Algoritmi e Principi dell'Informatica

Appello del 21 Febbraio 2013

Chi deve sostenere l'esame integrato (API) deve svolgere tutti gli esercizi in 3 ore.

Chi deve sostenere solo il modulo di Informatica teorica deve svolgere gli Esercizi 1, 2, 3 in 1 ora e 30 minuti.

Chi deve sostenere solo il modulo di Informatica 3 deve svolgere gli Esercizi 4, 5, 6 e in 1 ora e 30 minuti.

NB: i punti attribuiti ai singoli esercizi hanno senso solo con riferimento all'esame integrato e hanno valore puramente indicativo.

Modulo I

Esercizio 1 (punti 5)

Si considerino i seguenti linguaggi

$L_1 = \left\{ \left(a^{i_j} b^{i_j}\right)^k d \mid k > 0, 1 \leq j \leq k, i_j > 0 \right\}$ = {abd, aabbabd, aabbaabbd, abaabbd, … abaaabbbabd, … aabbabaaabbbd, …}

$L_2 = \left\{ a^h \left(b^{i_j} c a^{i_j}\right)^k d \mid h > 0,\ k > 0, 1 \leq j \leq k, i_j > 0 \right\}$ = {abcad, … aaabbcaad, aaabbbcaaad, aabbbcaaabcad, …}

1. Definire una grammatica non contestuale che generi L_1 e un automa a pila deterministico che lo accetti.
2. Definire una grammatica non contestuale che generi L_2 e un automa a pila, preferibilmente deterministico, che lo accetti.
3. Definire una grammatica non contestuale che generi $L_1 \cup L_2$. Se possibile, definire un automa a pila deterministico che lo accetti; altrimenti argomentare in modo informale ma convincente che il linguaggio $L_1 \cup L_2$ non può essere accettato da un automa a pila deterministico, e definire un automa a pila nondeterministico che accetti $L_1 \cup L_2$.

Esercizio 2 (punti 5)

Si specifichi in logica del prim'ordine una funzione f : A* × N → A, dove A è un alfabeto di simboli e N è l'insieme dei numeri naturali, che, data una stringa costruita sull'alfabeto A e un numero i, restituisce il carattere in posizione (i+1)-esima nella stringa (cioè il carattere con indice i nella stringa, dove 0 è l'indice del primo carattere, 1 del secondo, e così via). Nella specifica, è consentito l'uso delle seguenti funzioni:

$s \cdot t$ (concatenazione delle stringhe s e t)

$|s|$ (lunghezza di una stringa s)

e del predicato = (uguaglianza). Ovviamente *non* è consentito l'uso della notazione $s[i]$, che indica l'i-esimo carattere della stringa s.

Esercizio 3 (punti 5)

Si dica, giustificando brevemente la risposta, se la seguente funzione è calcolabile o no:

$f : N \times N \times N \rightarrow \{0, 1\}$

$f(0, 0, 0) = 0;$

$f(x, y, z) =$

se z è pari allora

0 se x è la radice quadrata intera esatta di y,

$\perp$ altrimenti;

se z è dispari allora

1 se ($\forall$x dispari $\exists$y pari e k, h $\in$ N tali che $k^x = h^y$,

$\wedge$

$\forall$x pari $\exists$y dispari e k, h $\in$ N tali che $k^x = h^y$)

$\perp$ altrimenti.

Modulo II

Esercizio 4 (punti 6)

Si risolva la seguente equazione alle ricorrenze (a meno della relazione O):

$T(n) = 2T(n-4) + O(\log(n))$.

Esercizio 5 (punti 6)

Scrivere un algoritmo che, dato un array A, determina se esiste un sottoarray (per sottoarray si intende una sequenza di elementi *consecutivi* dell'array base) di A che sia a sua volta scomponibile in 2 sottoarray tali che la somma degli elementi nel primo sia uguale alla somma degli elementi del secondo. L'algoritmo deve restituire il sottoarray più lungo tra quelli che soddisfano la condizione di cui sopra, se esiste.

Indicare la complessità temporale asintotica dell'algoritmo scritto.

Esercizio 6 (punti 5)

Si definisca (senza necessariamente codificarne tutti i dettagli) un macchina di Turing a nastro singolo che riconosca il linguaggio delle stringhe del tipo x#y (x, y $\in$ {a, b}*) con

(1) $|x| \neq |y|$ e

(2) |x| pari e |y| dispari o viceversa,

minimizzando la complessità temporale della macchina; se ne valuti la complessità spaziale e temporale a meno della relazione Θ.

Tali complessità cambierebbero se si rimuovesse il vincolo (2)?

