Informatica Teorica

Prima prova in itinere - 7 Maggio 2009
Sezione Cremona+Como

Tempo a disposizione: 2h

NB: il punteggio è espresso in 13-esimi e riflette il peso relativo della prova rispetto all'intero esame il cui punteggio è in 30-esimi. Come già in altre occasioni è tuttavia possibile ottenere un punteggio complessivo superiore a 13/13.

Esercizio 1 (10/13 punti)

Parte 1
Si formalizzi la nozione di automa riconoscitore finito *deterministico* FSA2W che è dotato di 2 testine per il nastro di ingresso: una *sinistra*, che inizialmente è posizionata all'inizio della stringa in ingresso, e si muove da sinistra verso destra, ed una *destra*, che inizialmente è posizionata alla fine della stringa in ingresso, e si muove da destra verso sinistra.
Inoltre, gli stati dell'automa sono partizionati in 2 sottoinsiemi, Q_1 e Q_2, tali che, se l'automa si trova in uno stato dell'insieme Q_1, esso legge con la testina sinistra (che si muove da sinistra verso destra), mentre se l'automa si trova in uno stato dell'insieme Q_2, esso legge con la testina destra (che si muove da destra verso sinistra).
L'automa può fare una mossa solo se la testina sinistra si trova più a sinistra della testina destra sulla stringa in ingresso (o, alla peggio, sullo stesso carattere della testina destra).

Parte 2
Dire, giustificando la risposta, quale è la potenza espressiva del modello FSA2W rispetto agli FSA abituali (dire se è equipotente, strettamente più potente, strettamente meno potente, o nessuno di questi).

Parte 3
Dire, giustificando la risposta, quale è la potenza espressiva del modello FSA2W rispetto ai PDA deterministici (equipotente, strettamente più potente, ecc.).

Esercizio 2 (5/13 punti)
Scrivere una grammatica G che generi il linguaggio L costruito sull'alfabeto {a, b, c}, costituito da tutte e sole le stringhe in cui il numero n_a di a è diverso dal numero n_b di b, ed il numero n_c di c è tale che $n_c = n_a + n_b$.

Soluzioni

Esercizio 1

Parte 1

Un FSA2W è una tupla (Q, I, δ, q_0, F) dove $Q = Q_1 \cup Q_2$ (con $Q_1 \cap Q_2 = \emptyset$), ed I, q_0, e F hanno lo stesso significato che negli automi a stati finiti tradizionali. La funzione di transizione $\delta: Q \times I \rightarrow Q$ definisce, per ogni coppia (q, i) di stato corrente e carattere in input i, lo stato prossimo q', cioè $\delta(q,i) = q'$.

Una configurazione c di un FSA2W è definita come la coppia $\langle q, x \rangle$ di stato corrente e stringa ancora da leggere.

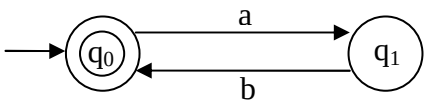
La transizione $c \vdash c'$ tra 2 configurazioni $c = \langle q, x \rangle$ e $c' = \langle q', x' \rangle$ è definita se e solo se:

- o $x = iy$ ($i \in I$, $y \in I^*$), $q \in Q_1$, $\delta(q,i) = q'$, e $x' = y$
- oppure $x = yi$ ($i \in I$, $y \in I^*$), $q \in Q_2$, $\delta(q,i) = q'$, e $x' = y$

Una stringa x è accettata se e solo se, detta $c_0 = \langle q_0, x \rangle$ la configurazione iniziale, si ha $c_0 \vdash^* c_F$, laddove $\vdash^*$ è la chiusura riflessiva e transitiva della relazione $\vdash$, e c_F è una configurazione $\langle q_F, \varepsilon \rangle$ tale che $q_F \in F$.

Parte 2

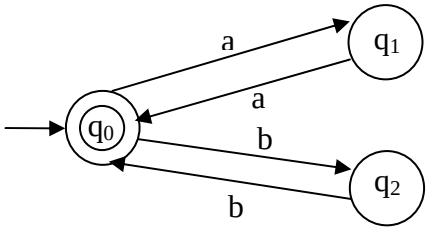
E' facile vedere che un FSA è un caso particolare di FSA2W, in cui, per esempio, $Q_1 = Q$ e $Q_2 = \emptyset$. Inoltre, il seguente FSA2W (con $q_0 \in Q_1$ e $q_1 \in Q_2$) riconosce il linguaggio $a^n b^n$, che, come è noto, non è riconoscibile da FSA.



Di conseguenza, gli FSA2W sono strettamente più potenti degli FSA.

Parte 3

E' facile vedere come il seguente FSA2W (con $q_0 \in Q_1$ e $q_1, q_2 \in Q_2$) riconosca il linguaggio ww^R ($w \in \{a,b\}^*$), che non è riconoscibile da PDA deterministici.



