A

Fondamenti dell'Informatica

A.A. 2005/2006

Corso A

Prima prova di esonero: 27/4/2006 ore 8.30 – 11.00

1. Dare la definizione di numerabilità e dimostrare che l'insieme $\mathbb{R}$ dei numeri reali non è numerabile. *(5 punti)*
2. Definire una macchina di Turing deterministica a nastro singolo. Definire il concetto di configurazione, di transizione, di riconoscimento e di accettazione di un linguaggio. Progettare una macchina di Turing deterministica in grado di riconoscere il linguaggio

   $L=\{w \in (0+1)^+ \mid w$ contiene un numero di 1 almeno doppio rispetto al numero di 0$\}$. *(7 punti)*

Svolgimento

Le parole di *L* sono sequenze di 0 e 1. Su di esse è imposto solo il vincolo che il numero di occorrenze delle cifre 1 sia ALMENO il doppio del numero di occorrenze delle cifre 0. Pertanto apparterranno a *L* le seguenti parole:

1, 11, 011, 101, 110, 111, 0111, 1011, 1101, 1110, 1111, …

L'algoritmo che risolve il problema di RICONOSCERE le parole di L è il seguente:

1. Cerca uno 0 e marcalo (con *)
2. Riporta la testina all'inizio della parola
3. Cerca il primo 1 e marcalo (sempre con *)
4. Cerca il secondo 1 e marcalo
5. Riporta la testina all'inizio della parola
6. Torna al passo 1

$M=\{\Gamma, \text{b}, Q, q_0, F, \delta\}$, con $\Gamma=\{0, 1, *, \text{b}\}$, $Q=\{ q_0, q_1, q_2, q_3, q_4, q_F \}$, $F=\{ q_F \}$

q_0: condizione iniziale, si va alla ricerca di uno 0 da marcare (se c'è)
q_1: trovato e marcato uno 0, si riporta la testina all'inizio della parola
q_2: trovato e marcato uno 0, si alla ricerca di un 1 da marcare (se c'è)
q_3: trovati e marcati sia uno 0 e sia un 1, si alla ricerca di un altro 1 da marcare (se c'è)
q_4: trovati e marcati sia uno 0 e sia due 1, si riporta la testina all'inizio della parola
q_5: verifica che la parola non sia vuota
q_F: stato finale

La matrice di transizione sarà:

	0	***1***	*****	***b***
q_0	(q_1, *, s)	(q_0, 1, d)	(q_0, *, d)	(q_5, b, s)
q_1	(q_1, 0, s)	(q_1, 1, s)	(q_1, *, s)	(q_2, b, d)
q_2	(q_2, 0, d)	(q_3, *, d)	(q_2, *, d)	-
q_3	(q_3, 0, d)	(q_4, *, s)	(q_3, *, d)	-
q_4	(q_4, 0, s)	(q_4, 1, s)	(q_4, *, s)	(q_0, b, d)
q_5	-	(q_F, 1, i)	(q_F, *, i)	-
q_F	-		-	-

3. Dimostrare che per ogni macchina di Turing non deterministica M esiste una macchina deterministica M' equivalente. Qual è il costo della simulazione? *(7 punti)*

4. Definire un programma RAM per il calcolo di f(n)=2^{2^n}, $n\geq 0$. Valutare la complessità del programma rispetto a un modello di costo logaritmico. A quale ordine di complessità porterebbe l'adozione di un modello di costo uniforme? (*7 punti*)

Svolgimento

Osserviamo che $f(n)=2^{2^n}=2^{2^{n-1}}\cdot 2^{2^{n-1}}= f(n-1)\cdot f(n-1)$. Pertanto sarà sufficiente un ciclo i=1, 2, …, n nel quale si moltiplicherà ripetutamente $f(i)$ per se stesso, partendo da $f(0)$=2.

