Informatica Teorica

Prima prova in itinere - 8 Maggio 2008

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)

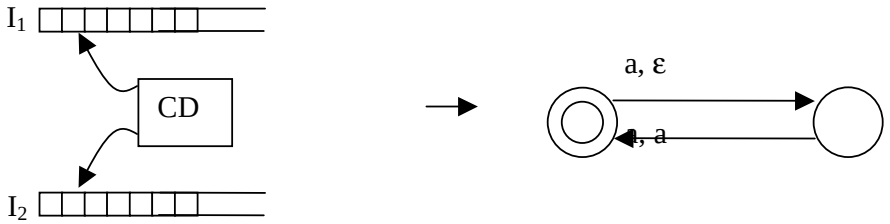
Parte 1

Si formalizzi la nozione di automa trasduttore finito deterministico FTε1 che, al contrario del modello di trasduttore finito presentato a lezione (chiamiamolo FT), può anche effettuare ε-mosse ma può emettere, a ogni transizione, al più un solo simbolo sul nastro di uscita.

Dimostrare che il modello FTε1 è almeno altrettanto potente del modello FT deterministico (ossia che qualsiasi FT può essere simulato da un opportuno FTε1).

Parte 2

Formalizzare poi la nozione di automa finito deterministico a due ingressi, 2FA, tratteggiato in figura a sinistra.



2FA può fare ε-mosse, non facendo avanzare la testina di ingresso su uno o entrambi i nastri di ingresso I_1 e I_2. L'automa 2FA può essere visto come traduttore nel modo seguente: la stringa di ingresso x viene accettata e tradotta in $\tau(x)$ se e solo se la coppia di stringhe $<x, \tau(x)>$ sui due nastri I_1 e I_2 è accettata. Per esempio, l'automa 2FA nella figura a destra definisce la traduzione $\tau(a^n)=a^{n/2}$ per $n\geq 0$ e pari. Mostrare che il modello 2FA è almeno altrettanto potente di FT deterministico, nel senso che se un FT deterministico accetta una stringa x e la traduce nella stringa $\tau(x)$, allora esiste un opportuno 2FA che accetta la coppia di stringhe $<x, \tau(x)>$ sui due nastri I_1 e I_2.

Valutare se il modello 2FA ha *esattamente* la stessa potenza espressiva di FT, cioè se qualsiasi traduzione definita da un 2FA nel modo sopra indicato può essere calcolata da un opportuno FT.

Esercizio 2 (4/13 punti)

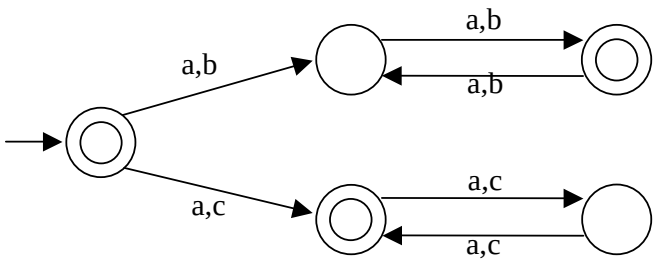
Scrivere una grammatica G che generi il linguaggio $\{a^nb^n$ se n è pari $>= 0$, a^nc^n se n è dispari$\}$.