D'altro canto, il linguaggio fatto di tutte e sole le stringhe della forma {a, b}* in cui $n_a = n_b$ è riconoscibile da un PDA deterministico, ma non da un FSA2W, in quanto questi non hanno la capacità di "contare" in modo illimitato, e le a e b possono essere in posizioni arbitrarie nella stringa. Di conseguenza, gli FSA2W hanno potenza riconoscitiva non confrontabile con quella dei PDA deterministici.

Esercizio 2

$S \rightarrow ABCCS \mid X \mid Y$
$X \rightarrow ACX \mid AC$
$Y \rightarrow BCY \mid BC$
$AB \rightarrow BA$
$BA \rightarrow AB$
$AC \rightarrow CA$
$CA \rightarrow AC$
$BC \rightarrow CB$
$CB \rightarrow BC$
$A \rightarrow a$
$B \rightarrow b$
$C \rightarrow c$

Informatica Teorica

Seconda prova in itinere - 30 Giugno 2009
Sezione Cremona+Como

Tempo a disposizione: 2h

NB: il punteggio è espresso in 17-esimi e riflette il peso relativo della prova rispetto all'intero esame il cui punteggio è in 30-esimi. Come già in altre occasioni è tuttavia possibile ottenere un punteggio complessivo superiore a 17/17.

Esercizio 1 (8/17 punti)

Si formalizzi mediante formule logiche il funzionamento di una Macchina di Turing *a nastro singolo deterministica* che riconosce un linguaggio costruito sull'alfabeto A ed ha un insieme di stati Q, in cui il sottoinsieme $F \subseteq Q$ corrisponde all'insieme degli stati finali.
Più precisamente, si usino i seguenti predicati logici:

- `T(k, j) : A` $\cup$ {_, Z_0} è una funzione che restituisce il contenuto della cella j-esima del nastro della MT al passo k di computazione; il nastro della MT è infinito solo a destra, e la prima cella del nastro ha posizione 0 (ed inizialmente contiene il simbolo Z_0).
- `pos(k) :` $\mathbb{N}$ è una funzione che restituisce la posizione della testina del nastro della MT al k-esimo passo di computazione.
- `S(k) : Q` è una funzione che restituisce lo stato della MT al k-esimo passo di computazione.
- `delta(q, i) : Q` $\times$ `A` $\times$ `{L, S, R}` è la funzione di transizione che, a partire da uno stato $q \in Q$ e da un simbolo $i \in A$, ritorna una tripla di elementi <q', i', m'> in cui il primo elemento rappresenta il prossimo stato della MT, il secondo il simbolo i' da sostituire a i, ed il terzo lo spostamento della testina del nastro della MT.
- `halt(k)` è un predicato che rappresenta quando la MT è in configurazione di halt, cioè è vero se al passo di computazione k la MT è bloccata.
- `accetta(k)` è un predicato che rappresenta quando la MT è in configurazione di accettazione, cioè è vero se al passo di computazione k la MT accetta.

Definire mediante formule logiche:

- La configurazione iniziale della MT: organo di controllo in stato q_0, nastro contenente Z_0 seguito dalla stringa di input e poi tutti "blank", testina su Z_0.
- La mossa della MT, e l'effetto che ha sullo stato dell'organo di controllo e sul nastro.
- La condizione in cui la MT è in configurazione di "halt".
- La condizione in cui la MT è in configurazione di accettazione.

Esercizio 2 (6/17 punti)

Il signor Tom Garmin ha deciso di creare un programma che controlli l'aggiornamento del software del suo navigatore satellitare, ma ha qualche dubbio sulla fattibilità di ciò che vuole fare.

Per aiutarlo, si risponda alle seguenti domande, motivando brevemente, ma in modo preciso, le risposte.

1. E' decidibile il problema di stabilire se, dato un generico programma che prende in ingresso mappe stradali e indirizzi di origine e di destinazione di un viaggio, il programma ritorna, per ogni coppia origine/destinazione, la strada più breve nella mappa indicata?
2. E' semidecidibile il problema di stabilire se, dato un generico programma come quello descritto al punto 1, per qualche coppia origine/destinazione il programma ritorna una strada che *non è* la più breve possibile?
3. E' decidibile il problema di stabilire se, dato un generico programma come quello descritto al punto 1, di cui però si sa che termina la computazione per ogni coppia origine/destinazione in ingresso, e data la cartografia stradale dell'Italia del 30/6/2009, il programma, presi 2 indirizzi qualsiasi nella provincia di Milano, ritorna la strada più breve tra di essi?

Esercizio 3 (6/17 punti)

Dire se sono vere o false le seguenti affermazioni.

1. Date una qualunque MT a nastro singolo ed una macchina RAM (valutata a criterio di costo logaritmico) che risolvono lo stesso problema, la MT a nastro singolo ha costo temporale strettamente maggiore (secondo la relazione Θ).
2. E' possibile trovare una MT a k nastri ed una macchina RAM (valutata a criterio di costo costante) che risolvono lo stesso problema, tali che la MT ha costo temporale strettamente minore della macchina RAM (secondo la relazione Θ).
3. Data una qualunque macchina RAM con costo temporale (valutato a criterio di costo logaritmico) $T_R(n)$, è sempre possibile costruire una MT a nastro singolo che risolve lo stesso problema con costo temporale $\Theta(T_R^4)$.