Cosa cambierebbe se si usasse una macchina a k nastri invece di una macchina a nastro singolo?

Tracce di soluzione

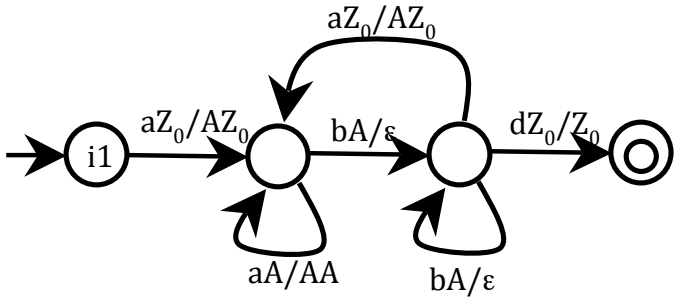
Esercizio 1

1. La grammatica seguente, con assioma S_1, genera L_1

$S_1 \rightarrow A\ S_1 \mid A\ d$

$A \rightarrow a\ A\ b \mid a\ b$

L'automa deterministico seguente accetta L_1.



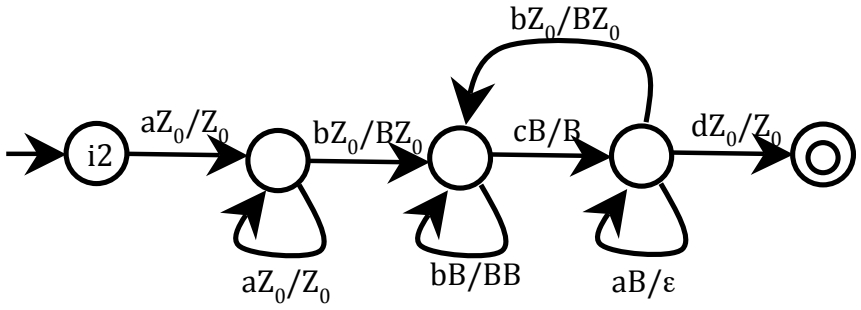
2. La grammatica seguente, con assioma S_2, genera L_2

$S_2 \rightarrow a\ S_2 \mid a\ X\ d$

$X \rightarrow C\ X \mid C$

$C \rightarrow b\ C\ a \mid b\ c\ a$

L'automa deterministico seguente accetta L_2.

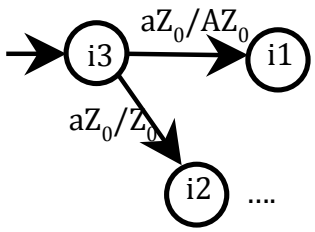


3. La grammatica, con assioma S, genera $L_1 \cup L_2$

$S \rightarrow S_1 \mid S_2$ … più le regole per i linguaggi L_1 ed L_2 …

L'automa a pila che accetta $L_1 \cup L_2$ non può essere deterministico: mentre legge il primo gruppo di simboli *a*, l'automa deve memorizzare nella pila un conteggio unario del numero di *a* e poi, mentre legge il primo gruppo di *b*, deve usare la pila per verificare (nel caso che la stringa $\in L_1$) che il numero delle *b*sia uguale al numero delle *a*. Ma facendo ciò il contenuto della pila va perso, quindi la pila non può essere usata per verificare (nel caso la stringa $\in L_2$) che il numero delle *a*che seguono la successiva*c*eguagli il numero delle *b*che la precedono.

Un automa nondeterministico che accetti $L_1 \cup L_2$ può essere ottenuto come semplice composizione dei due automi che accettano L_1ed L_2.



Esercizio 2

$\forall x\ \forall y\ \forall i\ (f(x, i) = y \leftrightarrow \exists w\ \exists z\ (x = w \cdot y \cdot z \wedge |y| = 1 \wedge |w| = i))$

Esercizio 3

f è calcolabile perché :

- è decidibile se z sia pari o dispari
- nel primo caso è decidibile stabilire se x sia la radice quadrata intera esatta di y o no e produrre 1 o, rispettivamente, $\perp$ nei due casi;
- nel secondo caso la domanda è chiusa (non importa stabilire se la condizione indicata sia vera o falsa: la funzione di fatto non dipende da x e y in quanto quantificate) e quindi una delle due macchine che per z dispari producono in output 1 oppure non terminano è la macchina che computa questa parte della funzione.

