Fondamenti dell'Informatica

A.A. 2002/2003

Corsi A e B

Prima prova di esonero: 24/4/2003 ore 9.00 – 11.00 (fac simile)

1. Sintetizzare una macchina di Turing trasduttore che computi la funzione identità sugli interi positivi scritti in notazione decimale. Si rappresenti la funzione di transizione mediante una matrice di transizione. Si specifichi per ogni stato qual è la funzione da esso svolta. (*6 punti*)

   Esempio: $q_0129 \vdash\!\!\!-\!\!*_M 129\text{b}q_F129$ con q_0 stato iniziale e q_F stato finale

2. Fornire la definizione di Macchina di Turing non deterministica e descriverne il comportamento mediante un esempio a scelta dello studente (assicurarsi che l'esempio non sia anche un caso di Macchina di Turing deterministica). Mostrare, sempre mediante l'esempio, una stringa accettata e una rifiutata dalla macchina di Turing. (*6 punti*)

3. Si enunci e si dimostri il teorema secondo il quale una macchina di Turing può essere simulata mediante una RAM (Suggerimento: Si consideri una MT con nastro semi-infinito e con alfabeto di nastro $\Gamma=\{0,1\}$). Qual è il costo della simulazione? (*6 punti*)

4. Dimostrare che la funzione $f(x,y)=x^y$ è ricorsiva primitiva (Suggerimento: si definisca ricorsivamente x^y assumendo che il prodotto è definibile mediante una funzione ricorsiva primitiva) (*3 punti*)

5. Si enunci e si dimostri il teorema della ricorsione (o teorema di Kleene). (*7 punti*)

6. Provare che l'insieme $K=\{x \mid \varphi_x(x) \text{ è definita}\}$ non è ricorsivo per qualche enumerazione delle funzioni ricorsive. (*5 punti*)[1]

Esercizio n. 1

Sia $M=\langle \Gamma, \text{b}, Q, q_I, F, \delta \rangle$ la macchina di Turing trasduttore che calcola la funzione identità su interi positivi con rappresentazione decimale. Allora l'alfabeto dei simboli del nastro è:
$\Gamma=\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \text{b}, \underline{1}, \underline{2}, \underline{3}, \underline{4}, \underline{5}, \underline{6}, \underline{7}, \underline{8}, \underline{9}\}$

L'algoritmo consiste nei seguenti passi:

1. se la testina è posizionata su un simbolo 0, 1, 2, …, 9 rimpiazzalo con il corrispondente barrato e vai a 3
2. se la testina è posizionata su b spostati a destra e fermati
3. spostati fino alla ricerca del primo b a destra
4. supera tutte le cifre a destra del b
5. al successivo b trovato a destra scrivi la cifra cancellata
6. torna indietro alla prima cifra barrata trovata, rimpiazzala con quella non barrata corrispondente, portati a destra e torna al punto 1.

Quindi gli stati in Q sono:
q_I : stato iniziale
q_i : ricorda di aver letto la cifra i, i=0,1, …, 9 mentre cerchi il primo b
$q_{i\text{b}}$: ricorda di aver letto la cifra i, i=0,1, …, 9 e di aver superato il primo b a destra
q_s : la cifra è copiata e devi tornare indietro alla prima cifra barrata.
q_F : stato finale
$F=\{ q_F \}$

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

per un totale di 23 stati. Per comodità rappresenteremo solo 5 stati, ovvero q_I, q_s, q_F e i generici stati q_i e $q_{i\text{b}}$ (sottintendendo che i=0, 1, …, 9). Ciò è possibile perché le transizioni per q_i e $q_{i\text{b}}$ saranno identiche, a parte il simbolo che verrà barrato (che indicheremo genericamente con $\bar{i}$) e che verrà depositato su nastro (indicato genericamente con *i*).

La matrice di transizione sarà:

	i	$\bar{i}$	***b***
q_I	(q_i, $\bar{i}$, d)	-	(q_F, b, d)
q_i	(q_i, i, d)	-	($q_{i\text{b}}$, b, d)
$q_{i\text{b}}$	($q_{i\text{b}}$, i, d)	-	(q_s, i, s)
q_s	(q_s, i, s)	(q_I, i, d)	(q_s, b, s)
q_F	-	-	-

Esercizio n. 2

Le macchine di Turing non deterministiche (MTND) sono definite come le loro corrispondenti deterministiche, con la sola differenza che la funzione di transizione può assumere un insieme di valori in corrispondenza di una coppia stato-simbolo. Formalmente, una Macchina di Turing non deterministica *M* è una sestupla $M = \langle \Gamma, b, Q, q_0, F, \delta_N \rangle$, in cui Γ è l'alfabeto dei simboli di nastro, $b \notin \Gamma$ è un carattere speciale denominato blank, *Q* è un insieme finito e non vuoto di stati, $q_0 \in Q$ è lo stato iniziale, $F \subseteq Q$ è l'insieme degli stati finali e la funzione di transizione δ_N è una funzione parziale $\delta_N: Q \times \Gamma \cup \{b\} \mapsto P(Q \times \Gamma \cup \{b\} \times \{d, s, i\})$, dove la notazione $P(X)$ sta a indicare l'insieme delle parti dell'insieme X.