Soluzioni

Esercizio 1

Configurazione iniziale:
$S(0) = q_0 \wedge pos(0) = 0 \wedge T(0, 0) = Z_0 \wedge$
$\exists p\ (p > 0 \wedge \forall j\ (j < p \rightarrow T(0,j) \neq \_ \wedge (T(0,j) \neq Z_0 \vee j = 0)) \wedge \forall j\ (j \geq p \rightarrow T(0,j) = \_))$

Mossa:
$\forall\ k,q,j,i,q',i',M\ (S(k) = q \wedge pos(k) = j \wedge T(k, j) = i \wedge delta(q, i) = <q', i', M>$
$\rightarrow$
$S(k+1) = q' \wedge T(k+1, j) = i' \wedge$
$(M = S \rightarrow pos(k+1) = j) \wedge (M = R \rightarrow pos(k+1) = j+1) \wedge$
$(M = L \wedge j>0 \rightarrow pos(k+1) = j-1)$

halt:
$\forall\ k\ (halt(k) \leftrightarrow \exists\ q, j, i\ (S(k) = q \wedge pos(k) = j \wedge T(k, j) = i \wedge$
$(\neg\exists q',i',M(delta(q, i) = <q', i', M>) \vee$
$\exists q',i'(delta(q, i) = <q', i', L> \wedge j = 0))))$

accettazione:
$\forall\ k\ (accetta(k) \leftrightarrow halt(k) \wedge S(k) \in F)$

Esercizio 2

1. Non è decidibile, per il teorema di Rice. Infatti, gli insiemi delle mappe possibili, degli indirizzi di origine/destinazione, e dei percorsi sono tutti quanti enumerabili, per cui è facile ridurre il problema in oggetto al calcolo di funzioni $\mathbb{N}\rightarrow\mathbb{N}$. I programmi che calcolano sempre la strada più breve tra origine e destinazione non sono l'insieme vuoto, e non sono l'insieme universo, dunque non sono un insieme decidibile per il teorema di Rice.

2. E' semidecidibile. E' sufficiente enumerare con la solita tecnica diagonale i passi di simulazione, le possibili mappe stradali ed i possibili indirizzi; se ad un certo punto si trova una combinazione mappa/origine/destinazione per la quale la risposta non è quella ottimale, questa viene segnalata; altrimenti l'algoritmo va avanti all'infinito.

3. E' decidibile. Le combinazioni origine/destinazione per la mappa stradale dell'Italia del 30/6/2009 sono finite. Siccome è dato che il programma in ingresso termina sempre, è sufficiente provare tutte le combinazioni per arrivare alla risposta desiderata.

Esercizio 3

1. FALSA. Se il problema è risolto senza fare uso della memoria (per esempio un problema di riconoscimento di un linguaggio regolare, che può essere risolto mediante un FSA), sia la MT che la macchina RAM lo possono risolvere in tempo $\Theta(n)$.

2. VERA. E' sempre possibile "peggiorare" arbitrariamente la complessità di un algoritmo. E' sufficiente, per esempio, scrivere una macchina RAM che, prima di risolvere un problema (per esempio di riconoscimento di $a^n b^n$) arbitrariamente esegue un calcolo con complessità esponenziale (anche se questo non serve al riconoscimento di $a^n b^n$).

3. VERA. Dal teorema di correlazione polinomiale sappiamo che, data una macchina RAM con costo (logaritmico) T_R, possiamo scrivere una MT a k nastri che la simula, con costo $\Theta(T_R^2)$. Inoltre, data una MT a k nastri con costo T_M, possiamo sempre scrivere una MT a nastro singolo che la simula, con costo $\Theta(T_M^2)$, in quanto ogni mossa della MT a k nastri richiede, per essere simulata dalla MT a nastro singolo, di scorrere tutto il nastro della MT a nastro singolo (che è, alla peggio, di lunghezza $\Theta(T_M)$).
In definitiva, è sempre possibile scrivere una MT a nastro singolo che simula la macchina RAM con costo temporale $\Theta(T_R^4)$.

Informatica Teorica

Appello d'esame - 16 Luglio 2009
Sezione Cremona+Como

Tempo a disposizione: 2h

Esercizio 1 (12 punti)

Si consideri il linguaggio

$L1 = \{ x \in \{a,b,c,d\}^* \mid \#a(x) - \#b(x) = \#c(x) - \#d(x) \}$

cioè l'insieme delle stringhe di alfabeto {a,b,c,d} il cui la differenza tra il numero delle a e delle b sia uguale alla differenza tra il numero delle c e delle d.

Si consideri ora il linguaggio

$$L2 = \{ x \in \{a,b,c,d\}^* \mid \exists k(|x|=2k) \wedge \forall y \in \{a,b,c,d\}^* (\exists z \in \{a,b,c,d\}^* (x=y \cdot z) \wedge \exists k(|y|=2k) \rightarrow \#a(y) - \#b(y) = \#c(y) - \#d(y)) \}$$

1. Descrivere a parole il linguaggio L2. In che relazione stanno i due linguaggi L1 ed L2?
2. Si scriva un automa riconoscitore a potenza minima che riconosce il linguaggio L1.
3. Si scriva un automa riconoscitore a potenza minima che riconosce il linguaggio L2.