Esercizio 4

Mediante il metodo dell'albero di ricorsione si ottiene il guess $O(2^{n/4} \cdot \log(n))$: il costo di ogni livello è $O(2^k \log(n - 4k))$; quindi la profondità dell'albero è $k = n/4$.
Poi si verifica che, con opportune costanti, $T(n) \leq c \cdot 2^{n/4} \cdot \log(n)$.
In effetti, per arrivare a concludere che $T(n) \leq c \cdot 2^{n/4} \cdot \log(n)$, è utile mostrare un vincolo leggermente più stretto, e cioè che $T(n) \leq c \cdot 2^{n/4} \cdot \log(n) - bn^2$. Infatti si ha:

$$\begin{aligned} T(n) &= 2 \cdot T(n-4) + d \cdot \log(n) \leq 2c2^{((n-4)/4)} \log(n-4) - 2b(n-4)^2 + d \cdot \log(n) = \\ &= c2^{n/4} \log(n-4) - bn^2 - bn^2 + 16bn - 32 + d \cdot \log(n) \leq \\ &\leq (c2^{n/4} \log(n) - bn^2) - bn^2 + 16bn - 32 + d \cdot \log(n) \leq \\ &\leq c2^{n/4} \log(n) - bn^2 \end{aligned}$$

per b ed n sufficientemente grandi.

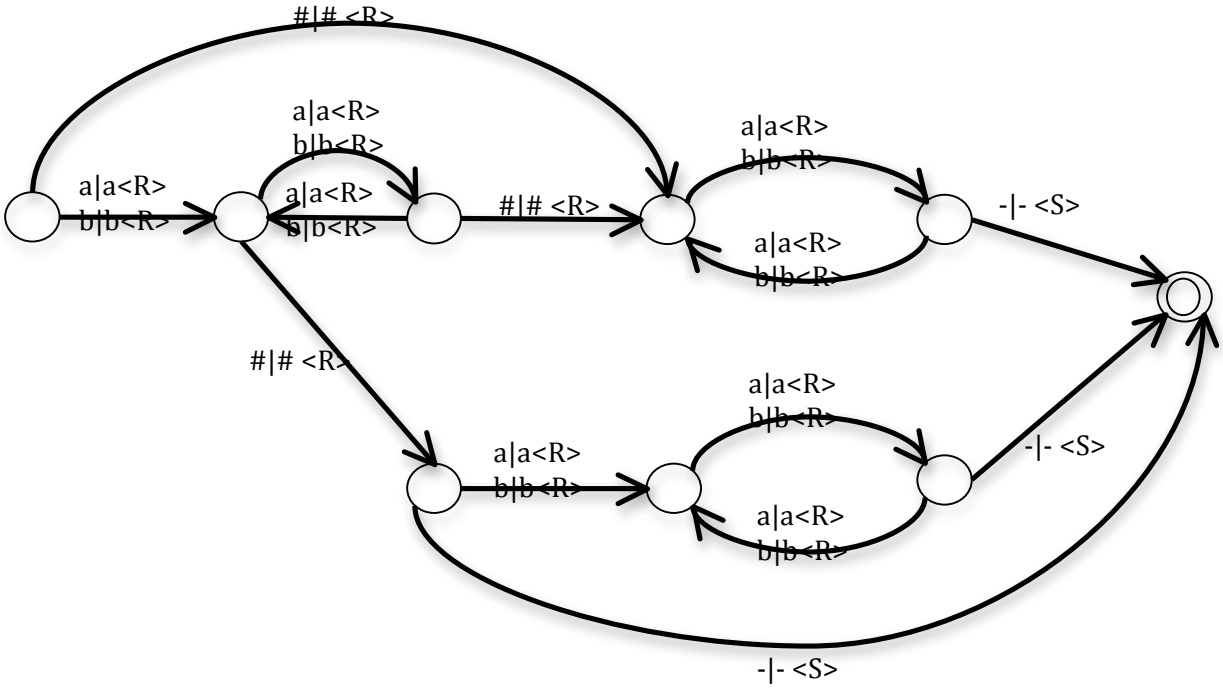
Esercizio 5

Un modo naturale per risolvere il problema consiste nell'elencare tutti gli n^2 sottoarray di A (lungo n), e controllare se questi hanno la proprietàdesiderata;tale verifica si può evidentemente fare in tempo O(n) per ogni sottoarray.

Di conseguenza, la complessità temporale complessiva è $\Theta(n^3)$.

