Nota 1. Considerate queste soluzioni come descrizioni dei procedimenti; non è richiesto, nello svolgimento del compito, di descrivere con tale completezza tutti i passi, come è fatto qui a scopo didattico.

Soluzioni del compito scritto di informatica teorica del 21 gennaio 2003

Esercizio 1. Sia L un linguaggio riconosciuto da una macchina di Turing T e tale che anche il suo complementare L^c è riconosciuto da una macchina di Turing R. Cosa può dirsi sul membership problem per L?

Soluzione:

Il membership problem per L è decidibile. Infatti, data una stringa w, basta darla come input alla macchina T e alla macchina R e fare iniziare contemporaneamente il computo alle due macchine.

Poiché o w è riconosciuta da T o w è riconosciuta da R, certamente, dopo un numero finito di passi, una delle due macchine si arresta e fornisce la risposta di accettazione. Se la macchina che si arresta è T, allora $w \in L$, altrimenti $w \notin L$.

Esercizio 2. Sia L il linguaggio costituito da tutte le parole sull'alfabeto $\{a, b, c\}$ tali che ogni a è seguita da b e nessuna c è seguita da b. Costruire un DFA che riconosce L.

Soluzione:

Tale linguaggio è l'intersezione dei linguaggi L_1 e L_2 definiti nel modo seguente:

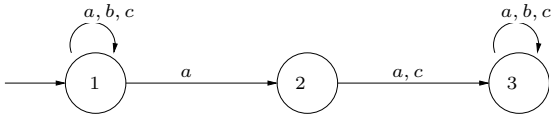
- L_1 è l'insieme delle stringhe su $\{a, b, c\}$ tali che aa e bc siano proibiti e inoltre le stringhe non possono terminare per a;
- L_2 è l'insieme delle stinghe in cui non compare il fattore cb.

In simboli

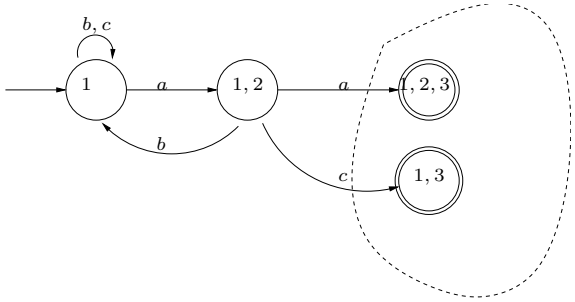
$$L_1 = L' \cap \Sigma^*(b + c)$$

dove $\Sigma = \{a, b, c\}$ e L' è l'insieme delle stringhe in cui non compare né aa né ac.

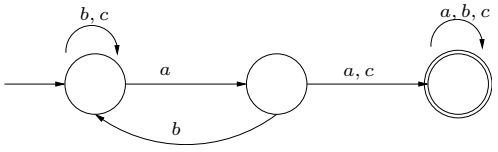
NFA per L'^c:



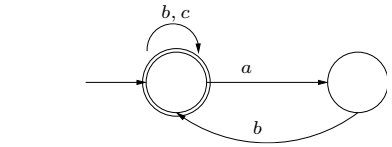
Determinizziamo:



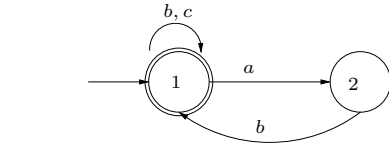
Nel processo di determinizzazione, tutte le transizioni che partono da "stati" (usiamo le virgolette perché ci riferiamo ai sottoinsiemi ottenuti con la subset construction) che contengono stati di accettazione dell'automa di partenza, conducono a "stati" che hanno la stessa proprietà; tutti questi "stati" sono pertanto equivalenti.