Esercizio 2 (7 punti)

Si consideri l'insieme delle funzioni computabili, aventi come argomento e come valore un numero naturale, che sono definite solo per valori dispari dell'argomento (o, detto diversamente, che non sono definite per alcun argomento pari: si noti che tale insieme include anche la funzione ovunque indefinita).

1. Adottando l'usuale notazione impiegata nel corso, scrivere una formula logica del primo ordine, con una variabile libera *x*, che funga da predicato caratteristico di tale insieme, cioè la formula sia vera esattamente per i valori di *x* che corrispondono agli elementi dell'insieme.
2. Dire se l'insieme in questione è decidibile.
3. Dire se l'insieme è semidecidibile.

Esercizio 3 (14 punti)

1.
Descrivere a parole (ma in modo sufficientemente preciso per poterne valutare la complessità) una MT a nastro *singolo* che, data in input una sequenza di caratteri alfabetici minuscoli, trova il simbolo che è a metà esatta della sequenza (si supponga pure che la sequenza abbia un numero dispari di simboli).
Dare la complessità (sia temporale che spaziale) della MT così ideata.

2.
Descrivere a parole (ma in modo sufficientemente preciso per poterne valutare la complessità) una MT a nastro *singolo* che, data in input una sequenza di caratteri alfabetici minuscoli, la ordina in ordine crescente.
Quale è la complessità (sia temporale che spaziale) della MT descritta?

3.
Descrivere a parole (ma in modo sufficientemente preciso per poterne valutare la complessità) una MT a nastro *singolo* che, data in input una sequenza di numeri naturali codificati in binario (separati uno dall'altro dal simbolo #), la ordina in ordine crescente.
Quale è la complessità (sia temporale che spaziale) della MT descritta?

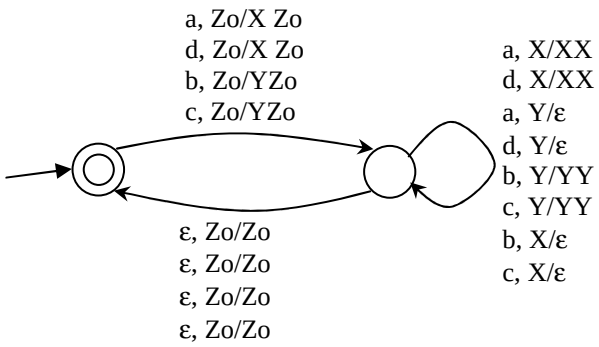
NB1: Come "parametri di dimensione dati" al punto 3 si usino il numero *n* di naturali presenti nella sequenza e il valore massimo *k* presente nella sequenza. Si supponga pure che ogni valore nella sequenza sia codificato con lo stesso numero di cifre binarie (corrispondente al numero di cifre necessario per codificare *k*).

NB2: Per tutti e tre i punti dell'esercizio, il punteggio massimo verrà assegnato se la MT ideata non usa altre celle del nastro oltre a quelle contenenti la stringa di input.

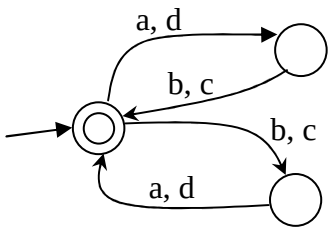
Soluzioni

Esercizio 1

1. L2 è l'insieme delle stringhe di lunghezza pari, nelle quali tutte le stringhe prefisso di lunghezza pari la differenza tra il numero delle a e delle b è uguale alla differenza tra il numero delle c e delle d. Quindi la proprietà, che in L1 è vera per tutta la stringa, in L2 è vera per tutti i suoi prefissi di lunghezza pari (inclusa la stringa intera stessa). Si noti che tutte le stringhe di L1 hanno lunghezza pari, così come quelle di L2. Quindi L2 è un sottoinsieme di L1
2. Se #a(x)-#b(x)=#c(x)-#d(x) allora #a(x)+#d(x)=#b(x)+#c(x), quindi durante la scansione della stringa occorre contare la differenza tra la somma delle a e delle d e la somma delle b e delle c. Tale differenza può crescere senza limiti con la lunghezza della stringa, quindi è necessario, e sufficiente, un automa a pila deterministico che memorizzi in forma unaria sulla pila il valore della differenza. Nella pila si usa il simbolo X per contare il surplus di a e d (cioè la pila contiene delle X quando, per la stringa y letta fino a quel momento, vale #a(y)+#d(y)>#b(y)+#c(y)) e il simbolo Y per contare un surplus di b e c. Di conseguenza, l'automa accetta una stringa x quando la pila (dopo aver letto tutta la stringa x) contiene solo il simbolo Z_0.



3. Nelle stringhe di L2 la differenza tra la somma delle a e delle d e la somma delle b e delle c in un qualsiasi prefisso deve avere valore assoluto <1, quindi è sufficiente un automa a stati finiti.



