Informatica Teorica

Appello d'esame del 16 luglio 2008

Si consideri la grammatica G seguente:

S → BAcB | ε
A → aAa | acBa
B → CcB | ε
C → aC | a

Esercizio 1 (punti 11)
Si fornisca un automa che riconosca il linguaggio L(G) generato da G.
L'automa deve appartenere alla classe di minima potenza riconoscitiva possibile per riconoscere L(G).
Si spieghi brevemente perché la classe dell'automa scritto è quella a potenza minima possibile.

Esercizio 2 (punti 12)

a. Si dica, giustificando brevemente la risposta, se è decidibile il problema di stabilire se L(G) intersecato con il linguaggio {(a*c)*} produce il linguaggio vuoto o no.

b. Si dica, giustificando brevemente la risposta, se è decidibile il problema di stabilire se, dato un generico automa a stati finiti A, il linguaggio L(A) riconosciuto da A è uguale al linguaggio L(G) oppure no.

c. Si dica, giustificando brevemente la risposta, se, ***fissata*** una grammatica G', è decidibile il problema di stabilire se, dato un generico automa a stati finiti A, il linguaggio L(A) riconosciuto da A è uguale al linguaggio L(G') generato da G' oppure no.

d. Si dica, giustificando brevemente la risposta, se è decidibile il problema di stabilire se, dati una ***generica*** grammatica G' ed un generico automa a stati finiti A, il linguaggio L(A) riconosciuto da A è uguale al linguaggio L(G') generato da G' oppure no.

Esercizio 3 (punti 10)
Si descrivano a grandi linee il funzionamento di una macchina di Turing deterministica e di una RAM che riconoscano L(G) e se ne valutino le complessità secondo la classe Θ , usando per la RAM il criterio di costo logaritmico.

Soluzioni schematiche

Esercizio 1

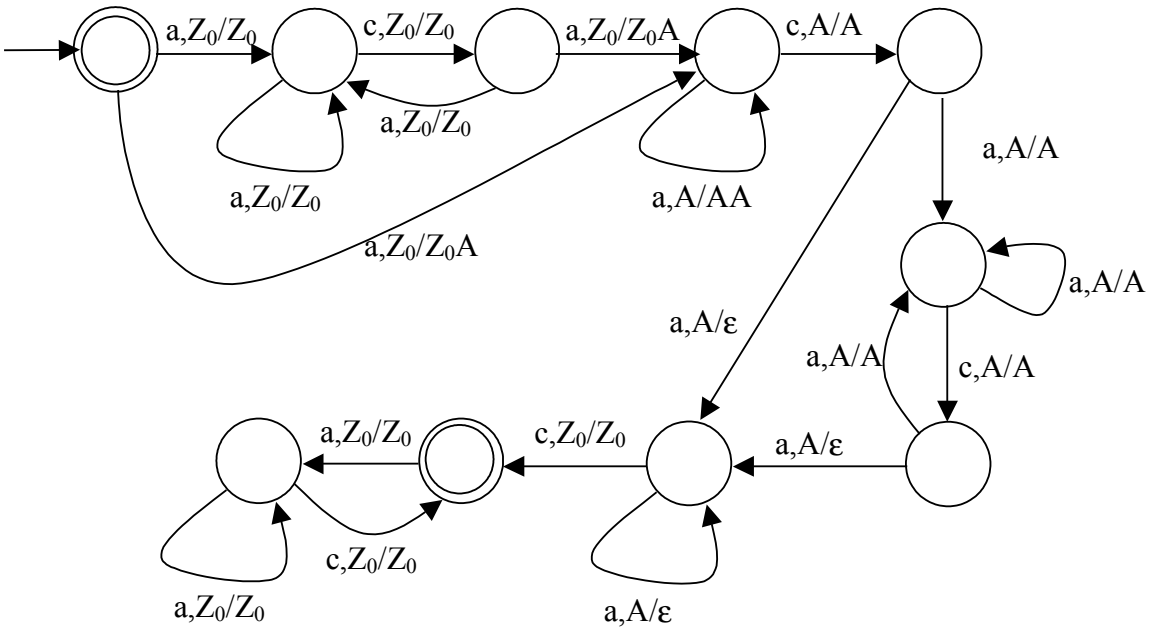
La grammatica genera il linguaggio:

$$L(G) = \{ (a^+c)^*a^nc(a^+c)^*a^nc(a^+c)^* \mid n > 0 \} \cup \varepsilon \ ,$$

dove $x^+ = x^* - \{\varepsilon\}$.