Soluzioni

Esercizio 1

- Un FTε1 è una tupla (Q, I, O, δ, q_0, F) dove Q, I, O, q_0, e F hanno lo stesso significato che negli automi trasduttori tradizionali. La funzione di transizione δ: $Q \times I \cup \{\varepsilon\} \rightarrow Q \times O \cup \{\varepsilon\}$ definisce, per ogni coppia (q, i) di stato corrente e carattere in input (oppure ε) lo stato prossimo q' e il singolo carattere scritto in output (oppure ε), cioè δ(q,i) = (q', o). Affinchè sia deterministico si richiede che se δ(q,ε) è definito, allora δ(q,i) è indefinito per tutti gli $i \in I$.
  Conviene poi fornire la definizione di configurazione, transizione tra configurazioni, e trasduzione seguendo la prassi standard usata per automi a pila e macchine di Turing.
- Dato un FT = (Q, I, O, δ, q_0, F) deterministico, è sempre possibile definite un FTε1 = (Q', I, O, δ', q_0, F) che definisce la stessa traduzione facendo in modo che, quando FT emette una stringa di lunghezza >1, l'FTε1 corrispondente emetta in sequenza i caratteri della stringa, uno alla volta, mediante una serie di ε-mosse che lo fanno passare attraverso una serie di stati aggiuntivi, diversi da tutti gli altri, introdotti a questo scopo. Q' e δ' sono quindi definiti in questo modo. Q' contiene tutti gli stati che sono in Q. Inoltre, per ogni δ(q,i) = (q', s) con $s \in O^*$, se $|s| \leq 1$ si ha semplicemente δ'(q,i) = (q', s); se $|s| > 1$si aggiungono |s| stati q^s_1, q^s_2, ..., $q^s_{|s|}$ a Q' e si pone: (1) δ'(q,i) = (q^s_1, s[1]); (2) per ogni $2 \leq k \leq |s|$: δ'(q^s_{k-1},ε) = (q^s_k, s[k]); (3) δ'($q^s_{|s|}$,ε) = (q', ε);.
- Un 2FA è una tupla (Q, I_1, I_2, δ, q_0, F) dove I1 e I2 sono gli alfabeti dei due nastri di ingresso, Q, q_0, e F hanno il significato convenzionale. La funzione di transizione δ: $Q \times I_1 \cup \{\varepsilon\} \times I_2 \cup \{\varepsilon\} \rightarrow Q$ definisce per ogni tupla (q, x, y) di stato corrente q, carattere sul primo nastro x (o ε), e carattere sul secondo nastro y (o ε) lo stato prossimo δ(q, x, y). Si richiede per il determinismo che se δ(q, x, ε) (rispettivamente δ(q, ε, y)) è definito allora δ(q, x, y) è indefinito per ogni $y \in I_2$ (rispettivamente per ogni $x \in I_1$).
  La definizione di configurazione e transizione tra configurazioni sono definite di conseguenza nel solito modo (includendo nella configurazione il contenuto dei due nastri dalla posizione della testina in poi).
- Il modello 2FA è almeno altrettanto potente del modello FTε1 (e quindi, transitivamente, del modello FT): dato un qualsiasi FTε1, esiste un 2FA che simula ogni sua transizione, leggendo dal secondo nastro di ingresso lo stesso carattere che l'FTε1 emette sul nastro di uscita ed effettuando ε-mosse sul primo o sul secondo nastro di ingresso quando, rispettivamente, l'FTε1 fa una ε-mossa (i.e., non consuma ingresso) o non emette alcun simbolo in uscita. Formalmente, dato un FTε1 = (Q, I, O, δ, q_0, F) si definisce il 2FA = (Q, I, O, δ', q_0, F) dove δ'(q, x, y) = q' se e solo se δ(q, x)=(q',y), $\forall q,q' \in Q$ e $\forall x \in I \cup \{\varepsilon\}$ e $\forall y \in O \cup \{\varepsilon\}$.
- Il modello 2FA è strettamente più potente del modello FT, perchè permette di definire una traduzione, accettandola come stringa in ingresso sul secondo nastro, anche nel caso in cui la traduzione dipenda da una proprietà della stringa di ingresso (quella sul primo nastro) che può essere decisa solo al termine della scansione, dopo un numero di passi illimitato a priori. Ciò non può essere fatto dal modello FT, a causa del vincolo di memoria finita che lo obbliga a emettere la traduzione "in tempo reale", durante la scansione della stringa in ingresso. Come esempio di ciò si consideri la seguente traduzione: $\tau(a^n)=b^n$ se n è pari, $\tau(a^n)=c^n$ se n è dispari. Nessun FT può calcolare questa traduzione, perchè il riconoscimento della stringa come avente lunghezza pari o dispari avviene solo al termine della scansione, mentre l'emissione della traduzione deve avvenire durante la scansione. Il 2FA mostrato in figura invece *definisce* la traduzione in questione accettando il linguaggio $L=\{(a^n,b^n) \mid n \geq 0, n \text{ pari } \} \cup \{(a^n,c^n) \mid n \text{ dispari } \}$

.



Esercizio 2

S → X | Y
X → aAb | ε
A → aXb
Y → aBc
B → aYc | ε

Informatica Teorica

Seconda prova in itinere – 30 Giugno 2008, Sezione Cremona+Como

Esercizio 1 (punti 8/17-esimi)

Si formalizzi mediante formule del prim'ordine il seguente comportamento di un sistema di allarme a tempo continuo.

- L'allarme si attiva e disattiva inserendo la chiave nell'apposito alloggiamento; mentre la disattivazione è immediata, l'attivazione avviene dopo 12 secondi dall'introduzione della chiave; un eventuale reinserimento della chiave durante questo periodo non ha effetto; l'estrazione della chiave dall'alloggiamento non ha alcun effetto e può essere trascurata.
- Ad allarme attivato l'ingresso di un corpo estraneo nel volume sotto controllo determina lo scatto dell'allarme dopo 12 secondi, a meno che durante tale intervallo non venga inserita la chiave nell'alloggiamento per disinserirlo.

