B

Fondamenti dell'Informatica

A.A. 2005/2006

Corso A

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

1. Dato un alfabeto $\Sigma=\{a_1, a_2, \ldots, a_n\}$ dare la definizione di parole o stringhe su Σ (Σ^*) e di linguaggio su Σ. Dimostrare che l'insieme delle parole è numerabile (suggerimento: definire un ordinamento lessicografico), mentre l'insieme dei linguaggi non lo è. Spiegare le implicazioni pratiche di quest'ultimo risultato. *(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

   $L=\{w\in(a+b)^+ \mid w$ contiene sequenze della stessa lunghezza e alternanti di a e b$\}$. *(7 punti)*

   Esempi di parole in L: aaabbbaaabbb, baba, abababab

   Esempi di parole non in L: aba, aab, a, abba

Svolgimento

Le parole di *L* sono sequenze di a e b. Su di esse è imposto il vincolo che devono essere "sequenze alternanti" del tipo a^nb^n, o b^na^n. Si nota immediatamente che se una parola inizia con a, avremo sequenze alternanti del primo tipo, mentre se inizia con b avremo sequenze alternanti del secondo tipo.

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

1. Leggi il primo carattere e marcalo (con $)
2. Se il primo carattere è una a (sequenza del primo tipo):
   2.1. Fino a quando non hai marcato tutte le a:
      2.1.1. Ricerca di una b da marcare che segue una serie di a
      2.1.2. Sposta la testina a sinistra sull'ultimo carattere marcato con $
      2.1.3. Leggi la prossima a e marcala (con $)
   2.2. Cerca il prossimo carattere significativo (non marcato): se è una a, marcala e vai ad 2.1., mentre se è ~~b~~ fermati e riconosci la parola
3. Altrimenti, se il primo carattere è una b (sequenza del secondo tipo):
   3.1. Fino a quando non hai marcato tutte le b:
      3.1.1. Ricerca di una a da marcare che segue una serie di b
      3.1.2. Sposta la testina a sinistra sull'ultimo carattere marcato con $
      3.1.3. Leggi la prossima b e marcala (con $)
4. Cerca il prossimo carattere significativo (non marcato): se è una b, marcala e vai ad 3.1., mentre se è ~~b~~ fermati e riconosci la parola

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

q_0: condizione iniziale, si va alla ricerca di un carattere (*a* o *b*) da marcare
q_a: sequenza del primo tipo:cerca una *b* da marcare (con *)
q_{ab}: sequenza del primo tipo, marcate *a* e *b*: torna indietro al $
$q_{a\$}$: sequenza del primo tipo, trovato $: si va alla ricerca di una *a* da marcare
q_b: sequenza del secondo tipo:cerca una *a* da marcare (con *)
q_{ba}: sequenza del secondo tipo, marcate *b* e *a*: torna indietro al $
$q_{b\$}$: sequenza del secondo tipo, trovato $: si va alla ricerca di una *b* da marcare

q_F: stato finale
La matrice di transizione sarà:

	a	***b***	***$***	*****	***Ƀ***
q_0	(q_a, $, d)	(q_b, $, d)	-	-	-
q_a	(q_a, a, d)	(q_{ab}, *, s)	-	(q_a, *, d)	-
q_{ab}	(q_{ab}, a, s)	-	($q_{a\$}$, $, d)	(q_{ab}, *, s)	-
$q_{a\$}$	(q_a, $, d)	-	-	($q_{a\$}$, *, d)	(q_F, Ƀ, i)
q_b	(q_{ba}, *, s)	(q_b, b, d)	-	(q_b, *, d)	-
q_{ba}	-	(q_{ba}, b, s)	($q_{b\$}$, $, d)	(q_{ba}, *, s)	-
$q_{b\$}$	-	(q_b, $, d)	-	($q_{b\$}$, *, d)	(q_F, Ƀ, i)
q_F	-		-	-	-

3. Enunciare e dimostrare il teorema relativo alla simulazione di una macchina a registri mediante una macchina di Turing [Suggerimento: Si consideri una MT multinastro dotata di nastro di input, nastro di output e tre nastri di lavoro.]. Analizzare il costo di simulazione. *(7 punti)*
4. Definire un programma RAM per il calcolo di $f(n)=2^{n!}$, $n\geq 0$. 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? *(9 punti)*

   [Suggerimento: al solo fine di valutare la complessità del programma RAM si tenga conto del seguente risultato matematico: $n!\approx\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ dove e è il numero di Nepero]

Osserviamo che $f(n)=2^{n!}=2^{1\cdot 2\cdot 3\ \ldots\ (n-1)\ \cdot n}=\left(\left(\left(\left(2^1\right)^2\right)^3\right).^{\cdot}\right)^n=f(n-1)^n$. Pertanto sarà sufficiente un ciclo i=1, 2, …, n nel quale si moltiplicherà ripetutamente $f(i)$ per se stesso i volte, partendo da $f(1)$=2. Il caso $f(0)$=2 sarà trattato a parte. I primi quattro registri conterranno, rispettivamente, l'indice del ciclo più esterno (che parte da n), il valore di f(i), l'indice del ciclo più interno, il valore massimo (i) dell'indice di ciclo più interno.