Per riconoscere L(G) è necessario e sufficiente un automa a pila nondeterministico. La pila è necessaria per effettuare il conteggio delle sottostringhe a^n. Anche il nondeterminsmo è necessario perchè con il solo determinismo non è possibile stabilire in quale punto delle sottostringhe fatte di 'a' è necessario iniziare ad usare la pila per contare tali 'a'. In altre parole, il nondeterminismo serve a discernere le sottostringhe del tipo $(a^+c)^*$, per le quali è sufficientemente un riconoscimento a stati finiti senza pile, da quelle del tipo a^n per le quali la pila è invece ovviamente necessaria.

Un possibile ND PDA che riconosce L(G) è il seguente:



Esercizio 2

a.Decidibile, la risposta è chiusa (Si/No). Anzi, in questo caso il problema è deciso, in quanto l'intersezione non è vuota.

b.Decidibile: L(G) non è un linguaggio regolare, quindi la risposta sarà sempre No, qualunque sia A.

c.Decidibile: se L(G') non è un linguaggio regolare la risposta sarà sempre No; se invece L(G') è regolare, siccome è decidibile il problema di stabilire se FSA riconoscono lo stesso linguaggio, ci sarà sempre una MT che, fissato un linguaggio regolare, determina se questo è uguale a quello riconosciuto da un FSA in input. Ora, fissato L(G'), non è detto che io sappia esattamente quale è questa MT (i linguaggi regolari sono un'infinità enumerabile, e non è detto che io capisca quale di questi è quello generato da G'), ma essa di certo esiste, rendendo il problema decidibile.

d.Indecidibile, in quanto riconducibile al problema di stabilire se una generica MT calcola una certa funzione (in questo caso, quella corrispondente all'FSA), oppure no.

Esercizio 3

Una macchina di Turing (deterministica) che riconosce L(G) può operare nella seguente maniera (al solito indichiamo con n la lunghezza dell'input):

- In primo luogo effettua una scansione completa dell'input per verificare che sia una stringa del tipo $(a^+c)^k$ con $k > 1$. Questa è una condizione necessaria perchè sia in L(G), e può essere verificata in tempo $\Theta(n)$.
- Se il test precedente è passato con successo, l'input appartiene a L(G) se e solo se esistono due sottostringhe (massimali, ossia racchiuse tra due 'c') di sole 'a' della stessa lunghezza. Questa verifica può essere fatta nella seguente maniera: per ogni sottostringa di 'a' incontrata sequenzialmente nell'input:
  - la sottostringa di 'a' viene ricopiata su un nastro di memoria;
  - si prosegue la scansione dell'input confrontando ogni successiva sottostringa di 'a' con quella presente sul nastro di memoria.

  Nel caso pessimo questo comporta una scansione completa dell'input per ogni sottostringa di 'a' incontrata. Possono esservi fino a $\Theta(n)$ sottostringhe di questo tipo, quindi la complessità temporale di caso pessimo è $\Theta(n^2)$, che è anche la complessità complessiva dell'algoritmo. La complessità spaziale è invece chiaramente $\Theta(n)$.

Una macchina RAM può usare un algoritmo simile a quello descritto per la macchina di Turing, usando ad esempio dei contatori per memorizzare lunghezza delle sottostringhe di 'a' incontrate. Si avrebbero dunque $\Theta(n)$ contatori, ciascuno memorizzante un intero $\Theta(n)$. Adottando il criterio di costo logaritmico, i $\Theta(n^2)$ confronti da effettuare porterebbero pertanto a una complessità temporale complessiva di $\Theta(n^2 \log n)$. La complessità spaziale, secondo le stesse considerazioni, sarebbe di $\Theta(n \log n)$.