Si suggerisce di usare i seguenti predicati logici (dall'ovvio significato), tutti quanti aventi un parametro a valori reali: *attiva_allarme*(*t*), *disattiva_allarme*(*t*), *allarme_attivo*(*t*), *inserimento_chiave*(*t*), *rilevamento_corpo_estraneo*(*t*), *scatto_allarme*(*t*).

Esercizio 2 (punti 6/17-esimi)

Si dica, giustificando brevemente la risposta, se i seguenti problemi sono decidibili o semidecidibili:

1. Stabilire se, data una generica macchina di Turing a nastro singolo e due configurazioni della medesima, le due configurazioni sono tra loro in relazione di transizione immediata.
2. Stabilire se, data una generica macchina di Turing a nastro singolo e due configurazioni della medesima, le due configurazioni sono tra loro in relazione di transizione non necessariamente immediata.

Esercizio 3 (punti 6/17-esimi)

1. Si supponga che una macchina di Turing deterministica simuli il comportamento di un automa a stati finiti nondeterministico nel seguente modo: ricevendo in ingresso una stringa, essa riproduce, una dopo l'altra, tutte le possibili esecuzioni che l'automa nondeterministico può svolgere su tale stringa. Si indichi la complessità minima (col criterio del caso pessimo) che tale simulazione richiede, sia per il tempo che per lo spazio, in funzione della lunghezza *n* della stringa in ingresso *x*.
2. Si supponga che una macchina di Turing deterministica abbia in ingresso la descrizione di un automa a stati finiti nondeterministico, e una stringa, da intendere come ingresso per l'automa, e scriva in uscita il valore 1 se la stringa appartiene al linguaggio accettato dall'automa, 0 altrimenti. Si valutino, giustificando brevemente la risposta:
    a. la complessità temporale e quella spaziale minime (col criterio del caso pessimo) come funzione del numero *m* degli stati dell'automa in ingresso;
    b. la complessità temporale e quella spaziale minime (sempre col criterio del caso pessimo) come funzione della lunghezza *n* della stringa in ingresso *x*.

Soluzioni schematiche

Esercizio 1

inserimento_chiave, *attiva_allarme*, *disattiva_allarme*, *rilevamento_corpo_estraneo*, *scatto_allarme* sono definiti a priori come eventi, ossia istantanei, *allarme_attivo* è invece definito a priori come uno stato che ha quindi una durata.

La formula seguente esprime i requisiti richiesti mediante la sintassi del calcolo dei predicati.

$$\forall t\left(\begin{array}{l} attiva\_allarme(t) \leftrightarrow \\ \quad inserimento\_chiave(t-12) \wedge \neg allarme\_attivo(t-12) \wedge \\ \quad \forall t_1(t-12 < t_1 < t \rightarrow \neg attiva\_allarme(t_1)) \end{array}\right)$$

$$\wedge$$

$$\forall t\left(disattiva\_allarme(t) \leftrightarrow inserimento\_chiave(t) \wedge allarme\_attivo(t)\right)$$

$$\wedge$$

$$\forall t\left(\begin{array}{l} allarme\_attivo(t) \leftrightarrow \\ \quad \exists t_1\left(\begin{array}{l} t_1 < t \wedge attiva\_allarme(t_1) \wedge \\ \forall t_2(t_1 < t_2 < t \rightarrow \neg disattiva\_allarme(t_2)) \end{array}\right) \end{array}\right)$$

$$\wedge$$

$$\forall t\left(\begin{array}{l} scatto\_allarme(t) \leftrightarrow \\ \quad rilevamento\_corpo\_estraneo(t-12) \wedge allarme\_attivo(t-12) \wedge \\ \quad \forall t_1(t-12 < t_1 < t \rightarrow \neg inserimento\_chiave(t_1)) \end{array}\right)$$

Esercizio 2

Quesito 1

Il problema è sicuramente decidibile: è infatti immediato costruire un semplice algoritmo che scandisca la configurazione iniziale e determini il simbolo in lettura e lo stato della macchina, individui la mossa definita dalla funzione δ della macchina e verifichi se la seconda configurazione coincide con quella che si otterrebbe applicando la δ.

Quesito 2

Questo problema è invece non decidibile. Infatti il classico problema dell'HALT è un caso particolare del problema posto: stabilire cioè se da una configurazione iniziale si può giungere a una configurazione di HALT. NB: per la precisione, visto che in generale vi possono essere diverse di configurazioni di HALT, per ricondurre il problema dell'HALT esattamente al problema in questione è necessario modificare la macchina in modo tale che essa abbia un'unica configurazione di HALT (ad esempio, nastro vuoto e stato q_F.)