Esercizio 2

1. $\forall z(f_x(z)\neq\perp\rightarrow\exists k(z=2k+1))$
2. No, per il teorema di Rice.
3. No, perché è semidecidibile il complemento dell'insieme in questione, cioè quello delle funzioni che sono definite per qualche valore *pari* dell'argomento, cosa che si può dimostrare con l'usuale argomento "diagonale": si eseguono a turno le funzioni per un numero crescente di passi, sugli argomenti pari.

Esercizio 3

1.

Il modo più efficiente per trovare il punto di mezzo di una sequenza mediante una MT a nastro singolo si basa sull'idea di "marcare" la prima e l'ultima lettera della sequenza (per esempio cambiando a in a', oppure b in b', ecc.), poi scorrere la sequenza avanti ed indietro, spostando i "marker" via via un simbolo avanti ed un simbolo indietro, fino a che essi non si sovrappongono.

La complessità temporale di una simile MT è $\Theta(n^2)$, mentre la complessità spaziale è $\Theta(n)$ (in effetti è *esattamente* n, che è la minima complessità spaziale possibile per MT a nastro singolo, in quanto il nastro di memoria contiene almeno la stringa di input).

2.

Una maniera efficiente di ordinare la sequenza desiderata è mediante un algoritmo di bubblesort, in cui elementi contigui e disordinati della sequenza (tali cioè il simbolo a sinistra è maggiore di quello a destra) sono scambiati tra loro fino a che la sequenza è ordinata. Più precisamente, la stringa viene scorsa da sinistra a destra, ed ogni volta che si trova un simbolo che è maggiore del simbolo immediatamente successivo, essi vengono scambiati. Inoltre, se viene fatto uno scambio di simboli, la MT tiene traccia nei suoi stati del fatto di avere effettuato uno scambio. Quando arriva in fondo alla stringa, se uno scambio è stato effettuato, la MT torna all'inizio del nastro e ricomincia a scorrerlo. La computazione termina quando, arrivando in fondo al nastro, non sono stati effettuati scambi di simboli.

La complessità temporale di una tale MT è $\Theta(n^2)$, in quanto il meccanismo è lo stesso del bubblesort classico studiato in corsi di algoritmi, mentre la complessità spaziale è $\Theta(n)$. In effetti, in questo caso la complessità spaziale è *esattamente* n, in quanto non serve memorizzare nulla sul nastro della MT a parte la stringa in ingresso.

Si noti come l'algoritmo di COUNTING-SORT che, nel caso di esecuzione su MT a k nastri o nel caso di macchina RAM sarebbe più efficiente (lineare per MT a k nastri, $\Theta(n \log(n))$ per macchina RAM a costo logaritmico), nel caso di MT a nastro singolo avrebbe comunque complessità quadratica (ed in compenso richiederebbe celle aggiuntive oltre a quelle necessarie per memorizzare la stringa in ingresso).

3.

Anche in questo caso un algoritmo efficiente è quello di bubblesort, con lo stesso meccanismo descritto al punto 2. Tuttavia, in questo caso il costo di confrontare e scambiare numeri contigui non è costante, ma è, $\log(k)^2$, in quanto richiede, alla peggio, di spostare di $\log(k)$ elementi la testina avanti e indietro per $\log(k)$ volte.

Quindi, la complessità temporale dell'algoritmo è $\Theta(n^2 \log(k)^2)$. La complessità spaziale è invece $\Theta(n \log(k))$, in quanto non è necessario usare altra memoria oltre a quella che serve per memorizzare la sequenza di dati di input (si noti che $\Theta(n \log(k))$ è in effetti la lunghezza della stringa in input alla MT).

Informatica Teorica

Appello d'esame – 3 Settembre 2009
Sezione Cremona+Como

Tempo a disposizione: 2h

Esercizio 1 (14 punti)
Si consideri il linguaggio

$L = \{ x \in \{a,b,c\}^* \mid \exists y,z \in \{a,b,c\}^*, w \in \{a,b,c\}^+ (x = y \cdot w \cdot z, \wedge |y| = |z| \wedge w = w^R) \}$

cioè l'insieme delle stringhe di alfabeto {a,b,c} che al centro hanno un palindromo (cioè una stringa che è uguale se letta da destra a sinistra o da sinistra a destra).

1. Si scriva una grammatica con il minor numero di nonterminali possibile che genera il linguaggio L.
2. La grammatica scritta al punto 1 è a potenza minima tra quelle che generano L?
3. Si scriva un automa riconoscitore a potenza minima che riconosce il linguaggio L.

Esercizio 2 (9 punti)
Siano dati i seguenti predicati che rappresentano le proprietà di una coda di persone.

- `davanti(x,y)` è un predicato che è vero se la persona `x` è in coda immediatamente prima della persona `y` (cioè se `x` è davanti ad `y`, ed in mezzo non c'e' nessuno).
- `istanteDiArrivo(x)` è una funzione che restituisce il valore dell'istante `t` in cui la persona `x` è arrivata in coda.

