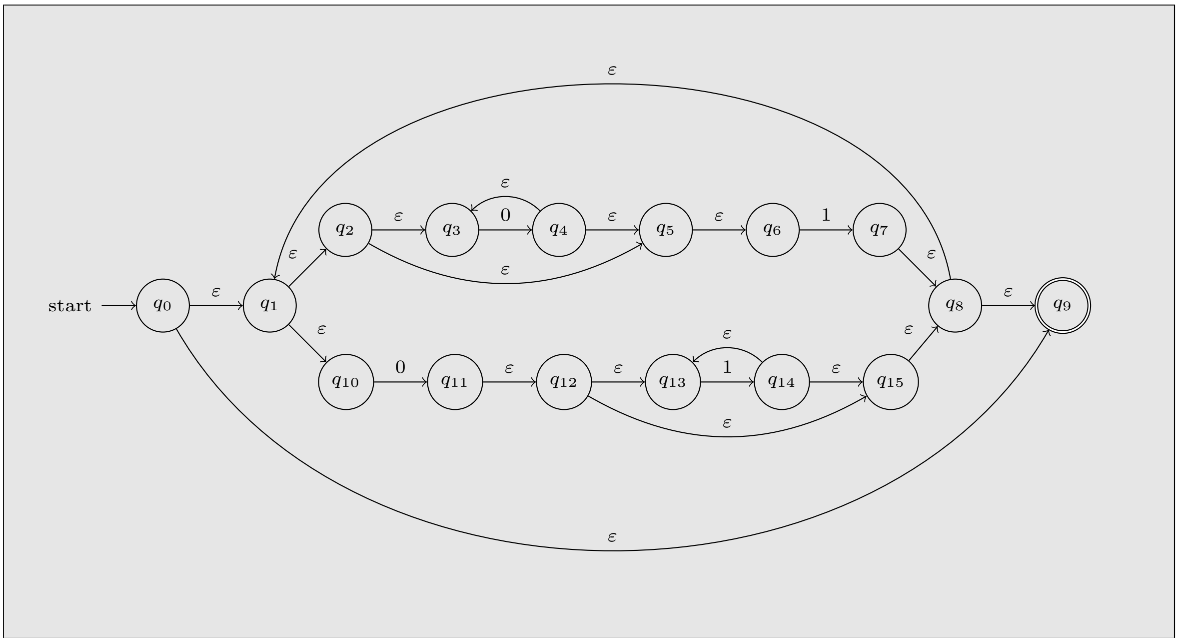
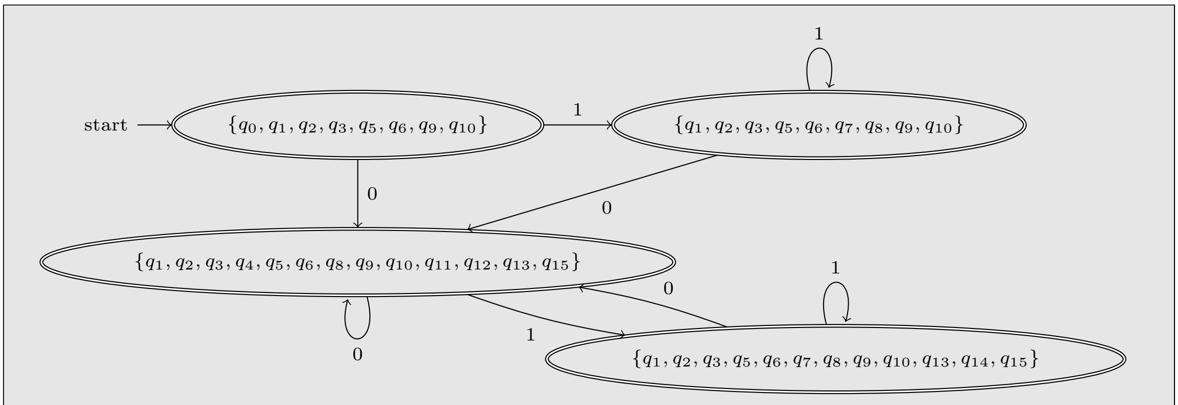
Automi e Linguaggi Formali
Esame del 12 Giugno 2018 – Parte I

Tempo a disposizione: 1 h 30 min

1. (a) Convertire l'espressione regolare $(\mathbf{0}^*\mathbf{1} + \mathbf{0}\mathbf{1}^*)^*$ in un ε-NFA usando le regole viste a lezione.



(b) Trasformare l'ε-NFA ottenuto al punto precedente in un DFA.



2. (a) Dimostrare che il linguaggio $L_1 = \{0^{2n}1^n : n \geq 0\}$ *non* è regolare.