Il problema è invece semidecidibile in quanto è sufficiente "far girare" la macchina a partire dalla prima configurazione: se la seconda è raggiungibile, prima o poi la si raggiunge e si risolve il problema. Non vale ovviamente il viceversa.

Esercizio 3

Quesito 1

Il meccanismo di simulazione della MT è quello classico di visita dell'albero delle computazioni; quindi la complessità temporale è nel caso pessimo $\Theta(2^n)$ (per semplicità assumiamo che l'albero sia binario). Se la macchina costruisce l'intero albero delle computazioni anche la complessità spaziale è dello stesso ordine. Tuttavia la macchina potrebbe limitarsi a memorizzare solo l'ultimo cammino visitato, ad esempio, in preordine sinistro; in tal modo potrebbe generarli ed esaminarli tutti in sequenza (ad esempio:

S, S, S, S, S -→ S,S,S,S,D → S, S, S, D, S → S, S, S, D, D → S, S, D, S, S ….)

senza costruire l'intero albero. In tal caso la complessità spaziale risulta $\Theta(n)$.

Quesito 2

Consideriamo una macchina di Turing che costruisce l'automa deterministico equivalente a quello non deterministico di partenza, e quindi effettua il riconoscimento della stringa simulando direttamente l'automa deterministico.

a.

E' noto che nel caso pessimo l'automa deterministico, equivalente a quello in input di m stati, ha $\Theta(2^m)$ stati e quindi la complessità spaziale della macchina è dominata da questo fattore. Allo stesso modo, la complessità temporale è dominata dalla costruzione di tale automa.

b.

Una volta costruito l'automa deterministico la MT può semplicemente simulare quest'ultimo con la stessa sua complessità, ossia $\Theta(n)$ per la complessità temporale e $\Theta(1)$ per quella spaziale, tenendo conto ovviamente che la memoria finita costituita dagli $\Theta(2^m)$ stati non è funzione della lunghezza della stringa di ingresso *x*.

Informatica Teorica

Appello d'esame del 16 luglio 2008, Sezione Cremona+Como

Si consideri la grammatica G seguente:

S → BAcB | ε
A → aAa | acBa
B → CcB | ε
C → aC | a

Esercizio 1 (punti 11)
Si fornisca un automa che riconosca il linguaggio L(G) generato da G.
L'automa deve appartenere alla classe di minima potenza riconoscitiva possibile per riconoscere L(G).
Si spieghi brevemente perché la classe dell'automa scritto è quella a potenza minima possibile.

Esercizio 2 (punti 12)

a. Si dica, giustificando brevemente la risposta, se è decidibile il problema di stabilire se L(G) intersecato con il linguaggio {(a*c)*} produce il linguaggio vuoto o no.

b. Si dica, giustificando brevemente la risposta, se è decidibile il problema di stabilire se, dato un generico automa a stati finiti A, il linguaggio L(A) riconosciuto da A è uguale al linguaggio L(G) oppure no.

c. Si dica, giustificando brevemente la risposta, se, ***fissata*** una grammatica G', è decidibile il problema di stabilire se, dato un generico automa a stati finiti A, il linguaggio L(A) riconosciuto da A è uguale al linguaggio L(G') generato da G' oppure no.

d. Si dica, giustificando brevemente la risposta, se è decidibile il problema di stabilire se, dati una ***generica*** grammatica G' ed un generico automa a stati finiti A, il linguaggio L(A) riconosciuto da A è uguale al linguaggio L(G') generato da G' oppure no.

Esercizio 3 (punti 10)
Si descrivano a grandi linee il funzionamento di una macchina di Turing deterministica e di una RAM che riconoscano L(G) e se ne valutino le complessità secondo la classe Θ, usando per la RAM il criterio di costo logaritmico.