	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(1)*	*O(1)*	*1*
	STORE	4	*l(4)+l(n)*	*O(n)*	*1*
	LOAD	1	*l(1)+l(n)*	*O(log n)*	*1*
7	JZERO	25	*l(n)*	*O(log n)*	*n+1*
	LOAD	4	*l(4)+l(n)*	*O(log n)*	*n*
	STORE	3	*l(3)+l(n)*	*O(log n)*	*n*
10	LOAD	3	*l(3)+l(n)*	*O(log n)*	*n(n-1)/2*
	SUB	#1	*l(1)+l(n)*	*O(log n)*	*n(n-1)/2*
	STORE	3	*l(3)+l(n)*	*O(log n)*	*n(n-1)/2*
	JZERO	18	*l(n)*	*O(log n)*	*n(n-1)/2*

	LOAD	2	$l(2)+l(2^{n!})$	$O(n!)$	$n(n-1)/2$
	MULT	2	$l(2) + l(2^{n!})$	$O(n!)$	$n(n-1)/2$
	STORE	2	$l(2) + l(2^{n!})$	$O(n!)$	$n(n-1)/2$
	JUMP	10	$l(10)$	$O(1)$	$n(n-1)/2$
18	LOAD	4	$l(4)+l(n)$	$O(\log n)$	n
	ADD	#1	$l(1)+l(n)$	$O(\log n)$	n
	STORE	4	$l(4)+l(n)$	$O(\log n)$	n
	LOAD	1	$l(1)+l(n)$	$O(\log n)$	n
	SUB	#1	$l(1)+l(n)$	$O(\log n)$	n
	STORE	1	$l(1)+l(n)$	$O(\log n)$	n
	JUMP	7	$l(1)+l(n)$	$O(\log n)$	n
25	WRITE	2	$l(2) + l(2^{n!})$	$O(n!)$	1
	HALT				

Pertanto la complessità, nel caso di modello a costi logaritmici, sarà $O(n^2 n!)$. Tenendo conto della approssimazione di Stirling, $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$, si nota subito che la complessità è esponenziale $O(n^{n+2.5})$. Al contrario, nel caso di modello a costi uniformi, sarà $O(n^2)$, 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.*

Infatti, una MT che calcola $f(n)=2^{n!}$ avrà un costo esponenziale, quanto meno perché dovrà utilizzare un numero di celle pari a $n!$ 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 $f(n)=2^{n!}$ continuerà ad avere un costo esponenziale, mentre la RAM avrà un costo quadratico in n, e verrà violata la Tesi di Cobham.

5. Scrivere un programma SLF per il calcolo di 3^2 (suggerimento: esprimere la funzione prodotto $f(x,y)=x^y$ mediante schemi di composizione e di ricorsione primitiva e quindi tradurli in SLF). (*3 punti*).

Osserviamo che la funzione *potenza(x,y)* si può definire ricorsivamente come segue:

potenza(x,0)=1
potenza(x,y+1)=prodotto(x,potenza(x,y))

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=S($0^{(1)}$)
h=prodotto($U_1^{(3)}$, $U_3^{(3)}$))

Il programma SLF per il computo di 3^2 è il seguente:

potenza(x,y) ⇐ if-then-else(y,1,prodotto(x,potenza(x,P(y))))

prodotto(x,y) ⇐ if-then-else(y,0,somma(y,prodotto(x,P(y))))

somma(x,y) ⇐ if-then-else(y,x,S(somma(x,P(y))))

potenza(3,2).

6. Si enunci e si dimostri a grandi linee che la funzione calcolata da una macchina a registri elementare è una funzione ricorsiva parziale [Suggerimento: Si rappresenti lo stato di una macchina a registri elementare (l, $\langle r_1, \ldots, r_m \rangle$) mediante un numero intero (e.g., $2^l 3^{Pm(r1, \ldots, rm)}$), dove $P_m(r_1, \ldots, r_m)$ è una funzione che associa una m-pla con un intero (quale?) Quindi si definisca una rappresentazione per una computazione finita $s_1, \ldots, s_t$, e una funzione binaria $T_\pi(x_1, \ldots, x_n, c)$. Come si può riscrivere la funzione $\lambda x_1 \ldots \lambda x_n.f(x_1, \ldots, x_n)$ calcolata da un programma π di una macchina a registri elementare?]. (*7 punti*) [1]

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