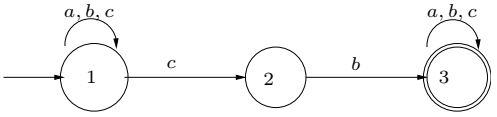
DFA per L'^c



DFA per L'

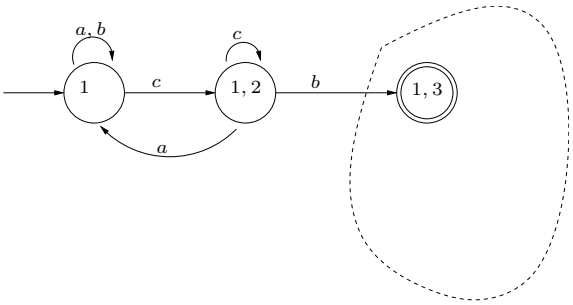


DFA per L_1



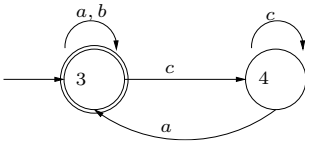
NFA per L_2^c

e quindi il DFA per L_2^c



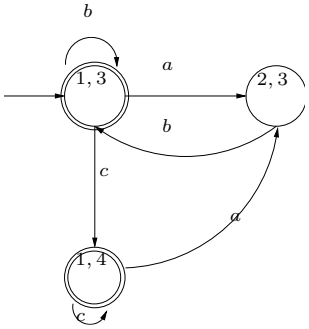
e anche qui valgono le considerazioni fatte per l'equivalenza di alcuni "stati" dopo il processo di determinizzazione.

Continuando si ha



DFA per L_2

Calcolo dell'intersezione:



Esercizio 3. Scrivere un'espressione regolare per il linguaggio nel problema precedente.

SOLUZIONE:

L'espressione regolare può essere scritta come

$$(b + cc^*ab + ab)^* + (b + cc^*ab + ab)^*c$$

o come

$$(bb + cc^*ab + ab)^*(\varepsilon + c)$$

o come

$$(b + (cc^* + \varepsilon)ab^*)^*(\varepsilon + c)$$

Esercizio 4. Sia (u, v) una coppia di stringhe sull'alfabeto $\{0, 1\}$, di uguale lunghezza. Se la stringa u viene scritta sopra la stringa v, alla coppia (u, v) si puó fare corrispondere una parola sull'alfabeto $\{A, B, C, D\}$ dove

$$A = \begin{matrix} 0 \\ 0 \end{matrix}, \quad B = \begin{matrix} 0 \\ 1 \end{matrix}, \quad C = \begin{matrix} 1 \\ 0 \end{matrix}, \quad D = \begin{matrix} 1 \\ 1 \end{matrix}$$

Ad esempio alla coppia (u, v) con $u = 011$ e $v = 110$ corrisponde la parola BDC.

Sia L il linguaggio sull'alfabeto $\{A, B, C, D\}$ costituito dalle parole corrispondenti a coppie (u, v) tali che v rappresenti in binario un intero che é il doppio di quello rappresentato da u. Ad esempio $BDC \in L$, in quanto u e v rappresentano in binario rispettivamente 3 e 6.

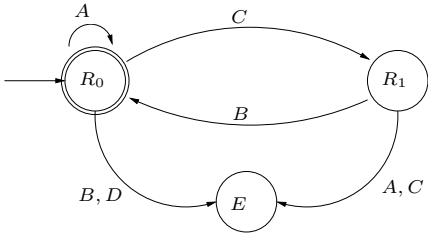
Costruire un DFA che riconosce L.

Soluzione:

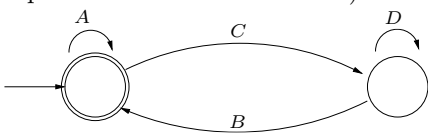
Si costruisce inizialmente l'automa che riconosce le stringhe del linguaggio leggendole da destra verso sinistra, ovvero, più formalmente, che riconosce il linguaggio L^R.

In una lettura da destra verso sinistra, ad ogni coppia di bit incontrati, per verificarne la correttezza occorre distinguere due casi: se si sta leggendo la coppia di bit trovandosi in uno stato di "riporto 0" o se si sta leggendo in uno stato di "riporto 1". Pertanto l'automa avrà due stati, denotati R_0 e R_1. Ovviamente R_0 è sia stato iniziale che di accettazione: infatti si parte da una situazione di "riporto 0" e, alla fine della verifica, ci si deve necessariamente trovare in uno stato di "riporto 0".