Soluzioni schematiche

Esercizio 1

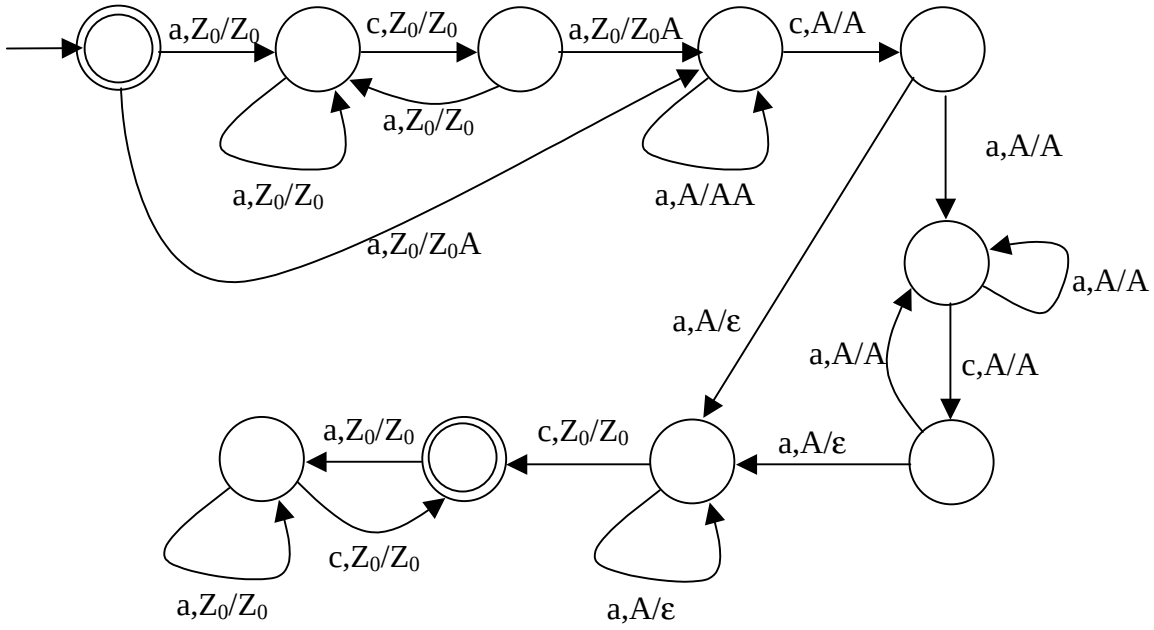
La grammatica genera il linguaggio:

$$L(G) = \{ (a^+c)^*a^nc(a^+c)^*a^nc(a^+c)^* \mid n > 0 \} \cup \varepsilon,$$

dove $x^+ = x^* - \{\varepsilon\}$.

Per riconoscere L(G) è necessario e sufficiente un automa a pila nondeterministico. La pila è necessaria per effettuare il conteggio delle sottostringhe a^n. Anche il nondeterminsmo è necessario perchè con il solo determinismo non è possibile stabilire in quale punto delle sottostringhe fatte di 'a' è necessario iniziare ad usare la pila per contare tali 'a'. In altre parole, il nondeterminismo serve a discernere le sottostringhe del tipo $(a^+c)^*$, per le quali è sufficientemente un riconoscimento a stati finiti senza pile, da quelle del tipo a^n per le quali la pila è invece ovviamente necessaria.

Un possibile ND PDA che riconosce L(G) è il seguente:



Esercizio 2

a. Decidibile, la risposta è chiusa (Si/No). Anzi, in questo caso il problema è deciso, in quanto l'intersezione non è vuota.
b. Decidibile: L(G) non è un linguaggio regolare, quindi la risposta sarà sempre No, qualunque sia A.
c. Decidibile: se L(G') non è un linguaggio regolare la risposta sarà sempre No; se invece L(G') è regolare, siccome è decidibile il problema di stabilire se FSA riconoscono lo stesso linguaggio, ci sarà sempre una MT che, fissato un linguaggio regolare, determina se questo è uguale a quello riconosciuto da un FSA in input. Ora, fissato L(G'), non è detto che io sappia esattamente quale è questa MT (i linguaggi regolari sono un'infinità enumerabile, e non è detto che io capisca quale di questi è quello generato da G'), ma essa di certo esiste, rendendo il problema decidibile.
d. Indecidibile, in quanto riconducibile al problema di stabilire se una generica MT calcola una certa funzione (in questo caso, quella corrispondente all'FSA), oppure no.

Esercizio 3

Una macchina di Turing (deterministica) che riconosce L(G) può operare nella seguente maniera (al solito indichiamo con n la lunghezza dell'input):

- In primo luogo effettua una scansione completa dell'input per verificare che sia una stringa del tipo $(a^+c)^k$ con k > 1. Questa è una condizione necessaria perchè sia in L(G), e può essere verificata in tempo $\Theta(n)$.
- Se il test precedente è passato con successo, l'input appartiene a L(G) se e solo se esistono due sottostringhe (massimali, ossia racchiuse tra due 'c') di sole 'a' della stessa lunghezza. Questa verifica può essere fatta nella seguente maniera: per ogni sottostringa di 'a' incontrata sequenzialmente nell'input:
  - la sottostringa di 'a' viene ricopiata su un nastro di memoria;
  - si prosegue la scansione dell'input confrontando ogni successiva sottostringa di 'a' con quella presente sul nastro di memoria.

  Nel caso pessimo questo comporta una scansione completa dell'input per ogni sottostringa di 'a' incontrata. Possono esservi fino a $\Theta(n)$ sottostringhe di questo tipo, quindi la complessità temporale di caso pessimo è $\Theta(n^2)$, che è anche la complessità complessiva dell'algoritmo. La complessità spaziale è invece chiaramente $\Theta(n)$.

