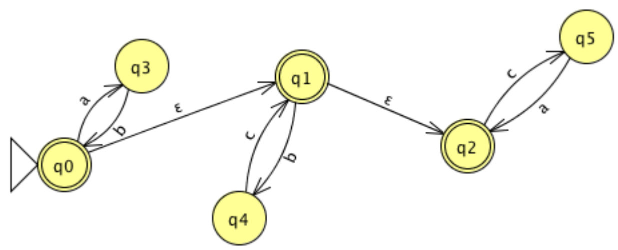
Teoria degli Automi e Calcolabilità a.a. 2021/22
Prova scritta 14 giugno 2022

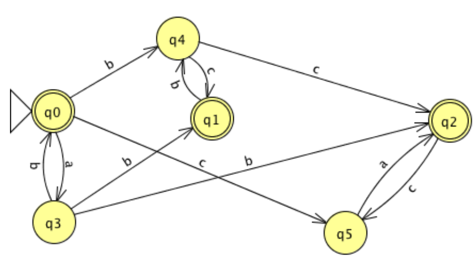
Esercizio 1 Si consideri il seguente automa a stati finiti non deterministico con transizioni ϵ.



1. Si dia un'espressione regolare che denoti il linguaggio riconosciuto.
2. Si definisca la ϵ-closure di ogni stato e si dia un NFA senza transizioni ϵ equivalente.

Soluzione

1. Un'espressione regolare che denota il linguaggio riconosciuto è $(ab)^\star(bc)^\star(ca)^\star$.
2. Si ha $\epsilon\text{-}closure(q_0) = \{q_0, q_1, q_2\}$, $\epsilon\text{-}closure(q_1) = \{q_1, q_2\}$, per tutti gli altri stati la ϵ-closure contiene solo lo stato stesso. Un NFA senza transizioni ϵ equivalente è il seguente:



	a	b	c
$\rightarrow *q_0$	q_3	q_4	q_5
$*q_1$		q_4	q_5
$*q_2$			q_5
q_3		q_0, q_1, q_2	
q_4			q_1, q_2
q_5	q_2		

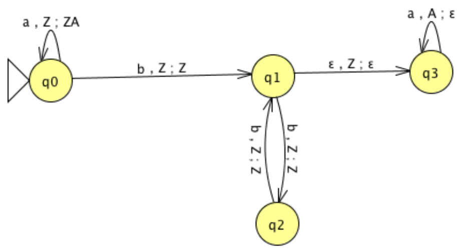
Esercizio 2 Si consideri il seguente linguaggio:

$$\{a^n b^m a^n \mid n, m \geq 0 \wedge m \text{ dispari}\}$$

1. Si dia un automa a pila che lo riconosca (per pila vuota).
2. È possibile che questo automa a pila sia deterministico?
3. È possibile riconoscere il linguaggio con un NFA?

Soluzione

1. Un automa a pila che riconosce il linguaggio per pila vuota è il seguente:



2. Non è possibile riconoscere il linguaggio con un automa a pila deterministico in quanto esistono due stringhe appartenenti al linguaggio delle quali una è prefisso dell'altra, per esempio b e bbb.

3. Non è possibile riconoscere il linguaggio con un NFA in quanto non regolare, come si può provare con il pumping lemma. Infatti, dato $n \geq 0$, consideriamo per esempio la stringa $a^n b a^n$ che appartiene al linguaggio. Decomponendo la stringa in uvw, con $|uv| \leq n$ e $|v| > 0$, si ha necessariamente che u e v sono formate di sole a, ossia:

   $u = a^x$
   $v = a^y$
   $w = a^z b a^n$, con $x + y + z = n$, $y > 0$.

   Quindi, per esempio, la stringa uv^0w ha un numero $(x + z)$ di a strettamente minore di n e quindi non appartiene al linguaggio.

Esercizio 3 Si consideri la funzione ricorsiva primitiva che restituisce la somma di due numeri naturali vista a lezione.

$sum(x, Z) = x$
$sum(x, S(y)) = S(sum(x, y))$

Abbreviamo con $\bar{x}$ l'espressione $S^x(Z)$.
Si diano due diverse computazioni per l'espressione $sum(sum(Z, \bar{1}), \bar{1})$.

Soluzione Due diverse computazioni sono le seguenti.

$sum(sum(Z, \bar{1}), \bar{1}) \to sum(S(sum(Z, Z)), \bar{1}) \to sum(\bar{1}, \bar{1}) \to S(sum(\bar{1}, Z)) \to \bar{2}$
$sum(sum(Z, \bar{1}), \bar{1}) \to S(sum(sum(Z, \bar{1}), Z)) \to S(sum(Z, \bar{1})) \to S(S(sum(Z, Z))) \to \bar{2}$

Esercizio 4 Si provino le seguenti affermazioni ($\overline{A}$ denota il complementare di A).

1. Se $\overline{A}$ è finito, allora A è ricorsivo.

2. Se $A \leq B$, allora $\overline{A} \leq \overline{B}$.

Soluzione

1. Un insieme finito è ricorsivo, e il complentare di un insieme ricorsivo è ricorsivo.

2. Se $A \leq B$, allora esiste una funzione di riduzione da A in B, ossia una $f: \mathbb{N} \to \mathbb{N}$ calcolabile, totale, e tale che $x \in A$ se e solo se $f(x) \in B$. Possiamo dire equivalentemente che $x \in \overline{A}$ se e solo se $f(x) \in \overline{B}$, quindi f è anche una funzione di riduzione da $\overline{A}$ in $\overline{B}$.

Esercizio 5 Data una stringa u, per ogni $n \leq |u|$, indichiamo con $[u]^n$ il prefisso di u di lunghezza n. Per esempio, se $u = abc$, si ha $[u]^0 = \epsilon$, $[u]^1 = a$, $[u]^2 = ab$, $[u]^3 = abc$.

1. Assumiamo che L sia decidibile, e sia $\mathcal{A}_L$ un algoritmo che decide L. Descrivere un algoritmo che decide $\{u \mid u' \in L \text{ per qualche } u' \text{ prefisso di } u\}$.

```
input u
for n=0 to |u|
  if (A_L([u]^n)) return true
return false
```

2. Assumiamo che L sia semidecidibile, e sia $\mathcal{A}_L^k$ l'esecuzione di (al più) k passi di un algoritmo che semi-decide L, che restituisce `false` se dopo k passi l'algoritmo che semi-decide L non è terminato. Descrivere un algoritmo che semi-decide $\{u \mid u' \in L \text{ per qualche } u' \text{ prefisso di } u\}$.

```
input u
k=0
while (true)
  for n=0 to |u|
    if (A_L^k([u]^n)) return true
  k++
```