Il linguaggio non è regolare. Supponiamo per assurdo che lo sia:

- sia h la lunghezza data dal Pumping Lemma;
- consideriamo la parola $w = 0^{2h}1^h$, che appartiene ad L_1 ed è di lunghezza maggiore di h;
- sia $w = xyz$ una suddivisione di w tale che $y \neq \varepsilon$ e $|xy| \leq h$;
- poiché $|xy| \leq h$, allora xy è completamente contenuta nel prefisso 0^{2h} di w, e quindi sia x che y sono composte solo da 0. Inoltre, siccome $y \neq \varepsilon$, possiamo dire che $y = 0^p$ per qualche valore $p > 0$. Allora la parola xy^2z è nella forma $0^{2h+p}1^h$, e quindi non appartiene al linguaggio perché il numero di 0 non è uguale al doppio del numero di 1 (dovrebbero essere $h + p/2$ mentre sono solo h).

Abbiamo trovato un assurdo quindi L_1 non può essere regolare.

(b) Considerate il linguaggio $L_2 = \{0^m 1^n : m \neq 2n\}$. Questo linguaggio è regolare? Giustificare formalmente la risposta (*la giustificazione non dovrebbe richiedere più di due righe di testo*).

Si può osservare che L_2 è il complementare del linguaggio L_1 (contiene tutte e sole le parole che non appartengono a L_1). Al punto precedente abbiamo dimostrato che L_1 non è regolare, quindi nemmeno L_2 può essere regolare perché i linguaggi regolari sono chiusi per complementazione.

3. Sia L un linguaggio regolare su un alfabeto Σ. Supponete che il simbolo # appartenga all'alfabeto Σ e dimostrate che il seguente linguaggio è regolare:

$$dehash(L) = \{dehash(w) : w \in L\}$$

dove $dehash(w)$ è la stringa che si ottiene eliminando tutti i simboli # da w.

Per dimostrare che $dehash(L)$ è regolare vediamo come è possibile costruire un automa a stati finiti che riconosce $dehash(L)$ a partire dall'automa a stati finiti che riconosce L.

Sia quindi $A = (Q, \Sigma, q_0, \delta, F)$ un automa a stati finiti che riconosce il linguaggio L. Costruiamo un ε-NFA $B = (Q, \Sigma, q_0, \delta_B, F)$ che ha lo stesso insieme di stati, lo stesso stato iniziale e gli stessi stati finali di A. La funzione di transizione del nuovo automa rimpiazza ogni transizione etichettata con # di A con una ε-transizione tra la stessa coppia di stati, lasciando inalterate le transizioni etichettate con gli altri simboli di Σ.

4. Si consideri la seguente grammatica libera da contesto G:

$$S \to iS \mid iSeS \mid \epsilon$$

(a) dare una descrizione del linguaggio generato da G nella forma $L = \{w \mid w \in \{i, e\}^* \text{ tali che} \ldots\}$ e dimostrare che vale $L \supseteq L(G)$; (opzionale: spiegare anche che vale $L \subseteq L(G)$)

$L = \{w \in \{i, e\}^* \mid \text{per ogni prefisso di } w \text{ il numero di } i \text{ è maggiore o uguale al numero di } e\}$
Dimostriamo che $L \supseteq L(G)$ per induzione *sulla lunghezza della derivazione.*
Base: lunghezza 1. In questo caso l'unica produzione è $S \Rightarrow \epsilon$. Poiché $\epsilon \in L$ la tesi è dimostrata.
Passo induttivo: lunghezza $n+1$. Assumiamo per ipotesi induttiva che la tesi sia vera per tutte le derivazioni di lunghezza minore o uguale a n.
La derivazione di lunghezza $n+1$ può essere fatta in due modi:

- $S \Rightarrow iS \Rightarrow^n iw' = w$. Per ipotesi induttiva $w' \in L$. Poiché aggiungo una i in più, la proprietà di bilanciamento del numero di i e di e rimane vera anche per iw' e quindi ho dimostrato che $w \in L$.
- $S \Rightarrow iSeS \Rightarrow^* iw'ew'' = w$. Per ipotesi induttiva w' e $w'' \in L$. Quindi la proprietà di bilanciamento del numero di i e di e rimane vera anche per iw' e per $iw'e$, e di conseguenza anche per $iw'ew''$. Quindi ho dimostrato che $w \in L$.

(b) dimostrare che la grammatica è ambigua;

G è ambigua perché posso derivare la parola iie in due modi diversi:

- $S \Rightarrow iSeS \Rightarrow iiSeS \Rightarrow^* iie$
- $S \Rightarrow iS \Rightarrow iiSeS \Rightarrow^* iie$

(c) osservando che questa grammatica modella l'annidamento di $if - then$ e $if - then - else$ nei programmi, fornire una grammatica non ambigua che generi lo stesso linguaggio della grammatica di partenza. Spiegare l'idea alla base della nuova grammatica.

$S \to iS \mid iS'eS \mid \epsilon$
$S' \to iS'eS' \mid \epsilon$