Formalizzare mediante formule logiche, usando *esclusivamente* i predicati elencati sopra, le seguenti proprietà:

- La coda è ben ordinata, cioè le persone sono in fila indiana, una dietro l'altra, senza biforcazioni.
- In testa alla coda c'è una sola persona.
- In coda ci sono due persone, separate da al più un'altra persona, che sono arrivate a distanza di non più di T istanti di tempo una dall'altra.

NB: Proprietà come il fatto che una persona non può essere davanti a se stessa, e che se `x` è davanti ad `y`, allora `y` non può essere davanti ad `x`, non devono essere formalizzate, e possono essere date per assodate.

Esercizio 3 (9 punti)
Si dica, motivando adeguatamente la risposta, se le seguenti affermazioni sono vere o false.

1. Fissata una funzione f(x), il problema di stabilire se la i-esima macchina di Turing calcola una funzione diversa da f(x) non è decidibile.
2. Fissata una funzione f(x) calcolabile, il problema di stabilire se la i-esima macchina di Turing calcola una funzione diversa da f(x) non è decidibile.
3. Fissate due funzioni calcolabili $f_1(x)$ ed $f_2(x)$, il problema di stabilire se la i-esima macchina di Turing calcola una funzione totale a valori pari (cioè il cui risultato è sempre un valore pari) uguale a $2(f_1(x) + f_2(x))$ non è decidibile.

Soluzioni

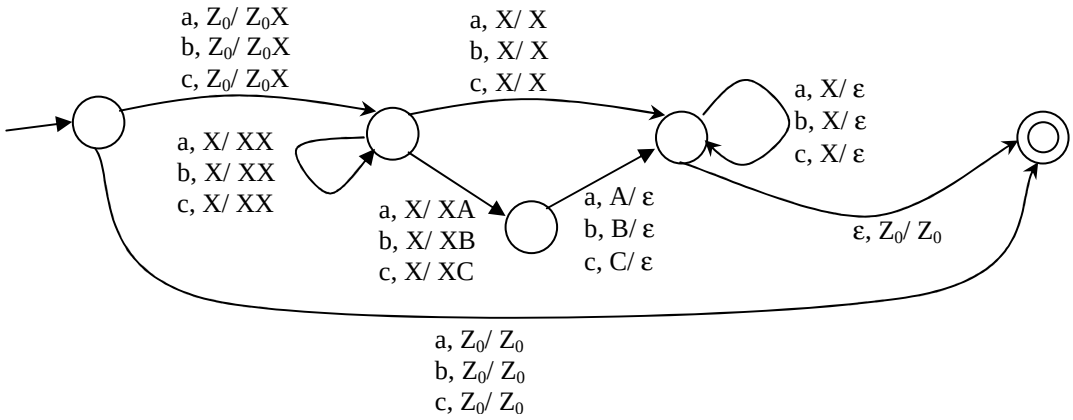
Esercizio 1

Si noti che, se la lunghezza della stringa x è dispari, x appartiene sempre al linguaggio L, in quanto è sufficiente che *esista* una sottostringa di lunghezza maggiore o uguale ad 1 al centro di x che è un palindromo, ed una stringa di lunghezza 1 lo è banalmente.
Se invece x è una stringa di lunghezza pari, condizione *necessaria e sufficiente* perché essa appartenga ad L è che i due caratteri centrali siano uguali.
In conseguenza di queste considerazioni, la seguente grammatica, che fa uso del solo nonterminale S, genera il linguaggio L desiderato.

$S \rightarrow aSa \mid aSb \mid aSc \mid bSa \mid bSb \mid bSc \mid cSa \mid cSb \mid cSc \mid a \mid b \mid c \mid aa \mid bb \mid cc$

La grammatica scritta (che è nonceontestuale) è anche a potenza generativa minima tra quelle che generano L. Infatti, per riconoscere il linguaggio L è necessario identificare, se la stringa è di lunghezza pari, quali sono i 2 simboli al centro della stringa, e per questo è necessario poter contare il numero di simboli prima e dopo i caratteri centrali, il che richiede una pila. Sfruttando il parallelo tra automi e grammatiche, se per riconoscere L occorre un automa a pila (nondeterministico), per generarlo serve una grammatica noncontestuale.
Si noti che, nel caso di stringhe li lunghezza dispari, non serve contare esattamente quanti caratteri ci sono nella stringa, ma basta assicurarsi che questi siano in numero dispari appunto, cosa per la quale basta un automa a stati finiti.

Per riconoscere il linguaggio L, come detto sopra, serve come minimo un automa a pila, come per esempio il seguente.



Esercizio 2

$\forall x,y,z(davanti(x,y) \wedge davanti(x,z) \rightarrow y = z)$
$\forall x,z\ (\neg\exists y(davanti(y,x)) \wedge \neg\exists y'(davanti(y',z)) \rightarrow x = z)$
$\exists x,y((davanti(x,y) \vee \exists z(davanti(x,z) \wedge davanti(z,y))) \wedge istanteDiArrivo(y) - istanteDiArrivo(x) \leq T)$

Esercizio 3

1. FALSO. La funzione f(x) potrebbe non essere calcolabile; in questo caso, una qualunque macchina di Turing computerebbe una funzione diversa da f(x), quindi l'insieme di indici di

macchine di Turing corrispondente sarebbe l'insieme universo, che per il teorema di Rice è decidibile.