Come per le MT deterministiche, anche per le MTDN si definiscono il concetto di configurazione x*q*y dove $x \in L_\Gamma$ e $y \in R_\Gamma$, e di transizione da una configurazione *c* a un'altra *c*'. La differenza sostanziale è che la transizione non è univocamente determinata e quindi c'è una scelta di applicazione di δ_N su *c*. Poiché una computazione si compone di diverse transizioni, si possono realizzare diverse computazioni a partire da una configurazione iniziale.

Le MTND sono tipicamente utilizzate come dispositivi per problemi decisionali. In questo contesto, alcune computazioni massimali saranno dette accettanti mentre altre rifiutanti, in dipendenza dello stato della configurazione nella quale termina la computazione massimale. Dato un alfabeto $\Sigma \subseteq \Gamma$, si dirà che una stringa $x \in \Sigma^*$ è accettata dalla macchina *M* se esiste almeno *una* computazione accettante $c_0, \ldots, c_n$ di M, con $c_0 = \{q_0 x\}$, ovvero se esiste una computazione accettante che parte dalla configurazione iniziale. Al contrario, si dirà che $x \in \Sigma^*$ è rifiutata da *M* se *tutte* le computazioni di *M*, a partire dalla configurazione $\{q_0 x\}$ sono rifiutanti.

Un esempio di MTND è il seguente:

	0	***1***	***b***
q_0	{(q_0,0,d), (q_1,1,d)}	{(q_0,1,d)}	-
q_1	{(q_1,0,d)}	{(q_F,1,d)}	-
q_F	-	-	-

che accetta le stringhe che contengono uno 0 seguito da almeno un 1 (esempio 01) mentre rifiutano le altre (ad esempio, 00).

Esercizio n. 3

In questa dimostrazione assumiamo per semplicità che la macchina di Turing abbia un solo nastro semi-infinito e abbia come alfabeto i due soli simboli 0 e 1. Allora esiste una RAM con relativo programma *P* tale che se *M* realizza la transizione dalla configurazione iniziale q_0x alla configurazione finale $q_F\ y$ e se la RAM è inizializzata con la stringa *x* nei registri 2, …, $|x|+1$, al termine della computazione la macchina a registri avrà nei registri 2, …, $|y|+1$ la stringa *y*.

DIMOSTRAZIONE. Alla base di questa dimostrazione c'è la corrispondenza stabilita tra la cella *i* del nastro di M e la cella *i*+1 della memoria della RAM. La prima cella è usata per rappresentare la posizione della testina di M e viene inizializzata a 2 in modo da puntare alla prima cella sotto la testina di lettura/scrittura. Per ogni stato q_i di *M* sono possibili due transizioni in base al simbolo letto, cioè $\delta(q_i,1)$ e $\delta(q_i,0)$. La dimostrazione consiste nel mostrare che ogni coppia di transizioni è simulabile con una parte del programma P. Il passaggio di stato da q_i a q_j sarà reso mediante un salto alla parte di codice che simula le transizioni relative a q_j. Ad esempio, se $\delta(q_1,0)=<q_2,\alpha,i>$ e $\delta(q_1,1)=<q_1,\alpha,s>$ allora P contiene il seguente frammento:

	.....		
q_1	LOAD	(1)	acquisizione del contenuto del nastro
	JGTZ	q_1'	lettura di un 1 o di uno 0?
	LOAD	#α	
	STORE	(1)	scrittura di α sul nastro
	JUMP	q_2	aggiornamento dello stato
q_1'	LOAD	#α	
	STORE	(1)	scrittura di α sul nastro
	LOAD	1	
	SUB	#1	
	STORE	1	movimento a sinistra della testina
	JUMP	q_1	aggiornamento dello stato
q_2	.....		

In questo codice il simbolo α può essere 0 o 1.

Il costo complessivo della simulazione è O(T log T), dove T è il numero di transizioni realizzate in una computazione da *M*.