Una macchina RAM può usare un algoritmo simile a quello descritto per la macchina di Turing, usando ad esempio dei contatori per memorizzare lunghezza delle sottostringhe di 'a' incontrate. Si avrebbero dunque $\Theta(n)$ contatori, ciascuno memorizzante un intero $\Theta(n)$. Adottando il criterio di costo logaritmico, i $\Theta(n^2)$ confronti da effettuare porterebbero pertanto a una complessità temporale complessiva di $\Theta(n^2 \log n)$. La complessità spaziale, secondo le stesse considerazioni, è un $O(n \log n)$.

Informatica Teorica

Appello d'esame dell'8 settembre 2008, Sezione Cremona+Como
Il tempo a disposizione è di 2 ore

Esercizio 1 (Punti 13)

Si considerino le seguenti regole relative al funzionamento di un apparecchio per l'uso del Bancomat:

1. L'apparecchio si trova inizialmente in uno stato di riposo
2. L'introduzione della tessera provoca la transizione in uno stato di immissione codice
3. Il codice consta di 5 cifre
4. L'immissione del codice è ultimata se l'utente ha digitato 5 cifre e successivamente premuto il tasto ENTER, oppure dopo che sono trascorsi 5 secondi dall'immissione della quinta cifra.
5. Una volta ultimata l'immissione del codice l'apparecchio passa in uno stato di verifica.
6. Se, nello stato di immissione, trascorrono più di 5 secondi senza che si digiti una cifra e non è stata ancora digitata la quinta cifra, viene espulsa la tessera e l'apparecchio torna nello stato di riposo.

Si formalizzino mediante formule del prim'ordine solo le regole 4, 5 e 6 (le altre servono da contesto ma non è necessario che vengano formalizzate esplicitamente). Si suggerisce di utilizzare i seguenti predicati che denotano possibili *stati* in cui l'apparecchio si trova in un generico istante t:

- Riposo(t)
- ImmissioneDati(t)
- Verifica(t)
- CifreImmesse(t, k) indica che all'istante t le cifre immesse sono k; lo stato ha senso solo durante lo stato ImmissioneDati.

e i seguenti predicati che denotano possibili *eventi* che si verificano in un generico istante t:

- PremiCifra(t)
- PremiEnter(t)
- EspulsioneTessera(t)

Esercizio 2 (Punti 11)

Dire se i seguenti problemi sono decidibili, solo semidecidibili, o né uno né l'altro:

1. Stabilire se, data una generica Macchina di Turing M, il linguaggio riconosciuto da M è vuoto.
2. Stabilire se, data una generica Macchina di Turing M che non ha cicli (cioè il cui grafo corrispondente è aciclico), il linguaggio riconosciuto da M è vuoto.
3. Stabilire se, data una generica Macchina di Turing M che non ha autoanelli (cioè il cui grafo corrispondente non ha archi che puntano al nodo da cui originano; un autoanello è un caso particolare di ciclo), il linguaggio riconosciuto da M **non è** vuoto.

Esercizio 3 (Punti 10)

Si consideri la traduzione di una stringa di ingresso $w \in \{a,b,c\}^+$ in una stringa $a^{n/2}b^m$, dove n è il numero di a nella stringa w e m è il numero di b nella stringa w.

1. Si descriva a parole una Macchina di Turing a k nastri che realizza la traduzione indicata, e se ne diano le complessità spaziale e temporale.
2. Si descriva a parole una Macchina RAM che realizza la traduzione indicata, e se ne diano le complessità spaziale e temporale usando il criterio di costo logaritmico.
3. Si descriva a parole una Macchina di Turing a nastro singolo che realizza la traduzione indicata, e se ne diano le complessità spaziale e temporale.

NB: Il punteggio conseguito sarà tanto più alto tanto migliori saranno le complessità delle macchine ideate.

Soluzioni

Esercizio 1

Come già in altre occasioni, per un generico stato S, Up_to_now_S(t) è una notazione abbreviata per la formula seguente

$$\exists\delta(\forall t_1(t-\delta < t_1 < t) \rightarrow S(t_1)) \qquad (\text{sottinteso}: \delta > 0)$$

Similmente From_now_on_S(t) è una notazione abbreviata per la formula seguente:

$$\exists\delta(\forall t_1(t \leq t_1 < t+\delta) \rightarrow S(t_1))$$

Ciò premesso le regole 4 e 6 possono essere formalizzate nel modo seguente, sottintendendo per ognuna la quantificazione universale ∀t, k:

Regole 4 e 5

(

Up_to_now_CifreImmesse(t, 5) ∧ PremiEnter(t)