2. VERO. Il problema in oggetto è il complemento del problema di stabilire se, fissata una funzione f(x) calcolabile, è decidibile l'insieme di indici di macchine di Turing che calcolano f(x). Tale problema è notoriamente indecidibile, quindi anche il suo complemento è indecidibile.

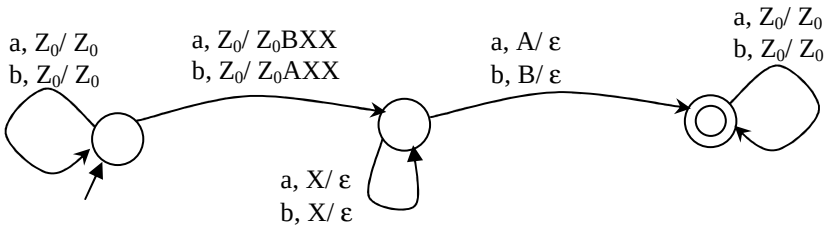
3. FALSO. Se una tra $f_1(x)$ ed $f_2(x)$ non è totale, neanche $2(f_1(x) + f_2(x))$ lo è. In questo caso, l'insieme degli indici di macchine di Turing che calcolano funzioni totali, a valori pari, ed uguali a $2(f_1(x) + f_2(x))$ è l'insieme vuoto che, per il teorema di Rice, è decidibile.

Informatica Teorica

Appello d'esame – 1 Febbraio 2010
Sezione Cremona+Como

Tempo a disposizione: 2h

Esercizio 1 (15 punti)
Si consideri il seguente automa a pila P1 (in cui la convenzione usata per descrivere la pila nelle transizioni è sinistra-basso e destra-alto):



1. Si scrivano 2 stringhe accettate dall'automa P1 (di cui una lunga almeno 6 caratteri) e 2 stringhe che invece P1 non accetta (di cui una lunga almeno 5 caratteri).
2. Descrivere a parole il linguaggio accettato dall'automa a pila P1.
3. E' possibile scrivere una macchina di Turing che accetta lo stesso linguaggio dell'automa a pila P1?
4. E' possibile scrivere un automa a stati finiti che accetta lo stesso linguaggio dell'automa a pila P1? Se no, giustificare la risposta. Se sì, disegnarlo.
5. Scrivere una formula logica che definisce il linguaggio $\neg L_{P1}$, corrispondente al complemento del linguaggio accettato dall'automa P1.

Esercizio 2 (9 punti)
Si risponda alle seguenti domande, motivando opportunamente le risposte.

1. E' decidibile il problema di stabilire se la seguente grammatica G1
G1: S → aSa | aSb | aSc | bSa | bSb | bSc | cSa | cSb | cSc | a | b | c | aa | bb | cc
genera lo stesso linguaggio dell'automa P1 dell'esercizio 1?

2. E' decidibile il problema di stabilire se, dato un generico automa a stati finiti, esso accetta lo stesso linguaggio dell'automa a pila P1?

3. E' decidibile il problema di stabilire se, data una generica macchina di Turing, essa accetta lo stesso linguaggio dell'automa a pila P1?

Esercizio 3 (10 punti)
Si descriva un algoritmo per la macchina RAM che calcoli la seguente funzione f: $\mathbb{N} \rightarrow \mathbb{N}$

f(x) = **if** x pari **then** $\sqrt[4]{\left(x^x\right)^x}$ **else** x^x

e se ne valuti l'ordine di grandezza sia della complessità temporale che della complessità spaziale sia a criterio di costo costante che a criterio di costo logaritmico.

NB1: L'esercizio verrà valutato tanto meglio quanto migliore sarà la complessità *temporale* dell'algoritmo definito.
NB2: Non è necessario codificare in dettaglio l'algoritmo: è sufficiente descriverlo (ad esempio mediante pseudolinguaggio di alto livello di tipo C-like) a un livello di precisione sufficiente da permettere la valutazione corretta e precisa della sua complessità (a meno della relazione Θ).

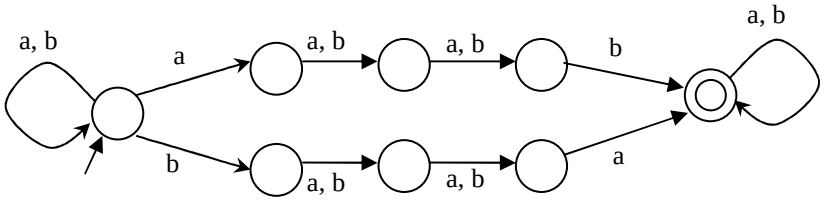
Soluzioni

Esercizio 1
1. Due stringhe accettate sono: abbbbb, bbabbaaa
Due stringhe non accettate, invece, sono: abbabb e bbbbbbbbb

2. L'automa accetta tutte e sole le stringhe (sull'alfabeto {a,b}) di almeno 4 caratteri in cui ci sono almeno 2 caratteri separati tra loro da altri 2 caratteri e diversi tra loro (o, se si preferisce, in cui c'è una sottosequenza di 4 caratteri in cui il primo e l'ultimo sono diversi tra loro).

