^kz) = r_{n+1} \in F \implies xy^kz \in L(D) = L$$

Inoltre, ne segue direttamente che $|y| > 0$ in quanto $i < j$ e che $|xy| \leq p$ in quanto $j \leq p + 1$

Esempio di applicazione:

Consideriamo il linguaggio $L = \{0^n 1^n \mid n \in \mathbb{N}\}$. Supponiamo per assurdo che L sia regolare. In tal caso, ne segue che per esso debba valere il pumping lemma, dove p è la lunghezza del pumping. Consideriamo quindi la stringa $w := 0^p 1^p \in L$. Poiché $|w| \geq p$, possiamo suddividerla in tre sottostringhe $x, y, z \in \Sigma^*$ tali che $w = xyz$,

Poiché la terza condizione del pumping lemma impone che $|xy| \leq p$ e poiché $w := 0^p 1^p$, ne segue che $xy = 0^m$ e $z = 0^{p-m} 1^p$, dove $m \in [1, p]$ Inoltre, per la seconda condizione del lemma, si ha che $|y| > 0$, dunque necessariamente si ha che $x = 0^{m-k}$ e $y = 0^k$, dove $k \in [1, m]$

A questo punto, consideriamo la stringa $xy^0 z$. Notiamo immediatamente che

$$xy^0 z = 0^{m-k} (0^k)^0 0^{p-m} 1^p = 0^{m-k} 0^{p-m} 1^p = 0^{p-k} 1^p$$

implicando dunque che $xy^0 z \notin L$, contraddicendo la prima condizione del lemma per cui si ha che $\forall i \in \mathbb{N} \;\; xy^i z \in L$. Dunque, ne segue necessariamente che L non possa essere regolare.

Esercizio 2.1

Dimostrazione:

Sia $f : \Sigma^* \to \Sigma^*$ la funzione calcolata dalla TM F definita come segue.

$F =$ "Data la stringa $\langle M, w\rangle$ in input:

1. Costruisci una TM M' definita come:

   $M' =$ "Data la stringa x in input:

   i. Esegui il programma di M su input w.

   ii. Se M accetta, M' *accetta*. Se M rifiuta, M' va in loop.

2. Restituisci in output la stringa $\langle M'\rangle$"

Supponiamo che $\langle M, w\rangle \in A_{\mathsf{TM}}$. In tal caso, per ogni input x la macchina M' accetterà sempre, implicando che essa sia un decisore e dunque che $\langle M'\rangle \in$ DECIDER$_{TM}$.

Viceversa, supponiamo che $\langle M, w\rangle \not\in A_{\mathsf{TM}}$. In tal caso, abbiamo due opzioni: $M(w)$ rifiuterà oppure andrà in loop. Se $M(w)$ rifiuta allora $M'(x)$ andrà in loop. Se invece $M(w)$ va in loop, allora anche $M'(x)$ andrà in loop poiché esegue $M(w)$ al suo interno. In entrambi i casi la macchina risulta non essere un decisore, dunque $\langle M'\rangle \notin$ DECIDER$_{TM}$. Ciò conclude che f sia una riduzione da A_{TM} a DECIDER$_{TM}$. Inoltre, poiché A_{TM} è indecidibile, concludiamo che anche DECIDER$_{TM}$ deve necessariamente essere indecidibile.

Esercizio 2.2

Dimostrazione:

Sia $\prec$ una relazione che ordina le stringhe di Σ^* in base alla loro lunghezza e (a parità di lunghezza) in base al loro ordine lessico-grafico.

Sia quindi $f : \mathbb{N} \to \Sigma^*$ la funzione definita come:

$$f(i) = i\text{-esima stringa di } \Sigma^* \text{ secondo } \prec$$

Poiché $\prec$ è un ordine totale, tale funzione risulta essere biettiva, implicando che $|\mathbb{N}| = |\Sigma^*|$. Consideriamo quindi il linguaggio $\mathcal{M} \subseteq \Sigma^*$ definito come:

$$\mathcal{M} = \{\langle M\rangle \mid M \text{ è una } \mathsf{TM}\}$$

Poiché $\mathcal{M} \subseteq \Sigma^*$ e Σ^* è numerabile, ne segue automaticamente che anche $|\mathbb{N}| = |\mathcal{M}|$. Consideriamo quindi l'insieme $\mathcal{L} = \mathcal{P}(\Sigma^*)$, corrispondente all'insieme di tutti i linguaggi definiti su Σ. Dato $L \in \mathcal{L}$, definiamo la sequenza binaria $\chi_L = b_1 b_2 \dots$ come:

$$b_i = \begin{matrix} 1 & \text{se } s_i \in L \\ 0 & \text{se } s_i \notin L \end{matrix}$$

dove $s_1, s_2, \dots$ sono tutte le stringhe di Σ^* ordinate secondo $\prec$. Consideriamo quindi la seguente funzione:

$$g : \mathcal{L} \to \mathcal{B} : L \mapsto \chi_L definita$$

Anche tale funzione è biettiva, implicando che $|\mathcal{L}| = |\mathcal{B}|$. Di conseguenza, poiché $\mathcal{B}$ non è numerabile, ne segue che anche $\mathcal{L}$ non sia numerabile. A questo punto, poiché $\mathcal{M}$ è numerabile e $\mathcal{L}$ no, concludiamo che la seguente funzione:

$$h : \mathcal{M} \to \mathcal{L} : M \mapsto L(M)$$

non possa essere biettiva, implicando che $\exists L \in \mathcal{L} \mid \nexists M \in \mathcal{M}$ t.c $L = L(M)$