→

From_now_on_¬ImmissioneDati(t) ∧ From_now_on_Verifica(t)

)

∧

(

Up_to_now_CifreImmesse(t, 5) ∧ PremiCifra(t-5) ∧

(∀t1 (t-5 < t1 < t) → (¬PremiEnter(t) ∧ ¬ PremiCifra(t)))

→

From_now_on_¬ImmissioneDati(t) ∧ From_now_on_Verifica(t)

)

Regola 6

Up_to_now_ImmissioneDati(t) ∧ Up_to_now_CifreImmesse(t, k) ∧ (0 ≤ k <5)

∧

(∀t1 (t-5 < t1 < t) → (¬PremiEnter(t) ∧ ¬ PremiCifra(t)))

→

From_now_on_¬ImmissioneDati(t) ∧ From_now_on_Riposo(t) ∧ EspulsioneTessera(t)

Esercizio 2

1.

Il problema non è né decidibile, né semidecidibile. Non è decidibile per il teorema di Rice (l'insieme delle MT che riconoscono il linguaggio vuoto non è l'insieme vuoto, né è l'insieme universo).

Non è neanche semidecidibile, in quanto è semidecidibile il suo complemento: enumerando opportunamente le stringhe in ingresso, con la solita tecnica diagonale è possibile simulare le esecuzioni della MT in modo che, se una stringa viene accettata, prima o poi questa viene trovata; in caso contrario, la computazione prosegue all'infinito.

2.

Il problema è decidibile (e quindi anche semidecidibile). Il numero di computazioni di una MT senza cicli è finito ed ognuna di esse è fatta di un numero finito di passi. E' quindi possibile enumerarle tutte, e determinare se almeno una di queste si porta in stato finale oppure no.

3.

Il problema non è decidibile, ma è semidecidibile. Data una MT qualunque (anche con autoanelli) è possibile costruirne in modo algoritmico una che sia invece priva di autoanelli (è sufficiente "sdoppiare" gli stati su cui ci sono autoanelli). Di conseguenza, è facile ridurre il problema della *emptiness* del linguaggio di una MT generica (cioè il determinare se il linguaggio accettato è vuoto o no) a quello di una MT senza autoanelli.

Se quindi fosse decidibile il problema in oggetto, lo sarebbe anche quello della *emptiness* di una MT generica (che è in effetti il complemento di quello in oggetto), che non è decidibile per quanto detto al punto 1.

Il problema è invece semidecidibile, in quanto, sempre per quanto detto al punto 1, per una generica MT (e quindi a maggior ragione per una senza autoanelli) è possibile, enumerando le stringhe in ingresso e le varie computazioni con meccanismo diagonale, determinare se questa accetta almeno una stringa (in caso contrario il procedimento algoritmico non termina).

Esercizio 3

1.

E' possibile scrivere una MT che effettua la traduzione desiderata senza fare uso di nastri di memoria, e in tempo $\Theta(n)$.

Tale macchina deve semplicemente scorrere il nastro di input 2 volte: una per produrre in uscita $a^{n/2}$, ed una per produrre in uscita b^m. Più precisamente, nella prima passata, la macchina produce, ogni 2 a incontrate, una a in uscita. Arrivata in fondo al nastro di input, essa lo scorre al contrario, producendo una b per ogni b letta.

2.

Una macchina RAM, non potendo scorrere al contrario il nastro di input, dovrà necessariamente ricorrere alla memoria per effettuare la traduzione desiderata.

Più precisamente, essa si può comportare nella seguente maniera: legge la stringa in ingresso e, per ogni a (risp. b) letta, incrementa un opportuno contatore memorizzato in M[1] (risp. M[2]). Alla fine della lettura divide M[1] per 2, quindi ripete M[1] volte la scrittura in uscita di a, ed M[2] volte la scrittura in uscita di b.

Le complessità (a criterio di costo logaritmico) di una simile macchina sono $\Theta(n \log(n))$ per quella temporale, e $\Theta(\log(n))$ per quella spaziale.

3.

Una Macchina di Turing a nastro singolo può innanzi tutto fare una passata sul nastro per dimezzare le a (per esempio scrivendo, al posto di ogni seconda a, una c); essa può quindi realizzare la traduzione desiderata con un meccanismo simile a quello dell'insertion sort per portare prima tutte le a in testa alla stringa, farle seguire dalle b, e quindi chiudere con le c.

Una volta fatto questo, è sufficiente cancellare le c in coda per ottenere la traduzione desiderata.

La complessità temporale di una simile macchina di Turing è, come per l'algoritmo di insertion sort, $\Theta(n^2)$; quella spaziale è $\Theta(n)$.

Informatica Teorica

Appello d'esame del 3 febbraio 2009, Sezione Cremona+Como

Il tempo a disposizione è di 1h30.