Esercizio n. 4

Vogliamo dimostrare che la funzione *potenza* = $\lambda x\lambda y.x^y$ è ricorsiva primitiva. La funzione è esprimibile ricorsivamente come segue:

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

Per definizione, se le funzioni $h^{(n+2)}$, $g^{(n)}$ sono ricorsive primitive allora anche la funzione $f^{(n+1)}$ definita come segue:
- $f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n)$,
- $f(x_1, \ldots, x_n, y + 1) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$

è ricorsiva primitiva.

In questo caso, *n*=1, $g=\lambda x.S(0^1(x))$ e $h=\lambda x\lambda y\lambda z.U_1^{(3)}(x,y,z)*U_3^{(3)}(x,y,z)$. Poiché le funzioni S, 0^1, $U_1^{(3)}$, $U_3^{(3)}$ e * sono ricorsive primitive anche la funzione potenza è ricorsiva primitiva.

Esercizio n. 5

Il teorema della ricorsione (o di Kleene) è uno dei risultati più importati della teoria generale della calcolabilità, cioè della teoria astratta della calcolabilità, uniforme rispetto alla classe di macchine che esegue il calcolo. Ricordiamo che nella teoria generale della calcolabilità le funzioni ricorsive vengono enumerate sulla base di una enumerazione dei programmi/macchine definiti per qualche modello di calcolo della stessa potenza di quello delle macchine di Turing deterministiche. Il valore x associato alla funzione ricorsiva j_x viene detto indice *indice della funzione*.

L'enunciato del teorema di Kleene è il seguente:

Sia data una enumerazione delle funzioni ricorsive. Se t è una funzione calcolabile totale, allora esiste $e \in \mathbb{N}$ *tale che* $j_e = j_{t(e)}$.

Dimostrazione. Inizialmente dimostriamo il teorema per una qualunque enumerazione delle macchine di Turing. Consideriamo la famiglia parametrica di MT M[u] che calcolano le funzioni parziali:

$$\varphi_{g(u)}(x) = \begin{cases} \varphi_{\varphi_u(u)}(x) & \text{se } \varphi_u(u) \text{ è definita} \\ \text{indefinita} & \text{altrimenti} \end{cases}$$

In altri termini, la macchina di Turing M[u] cerca inizialmente di calcolare $z = j_u(u)$. Se questa computazione termina, cioè se $j_u(u)$ è definita, allora utilizza questo risultato per calcolare $j_z(x)$. Ovviamente, ogni M[u] definirà un indice di funzione che essa calcola nel modo suddetto. Indichiamo con $g(u)$ la funzione che ci restituisce questo indice per ogni macchina nella famiglia parametrica M[u] (osserviamo che g è ricorsiva, in base a quanto stabilito per enumerare le MT). Quindi alla famiglia parametrica M[u] corrisponde una famiglia parametrica di funzioni ricorsive $j_{g(u)}$. Consideriamo ora la funzione $\lambda x.t(g(x))$. Essa è una funzione ricorsiva, perché composizione di t che è calcolabile totale e quindi ricorsiva, e di g che abbiamo appena osservato essere ricorsiva. Quindi deve esistere un indice v per tale funzione, ovvero deve esistere una MT che la calcola. Possiamo quindi scrivere che $j_v = \lambda x.t(g(x))$. Abbiamo pertanto che:

$j_{g(v)}(x) = j_{j_v(v)}(x) = j_{g(t(v))}(x)$

Ponendo $e = g(v)$ il teorema è dimostrato.

Al fine di estendere tale proprietà ad enumerazioni definite da altri modelli di calcolo, osserviamo che, dati due modelli di calcolo MC_1, MC_2 dotati di potere computazionale equivalente alle macchine di Turing, esiste una funzione calcolabile che associa al codice di una macchina in MC_1 il codice di una macchina in MC_2. Quindi se esiste j_v per le macchine di Turing allora la stessa funzione potrà essere calcolata da un altro modello di calcolo MC_2 ma avrà eventualmente un indice diverso w.

Esercizio n. 6

Sia $K = \{x \mid \varphi_x(x) \text{ è definita}\}$. Per assurdo, se K fosse ricorsivo, esisterebbe una funzione ricorsiva totale caratteristica dell'insieme, così definita:

$$C_K(x) = \begin{cases} 1 & \text{se } x \in K, \text{ ovvero se } \varphi_x(x) \text{ è definita} \\ 0 & \text{altrimenti} \end{cases}$$

Ma questa non può essere una funzione ricorsiva (tanto meno totale). Infatti se lo fosse, potremmo definire una funzione ricorsiva:

$$f(x) = \begin{cases} 1 & \text{se } C_K(x)=0 \\ \text{indefinita} & \text{altrimenti} \end{cases}$$

Detto e il suo indice, la valutazione di $f(e) = \varphi_e(e)$ ci porterebbe a una contraddizione.