Occorrerà anche disporre di un terzo stato, che denoteremo con E ("errore"). Quando nel corso della verifica si legge una coppia di bit che non è compatibile con ciò che richiede il problema, si avrò una transizione nello stato E. Ovviamente E si comporta come uno "stato pozzo", in quanto, una volta che si rileva un errore, la stringa non sarà più accettata. Le transizioni dell'automa vengono determinate, verificando se la coppia di bit incontrati, tenendo anche conto del riporto, è compatibile con ciò che richiede il problema. Si costruisce quindi l'automa seguente



L'automa che riconosce L si ottiene dal precedente invertendo lo stato iniziale con lo stato d'accettazione e invertendo l'orientamento degli archi (e eventualmente determinizzando, ma in questo caso non è necessario). Il risultato è



Esercizio 5. Data l'espressione regolare

$$(a + b(ab)^*b)^*,$$

costruire, usando l'algoritmo di Berry e Sethi, un automa a stati finiti che riconosce il linguaggio corrispondente.

Soluzione:

Si crea un alfabeto ausiliario costituito da 5 simboli

$$\Sigma' = \{a_1, a_2, a_3, a_4, a_5\}$$

e si determina la tabella di conversione tra i nuovi simboli e quelli dell'alfabeto originario:

$$a_1 \to a$$
$$a_2 \to b$$
$$a_3 \to a$$
$$a_4 \to b$$
$$a_5 \to b$$

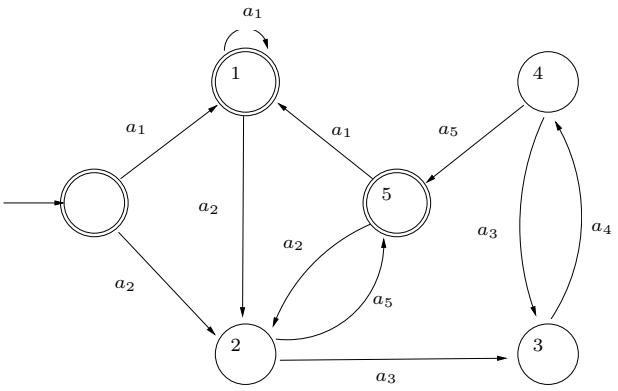
Si ottiene una nuova espressione regolare che, questa volta, è lineare

$$(a_1 + a_2(a_3a_4)^*a_5)^*$$

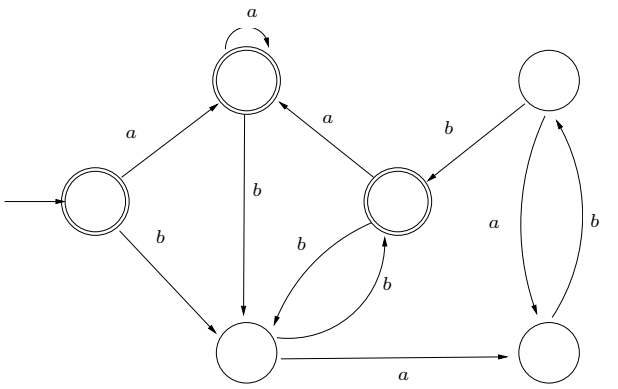
Sappiamo che un'espressione lineare corrisponde a un linguaggio locale. Possiamo quindi ricavare direttamente la quadrupla che definisce il linguaggio

$$\Lambda = \{\varepsilon\}$$
$$P = \{a_1, a_2\}$$
$$S = \{a_1, a_5\}$$
$$F = \{a_1a_1, a_1a_2, a_2a_3, a_3a_4, a_4a_3, a_4a_5, a_2a_5, a_5a_1, a_5a_2\}$$

L'automa locale corrispondente è



Applicando la tabella di conversione si ottiene il seguente NFA:



Esercizio 6. Una parola v sull'alfabeto $A = \{a, b, c\}$ si dice *senza quadrati* se, comunque presi $x, y, z \in A^*$, $v = xy^2z$ implica che $y = \epsilon$.