Esercizio 1 (punti 8)

Sia data la seguente grammatica G1:

$S \rightarrow AcS \mid bBS \mid bac$
$A \rightarrow ba$
$B \rightarrow ac$

1) Si dica quale è il linguaggio generato da G1.

2) Si scriva un automa che riconosce il linguaggio generato da G1. L'automa deve appartenere alla classe di potenza riconoscitiva minima tra quelle che riconoscono il linguaggio generato da G1.

Esercizio 2 (punti 12)

Si dica, giustificando brevemente la risposta, se è decidibile l'equivalenza tra:
1. Un automa a stati finiti e una macchina di Turing.
2. Una grammatica non contestuale e una macchina di Turing.
3. Una macchina di Turing e la grammatica G1 dell'esercizio 1.
4. Due automi a stati finiti.
5. Un automa a stati finiti e un automa a pila determinitico.
   (**Suggerimento**: si tenga presente che l'intersezione tra il linguaggio riconosciuto da un automa a stati finiti e quello riconosciuto da un automa a pila deterministico è riconoscibile da un automa a pila deterministico costruibile algoritmicamente sulla base dei due automi dati.)

Esercizio 3 (punti 11)
Si descriva a grandi linee il funzionamento di una macchina di Turing deterministica che riconosce il linguaggio $L = \{w^4 \mid w \in \{a, b, c\}+\}$ (laddove $w^4 = wwww$).
Si valutino le complessità spaziale e temporale della macchina di Turing descritta.

La macchina di Turing deve essere tale da minimizzare la complessità spaziale.

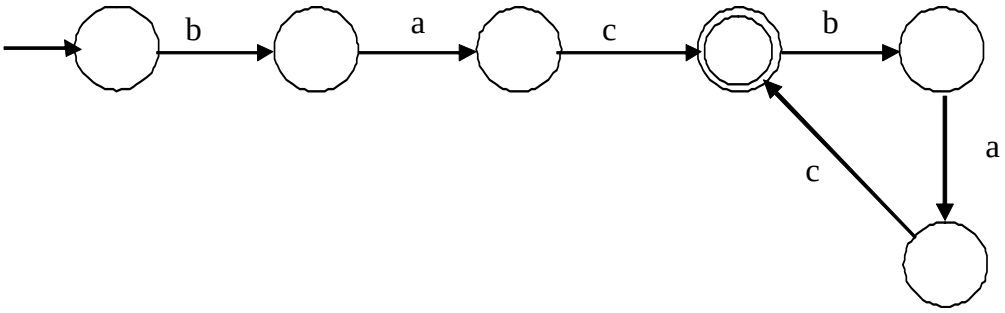
Soluzioni schematiche

Esercizio 1
La grammatica genera il linguaggio:

$L(G1) = \{ (bac)^+ \}$

L(G1) è un linguaggio regolare, cioè riconoscibile da un automa a stati finiti.

Un automa a stati finiti che riconosce il linguaggio è il seguente:



Esercizio 2
Le prime 3 domande hanno risposta negativa sulla base di normali ragionamenti in applicazione al teorema di Rice.
La domanda 4 ha risposta positiva, grazie alla chiusura del formalismo rispetto alle operazioni booleane: $L1 \equiv L2 \Leftrightarrow (L1 \cap \neg L2) = \varnothing \wedge (\neg L1 \cap L2) = \varnothing$.
Alla stessa maniera, sfruttando l'informazione fornita nel suggerimento e il fatto che sia i linguaggi regolari che quelli noncontestuali deterministici sono chiusi rispetto al complemento, si dimostra decidibile anche il quesito 5.

Esercizio 3
Una macchina di Turing (deterministica) a 2 nastri che riconosce il linguaggio desiderato può operare nella seguente maniera (al solito indichiamo con n la lunghezza dell'input):

1- conta gli elementi dell'input in binario (usando il nastro T1)
2- se il numero su T1 è divisibile per 4 (deve terminare con 2 zeri), allora elimina gli ultimi 2 zeri per dividere per 4
3- torna all'inizio dell'input
4- copia il contenuto di T1 nel nastro T2, tenendo traccia nello stato del simbolo letto
5- si sposta sul nastro di input decrementando T2 ad ogni mossa
6- quando T2 è a zero, confronta il simbolo sotto la testina con quello memorizzato, se sono diversi, si blocca
7- si sposta di una cella in avanti sul nastro di input; se è in fondo al nastro, stop con successo
8- torna indietro di T1 elementi sul nastro (usando ancora T2 come contatore)
9- ripete dal passo 4

La complessità spaziale della macchina è log(n).
La complessità temporale, invece, è $n^2 \log(n)$.