Esercizio 3.1

Dimostrazione:

Procederemo in modo simile al padding argument che dimostra come se $\mathsf{P} = \mathsf{NP}$ allora $\mathsf{EXP} = \mathsf{NEXP}$. Dato un linguaggio $L \in \text{NTIME}(n^5)$, sia V il verificatore per L con runtime $O(n^5)$. Consideriamo il linguaggio $L_{pad} = \{x\#1^{|x|^{\frac{5}{2}}} \mid x \in L\}$ e definiamo il seguente verificatore V_{pad}:

$V_{pad} =$ "Dati x e w in input:

1. Controlla se $x = y\#1^{|y|^{\frac{5}{2}}}$ per qualche stringa y.
2. Se falso, rifiuta.
3. Se vero, esegui $V(y, w)$ e ritorna il suo risultato"

Poiché l'input di V_{pad} ha dimensione $m = \left|y\#1^{|y|^{\frac{5}{2}}}\right|$, la prima istruzione di V_{pad} richiede tempo $O(m)$, mentre l'ultima istruzione può essere eseguita in tempo $O(m^2)$. Ciò conclude che $L_{pad} \in \text{NTIME}(n^2)$.

Di conseguenza, sotto l'assunzione $\text{DTIME}(n^2) = \text{NTIME}(n^2)$, ne segue che esiste un decisore M_{pad} per L_{pad} con runtime $O(n^2)$. Costruiamo quindi una nuova TM definita come:

$M =$ "Dato x in input:

1. Costruisci la stringa $y = x\#1^{|x|^{\frac{5}{2}}}$.
2. Esegui $M_{pad}(y)$ e ritorna il suo risultato.

Chiaramente M risulta essere un decisore per L. Inoltre, per via della costruzione della stringa y, il suo runtime risulta essere $O(n^5)$, concludendo che $L \in \text{DTIME}(n^5)$.

Esercizio 3.2

Dimostrazione:

Per definizione delle due classi sappiamo già che PSPACE $\subseteq$ NPSPACE. Per ottenere l'altra inclusione, dimostriamo il teorema di Savitch, il quale afferma che NSPACE$(f(n)) \subseteq$ DSPACE$(f^2(n))$.

Sia N una NTM tale che $L(N) \in$ NSPACE$(f(n))$. Assumiamo N abbia un solo stato accettante q_{accept}. Sia inoltre q_{start} lo stato iniziale di N. Dato un input w, consideriamo quindi il grafo delle configurazioni $G_{N,w}$ per $N(w)$ definito come:

- Ad ogni nodo di V corrisponde una configurazione possibile di N durante la computazione di w
- Per ogni nodo $v_i, v_j \in V$, esiste un arco se e solo se la computazione può passare dalla configurazione c_i alla configurazione c_j tramite δ

Definiamo quindi la seguente procedura $\texttt{PathG}_{\texttt{N,w}}?(x, y, k)$:

$\texttt{PathG}_{\texttt{N,w}}?(x, y, k)$:

1. Se $k = 0$, verifica se $(x, y) \in E(G_{N,w})$ o se $x = y$. Se vero, *accetta.* Altrimenti, *rifiuta*
2. Se $k > 0$, ripeti la seguente istruzione per ogni nodo v in $V(G_{N,w})$:
   3. Se $\texttt{PathG}_{\texttt{N,w}}?(x, v, k-1)$ accetta e $\texttt{PathG}_{\texttt{N,w}}?(v, y, k-1)$ accetta, allora la procedura *accetta.* Altrimenti, essa *rifiuta*

Per costruzione stessa di $\texttt{Path}?$, si ha che:

$$\texttt{Path}?(x, y, k) \text{ accetta} \iff \text{Esiste cammino } x \to y \text{ di massimo } 2^k \text{ nodi}$$

Il costo in termini di spazio per questa procedura risulta essere $O(\log^2 k)$, dove k è il valore dato in input alla prima chiamata della procedura. Definiamo quindi la seguente TM M:

$M =$ "Data la stringa w in input:

1. Esegui la procedura $PathG_{N,w}?(c_{\text{start}}, c_{\text{accept}}, \lceil \log m \rceil)$ costruendo il grafo $G_{N,w}$ durante la sua esecuzione (conservando sempre solo il nodo attuale e successivo)

2. Se la procedura accetta, *accetta*. Altrimenti, *rifiuta*"

Per costruzione stessa di M, si ha che:

$$w \in L(M) \iff PathG_{N,w}?(c_{\text{start}}, c_{\text{accept}}, \lceil \log 2^{f(n)} \rceil) \text{ accetta } \iff$$

$$\text{Esiste cammino } c_{\text{start}} \to c_{\text{accept}} \text{ di } 2^{f(n)} \text{ nodi in } G_{N,w} \iff w \in L$$

implicando che $L(M) = L$. Notiamo quindi che la costruzione parziale del grafo richieda solo qualche "variabile" di appoggio, mantenendo il costo della procedura inalterato in termini di spazio, ossia pari a $O(\log^2 2^{f(n)}) = O(f^2(n))$, concludendo che $L \in \text{DSPACE}(f^2(n))$