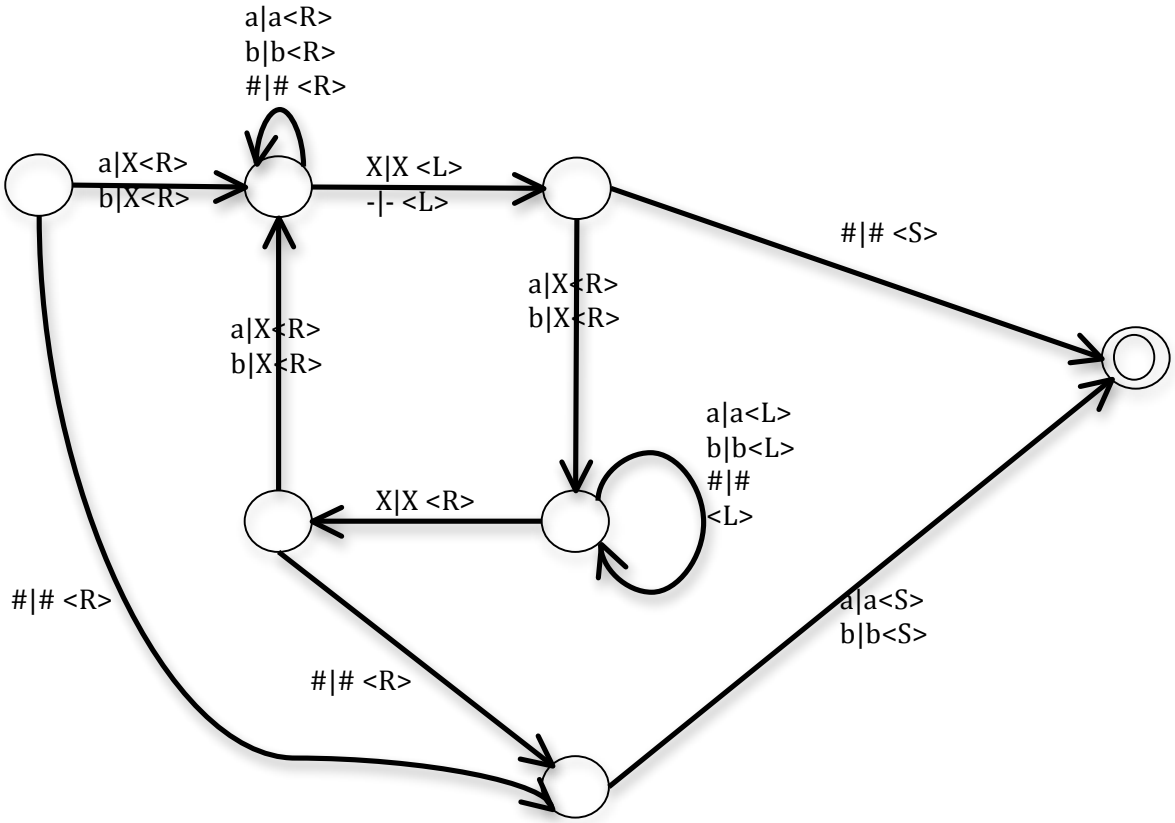
Esercizio 6

Una MT di Turing a nastro singolo che riconosce il linguaggio descritto opera con complessità temporale e spaziale Θ(n). Il linguaggio descritto è, infatti, un linguaggio regolare e il controllo sulla condizione (2) (che *implica la condizione* (1)) può essere fatto usando solo gli stati come nella macchina seguente.



La complessità spaziale è data dal fatto che l'unico nastro contiene la stringa di ingresso.

Se si rimuovesse il vincolo (2) il linguaggio non sarebbe più regolare e una macchina che minimizza la complessità temporale potrebbe operare nel seguente modo:



La complessità temporale diventerebbe quindi $\Theta(n^2)$, mentre quella spaziale rimarrebbe pari a $\Theta(n)$.

Se invece che una macchina a nastro singolo si usasse una macchina a k nastri le complessità cambierebbero nel seguente modo.

Nel primo caso la macchina a k nastri opererebbe esattamente come la macchina a nastro singolo, mantenendo quindi una complessità temporale $\Theta(n)$, mentre la complessità spaziale diventerebbe $\Theta(1)$, poiché i nastri di memoria e ingresso sono distinti e quindi la memoria in questo caso, non viene usata.

Rimossa la condizione (2), una macchina con un nastro di memoria può riconoscere il linguaggio scorrendo la prima parte della stringa e copiandola nel nastro di memoria. Una volta raggiunto il simbolo # legge la seconda parte della stringa, avvolgendo indietro il nastro in cui è stata copiato la prima parte della stringa. Se il nastro raggiunge Z_0 prima che la stringa sia finita o la stringa finisce prima di arrivare a Z_0 sul nastro, la parola viene accettata.

La complessità temporale diventa quindi $\Theta(n)$, così come quella spaziale.