	READ	1	$l(1)+l(n)$	$O(\log n)$	1
	LOAD	#2	$l(2)$	$O(1)$	1
	STORE	2	$l(2)+l(2)$	$O(1)$	1
	LOAD	1	$l(n)+l(1)$	$O(\log n)$	1
5	JZERO	13	$l(n)$	$O(\log n)$	$n+1$
	LOAD	2	$l(2^{2^{n-1}})+l(2)$	$O(2^{n-1})$	n
	MULT	2	$l(2^{2^{n-1}})+l(2)+l(2^{2^{n-1}})$	$O(2^n)$	n
	STORE	2	$l(2^{2^n})+l(2)$	$O(2^n)$	n
	LOAD	1	$l(n)+l(1)$	$O(\log n)$	n
	SUB	#1	$l(1)+l(n)$	$O(\log n)$	n
	STORE	1	$l(1)+l(n)$	$O(\log n)$	n
	JUMP	5	$l(1)+l(n)$	$O(\log n)$	n
13	WRITE	2	$l(2)+l(2^{2^n})$	$O(2^n)$	1
	HALT				

Pertanto la complessità, nel caso di modello a costi logaritmici, sarà $O(n2^n)$, cioè esponenziale. Al contrario, nel caso di modello a costi uniformi, sarà $O(n)$, cioè polinomiale.

N.B.: Questo prova quanto enunciato a lezione.

> **Teorema 4** *Nel modello a costi uniformi una RAM con l'operazione di moltiplicazione può calcolare in tempo polinomiale una funzione che richiede tempo esponenziale per essere calcolata da una macchina di Turing.*
>
> Per dimostrarlo è sufficiente considerare il calcolo della funzione $f(n) = 2^{2^n}$ su una RAM a costi uniformi dotate di operatore di moltiplicazione e su macchine di Turing.

Infatti, una MT che calcola $f(n)=2^{2^n}$ avrà un costo esponenziale, quanto meno perché dovrà utilizzare un numero di celle pari a 2^n per rappresentare $f(n)$. Questo risultato è coerente con quanto dimostrato sopra nel caso di modello con costi logaritmici e con un altro teorema che afferma che la simulazione di un programma RAM con MT ha costi polinomiali se per la RAM adottiamo un modello a costi logaritmici. In altri termini, con modello di costi logaritmici, tanto il programma RAM riportato sopra che la MT simulante avranno un costo esponenziale. Diversamente, se la

RAM ha costi uniformi, la MT che calcola $f(n)=2^{2^n}$ continuerà ad avere un costo esponenziale, mentre la RAM avrà un costo lineare in n, e verrà violata la Tesi di Cobham.

5. Dimostrare che la funzione

$$spot3(n) = \sum_{i=1}^{n} i^3$$

è ricorsiva primitiva (*4 punti*). Scrivere il programma SLF per il calcolo di *spot3(5)*. (*3 punti*)

Svolgimento

Osserviamo che la funzione *spot3(n)* si può definire ricorsivamente come segue:

spot3(0)=0
spot3(n+1)=somma(spot3(n),cubo(n+1))

quindi rispetto allo schema di ricorsione primitiva:

$f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n)$,
$f(x_1, \ldots, x_n, y + 1) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$

abbiamo

$g=0^{(0)}$
$h=somma(U_2^{(2)}, cubo(S(U_1^{(2)})))$

Inoltre si ha che *cubo(n)=prodotto(n,prodotto(n,n))*, ed è quindi ottenuta per composizione della funzione *prodotto*. Poiché le funzioni *somma* e *prodotto* sono ricorsive primitive, lo sono anche *cubo* e *spot3*.

Il programma SLF per il computo di spot3(5) è il seguente:

spot3(n) $\Leftarrow$ *if-then-else(n,0,somma(cubo(n),spot3(P(n))))*

cubo(x) $\Leftarrow$ *prodotto(x, prodotto(x,x))*

prodotto(x,y) $\Leftarrow$ *if-then-else(y,0,somma(y,prodotto(x,P(y))))*

somma(x,y) $\Leftarrow$ *if-then-else(y,x,S(somma(x,P(y))))*

spot3(5)

6. Definire una codifica mediante numeri interi per lo stato di una MREL quindi definire una funzione

$$R_\pi: \mathbb{N}^2 \to \mathbb{R}$$

che restituisca 1 se una coppia di interi corrisponde a una transizione di stato permessa dal programma MREL π, 0 altrimenti. (*5 punti*) [1]

[1] La totalizzazione di un punteggio superiore a 30 punti equivale al *30 con lode*.