3. Naturalmente sì, visto che gli automi a pila sono un caso particolare delle macchine di Turing (dato un qualunque automa a pila, è sempre possibile scrivere una macchina di Turing, per esempio ad un nastro, che si comporta esattamente come l'automa a pila usando il nastro a mo' di pila).

4. Sì:



5. $x \in \neg L_{P1} \leftrightarrow$
$x \in \{a,b\}^* \wedge \forall w \in \{a,b\}^4 \, (\exists y,z \in \{a,b\}^* \, (x = y \cdot w \cdot z) \rightarrow \exists l \in \{a,b\}^2 \, (w = a \cdot l \cdot a \vee w = b \cdot l \cdot b))$

In alternativa, in modo equivalente, sfruttando in modo più letterale la definizione del linguaggio L_{P1} data al punto 2:
$x \in \neg L_{P1} \leftrightarrow x \in \{a,b\}^* \wedge \neg \exists l,y,z \, (l \in \{a,b\}^2 \wedge y,z \in \{a,b\}^* \wedge (x = y \cdot a \cdot l \cdot b \cdot z \vee x = y \cdot b \cdot l \cdot a \cdot z))$

Esercizio 2
1. Il problema è banalmente decidibile in quanto la risposta è chiusa, o "sì", o "no".
In effetti, in questo caso il problema è anche deciso, osservando che la grammatica genera il linguaggio L dell'esercizio 1 del tema d'esame del 3/9/2010, e cioè l'insieme delle stringhe di alfabeto {a,b,c} che al centro hanno un palindromo.

2. Il problema è decidibile in quanto l'automa P1 riconosce un linguaggio riconoscibile tramite automi a stati finiti. Siccome tutte le operazioni sugli automi a stati finiti (intersezione, complemento, linguaggio vuoto, ecc.), è facile scrivere una macchina di Turing che prende in ingresso un automa a stati finiti F, e determina l'equivalenza o meno del linguaggio riconosciuto da questo con L_{P1} (sfruttando per esempio le operazioni sugli automi per confrontare F con l'automa del punto 4 dell'esercizio 1.

Anche senza sfruttare il fatto che il linguaggio L_{P1} è riconoscibile da un automa a stati finiti, si può comunque determinare che il problema è decidibile osservando che, se L_{P1} non fosse riconoscibile da un FSA, allora il problema sarebbe banalmente deciso (nessun FSA accetta L_{P1}); in caso contrario, valgono le considerazioni fatte sopra sulla decidibilità delle operazioni tra FSA.

3. In questo caso il problema non è decidibile, in quanto analogo al problema di stabilire la "correttezza" di una macchina di Turing.

Esercizio 3

Il problema è analogo a quello dell'esercizio 3.1.46 dell'eserciziario (II ed.).

Sfruttando le proprietà delle potenze, ed in particolar modo il fatto che $\sqrt[4]{\left(x^x\right)^x} = \sqrt[4]{x^{x^2}} = x^{\left(\frac{x}{2}\right)^2}$ un semplice algoritmo che calcola la funzione richiesta è

```
if x è pari then
  m := (x/2)*(x/2)
else
  m := x
ris := 1
for i := 1 to m do ris := ris*x
```

Calcolata a criterio di costo costante, la complessità temporale dell'algoritmo è $\Theta(x^2)$, mentre la complessità spaziale è $\Theta(1)$.
Calcolata a criterio di costo logaritmico, la complessità temporale dell'algoritmo è $\Theta(x^2 \log(x^{(x/2)^2}))$, cioè $\Theta(x^4 \log(x))$. La complessità spaziale invece è $\Theta(\log(x^{(x/2)^2}))$, cioè $\Theta(x^2 \log(x))$.

Un algoritmo più efficiente è però il seguente:

```
if x è pari then
  m := (x/2)*(x/2)
else
  m := x
ris := 1; xi := x;
while m > 0 do begin
  if m mod 2 = 1 then
    ris := ris* xi; xi := xi*xi; m := m div 2
end;
```

In questo caso, calcolata a criterio di costo costante, la complessità temporale dell'algoritmo è $\Theta(\log(x^2))$, cioè $\Theta(\log(x))$, in quanto il ciclo viene eseguito solo log(m) volte (m viene diviso per 2 ad ogni iterazione), ed m nel caso peggiore vale $(x/2)^2$; la complessità spaziale invece è ancora $\Theta(1)$.

Calcolata a criterio di costo logaritmico, la complessità temporale dell'algoritmo è $\Theta(\log(x) \log(x^{(x/2)^2}))$, cioè $\Theta(x^2 \log^2(x))$. La complessità spaziale invece rimane $\Theta(x^2 \log(x))$.