Denotiamo con SQ il linguaggio delle parole senza quadrati sull'alfabeto A. E' noto che il linguaggio SQ é infinito. Esiste un automa a stati finiti che riconosce SQ? Motivare la risposta.

SOLUZIONE:

SQ non è riconoscibile in quanto non soddisfa il lemma di iterazione. Infatti, poiché SQ è un linguaggio infinito, comunque preso un intero N (che interviene nel lemma di iterazione) esiste una stringa $w \in SQ$ con $|w| > N$. Applicando il lemma di iterazione si ha

$$w = xyz$$

e, per ogni $n \geq 0$, $xy^n z \in SQ$, ma ciò contraddice la definizione di SQ.

Esercizio 7. Dati due linguaggi L_1 e L_2 sullo stesso alfabeto, la *differenza simmetrica* tra L_1 e L_2 é definita nel modo seguente:

$$\delta(L_1, L_2) = L_1 \setminus L_2 \cup L_2 \setminus L_1.$$

Si dice che L_1 e L_2 hanno *differenza finita* se $\delta(L_1, L_2)$ é un linguaggio finito. Dati due linguaggi regolari, é decidibile se hanno differenza finita? Motivare la risposta.

SOLUZIONE:

$\delta(L_1, L_2)$ può anche scriversi:

$$\delta(L_1, L_2) = L_1 \cap L_2^c \cup L_2 \cap L_1^c$$

Se $\mathcal{A}_1$ e $\mathcal{A}_2$ sono gli automi che riconoscono rispettivamente L_1 e L_2, è possibile, utilizzando le costruzioni relative alle operazioni booleane, costruire un automa $\mathcal{A}$ che riconosce $\delta(L_1, L_2)$.

La risposta al problema si ottiene quindi dalla decidibilità del "Finiteness Problem" per gli automi a stati finiti.

Esercizio 8. Costruire una grammatica context-free che genera il linguaggio definito nel problema 2.

SOLUZIONE:

$$\Omega \to b\Omega$$
$$\Omega \to b$$
$$\Omega \to aA$$
$$\Omega \to cB$$
$$\Omega \to c$$
$$A \to b\Omega$$
$$A \to b$$
$$B \to aA$$
$$B \to cB$$
$$B \to c$$

Esercizio 9. Costruire una grammatica in forma normale di Chomsky per il linguaggio $\{a^n b^n | n \geq 1\}$.

SOLUZIONE:

Una grammatica per il linguaggio in questione è

$$\Omega \to a\Omega b$$
$$\Omega \to ab$$

Questa grammatica non è in CNF. Applicando la prima trasformazione, si ottiene la grammatica

$$\Omega \to A\Omega B$$
$$\Omega \to AB$$
$$A \to a$$
$$B \to b$$

Applicando la seconda trasformazione, si ottiene una grammatica CNF

$$\Omega \to AC$$
$$C \to \Omega B$$
$$\Omega \to AB$$
$$A \to A$$
$$B \to b$$

Esercizio 10. Il linguaggio

$$L = \{a^n b^m c^n d^m | n, m \geq 1\}$$

puó essere generato da una grammatica context-free?

SOLUZIONE:

Il linguaggio non soddisfa il lemma di iterazione per i linguaggi context-free. Infatti, qualunque sia l'intero N (che interviene nell'enunciato del lemma), consideriamo una stringa del linguaggio

$$a^n b^m c^n d^m$$

con $n, m > N$.

Facendo un'analisi per casi, si vede facilmente che l'unica possibilità per ottenere una coppia iterante sarebbe quella di considerare il primo elemento della coppia come fattore di a^n e il secondo elemento come fattore di c^n (ovvero il primo elemento della coppia come fattore di b^m e il secondo come fattore di d^m); in entrambi i casi, però, i due elementi della coppia avrebbero una distanza maggiore di N, contraddicendo il lemma d'iterazione.