A

Fondamenti dell'Informatica

A.A. 2003/2004

Corso A

Prima prova di esonero: 26/4/2005 ore 8.30 – 11.00

1. Definire una macchina di Turing deterministica *M* a nastro singolo e i concetti di configurazione e di transizione. Sintetizzare una macchina di Turing trasduttore che trovi il massimo di due numeri rappresentati in notazione binaria. Si supponga che i numeri abbiano lo stesso numero di cifre.

   Esempio: $q_0$11100001#11001000 $\vdash\!\!\!-\!\!\!-^*_M$ 11100001#11001000~~b~~q_F 11100001

   con q_0 stato iniziale e q_F stato finale.

   Si rappresenti la funzione di transizione mediante una matrice di transizione. Si specifichi per ogni stato qual è la funzione da esso svolta. (*7 punti*)

Svolgimento

Si copia il primo numero in coda ai due, trasformando gli 0 in a e gli 1 in b

q_I11100001#11001000$\vdash\!\!\!-\!\!\!-^*_M$ 11100001q_0#11001000bbbaaaab

Per questa copia occorrono sei stati. Quindi si inizia il confronto di ogni cifra del secondo numero con la corrispondente cifra del numero copiato. Se le coppie sono (0,a) e (1,b) si prosegue nel confronto, fino a esaurimento del primo numero, rimpiazzando le *a* con α e le *b* con β. Per questo occorrono altri 4 stati.

11100001q_0#11001000bbbaaaab$\vdash\!\!\!-\!\!\!-^*_M$ 11100001q_0#11001000ββbaaaab

Se si trova la coppia (0,b) allora il primo numero è il più grande. Quindi occorrerà semplicemente convertire tutte le restanti *a* e *b* in α e β rispettivamente. Per questo è sufficiente uno stato.

11100001q_0#11001000ββbaaaab$\vdash\!\!\!-\!\!\!-^*_M$11100001q_0#11001000βββαααaβ

Se si trova la coppia (1,a) allora il secondo numero è il più grande. Quindi occorrerà scrivere β al posto di *a* e entrare in un ciclo che completa la copia del 2° numero rimpiazzando tutte le *b* e le *a* rimaste (si sfrutta così l'ipotesi che i due numeri hanno stesso numero di cifre). Occorrono 3 stati per completare la copia.

Al termine avremo il massimo in termini di α e β. Con due stati sarà possibile riconvertirli tutti in 0 e 1 introducendo uno spazio b prima della prima cifra.

2. Definire una macchina di Turing non deterministica (MTND). Riportare un esempio di MTND riconoscitore con grado di non determinismo pari a 2 e mostrare un albero di computazione relativo all'accettazione di una stringa e uno relativo al rifiuto di una stringa. (*4 punti*). In generale, potresti simulare il comportamento di una MTND con una deterministica? Come? E quanto costerebbe la simulazione? (*2 punti*).

3. Definire un programma RAM per il calcolo di

   $\lambda n\lambda k.1+n+n^2+n^3+\ldots+n^k$ con $n,k\geq 0$.

   Valutare la complessità del programma rispetto a un modello di costo logaritmico. [Suggerimento: al solo fine di valutare la complessità si tenga conto del seguente risultato matematico $\sum_{j=0}^{k} n^j = \frac{n^{k+1}-1}{n-1}$] (*7 punti*)

Svolgimento

Si utilizzano 5 registri:

1	*3*	*3*	*4*	*5*
n	k	i=*n,n-1, …, 2, 1*	$\sum n^i$	n^i

	READ	1	l(1)+l(n)	O(log *n*)	1	O(log *n*)
	READ	2	l(2)+l(k)	O(log *k*)	1	O(log *k*)
	LOAD	#1	l(1)	O(1)	1	O(1)
	STORE	4	l(4)+l(1)	O(1)	1	O(1)
	STORE	5	l(5)+l(1)	O(1)	1	O(1)
	LOAD	2	l(2)+l(k)	O(log *k*)	1	O(log *k*)
7	STORE	3	l(2)+l(i)	O(log *k*)	k+1	O(*k*log *k*)
	JZERO	17	l(k)	O(log *k*)	k+1	O(*k*log *k*)
	LOAD	5	l(1)+l(n^i)	O(log *n*)	k	O(*k*log *n*)
	MULT	1	l(1)+l(n)+ l(n^i)	O(log *n*)	k	O(*k*log *n*)
	STORE	5	l(5)+l(n^i)	O(log *n*)	k	O(*k*log *n*)
	ADD	4	l(4)+l(n^i)+l($\sum n^i$)	O(*k*log *n*)	k	O(k^2log *n*)
	STORE	4	l(4)+l($\sum n^i$)	O(*k*log *n*)	k	O(k^2log *n*)
	LOAD	3	l(3)+l(i)	O(log *k*)	k	O(*k*log *n*)
	SUB	#1	l(1)+l(i)	O(log *k*)	k	O(*k*log *n*)
	JUMP	5	l(5)	O(1)	k	O(*k*)
17	WRITE	4	l(3)+ l($\sum n^i$)	O(log *n*)	1	O(*k*log *n*)
	HALT		1	O(1)	1	O(1)

Nello svolgimento si è tenuto conto di quanto suggerito, cioè che

$$\sum_{j=0}^{k} n^j = \frac{n^{k+1}-1}{n-1}$$

da cui discende che:

$$\log \sum_{j=0}^{k} n^j = \log \frac{n^{k+1}-1}{n-1} = \log(n^{k+1}-1) - \log(n-1) = O(\log n^{k+1}) = O(k \log n)$$

Quindi la complessità del calcolo della sommatoria è: O(k^2log *n*).

4. Dimostrare che la funzione "sommatoria di Gauss"

$$gauss(n) = \sum_{k=0}^{n} k$$

è ricorsiva primitiva (*4 punti*). Scrivere il programma SLF per il calcolo di *Gauss(10)*. (*3 punti*)

Svolgimento

Osserviamo che :

gauss(0) = 0

gauss(y+1) = (y+1) + gauss(y)

pertanto :

gauss(0)=0^0

gauss(y+1)=h(y,z)=somma(S(U^2_1(y,z)),U^2_2(y,z))

dove *somma* è la funzione ricorsiva primitiva definita a lezione. Quindi *gauss(x)* è ricorsiva primitiva perché definita mediante lo schema di ricorsione primitiva a partire da funzioni ricorsive primitive (somma e quelle di base).

Il programma SLF per il calcolo di *gauss(10)* è :

gauss(y) ← *if-then-else(y,0,somma(y,gauss(P(y))))*

gauss(10)

Alternativamente avremmo potuto scrivere:

gauss(0) = 0

gauss(y+1) = (y+ gauss(y))+1

da cui discende che :

gauss(0)=0^0

gauss(y+1)=h(y,z)= S(somma(U^2_1(y,z),U^2_2(y,z))

5. Dato un programma π, dimostrare che il predicato di Kleene:

$$T_\pi(x_1, \ldots, x_n, c) = \begin{cases} 1 & \text{se } c=\langle s_1, \ldots, s_F\rangle \text{ e } s_1, \ldots, s_F \text{ è la computazione di } \pi \text{ con input } x_1, \ldots, x_n; \\ 0 & \text{altrimenti} \end{cases}$$

è definibile per ricorsione primitiva. (*10 punti*) [1]

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