Informatica Teorica

PRIMO COMPITINO – 26 APRILE 2004

Si consideri il linguaggio L definito sull'alfabeto {a,b,c} e consistente di tutte e sole le stringhe appartenenti ad $\{a,b,c\}^*$ contenenti almeno un b e terminanti con 5 c consecutive.

Esercizio 1 (punti 3/15)
Si fornisca una formula del prim'ordine che caratterizzi tutte e sole le stringhe di L.

Esercizio 2 (punti 3/15)
Si descriva una macchina astratta che riconosca L. Si preferisce una macchina a "potenza minima" ovvero scelta tra la famiglia di automi meno potente tra quelli in grado di riconoscere L.

Esercizio 3 (punti 4/15)
Si costruisca una grammatica G che generi L. In questo caso si preferisce una grammatica con numero minimo di produzioni sintattiche, indipendentemente dalla classe di appartenenza e dal vocabolario nonterminale.

Esercizio 4.a (punti 5/15)
Si definisca in modo formale un automa a due code, informalmente descritto come segue:
L'automa agisce come riconoscitore di linguaggio operando su stringhe memorizzate su un unico nastro di ingresso; è dotato di un organo di controllo a stati finiti e di due ulteriori organi di memoria gestiti secondo la disciplina FIFO (code). Ogni mossa dell'automa consiste nei seguenti passi:

- lettura del carattere in ingresso, oppure mancata lettura senza spostamento della testina (ε-mossa)
- lettura del carattere sul fronte di ognuna delle due code
- lettura dello stato dell'organo di controllo

(Sulla base dei dati acquisiti con le precedenti letture):

- cambiamento dello stato dell'organo di controllo
- eventuale spostamento a destra della testina del nastro di ingresso (se non si tratta di ε-mossa)
- rimozione del carattere sul fronte di ognuna delle due code
- scrittura di una stringa in fondo a ognuna delle due code

Si definisca poi la configurazione di un tale automa, la relazione di transizione tra configurazioni che formalizza la mossa dell'automa e l'accettazione della stringa di ingresso mediante il posizionamento su uno stato di accettazione quando l'automa giunge a fine stringa di ingresso.

Esercizio 4.b (punti 3/15)
Si mostri come un automa a coda singola, definito in modo del tutto analogo al precedente ma dotato di una sola coda, possa simulare il comportamento di un automa a coda doppia; in altre parole, si mostri come, dato un automa a coda doppia se ne possa costruire uno a coda singola che riconosce lo stesso linguaggio. Non è necessario formalizzare anche l'automa a coda singola: basta descrivere in maniera succinta ma precisa il comportamento del medesimo per simulare quello dell'automa a coda doppia.

NB
I punteggi sono espressi in 15esimi. Il puntegio complessivo ottenuto sarà sommato a quello del scondo compitino, pure espresso in 15esimi. Il massimo punteggio raggiungibile sarà perciò espresso in 30esimi ma sarà superiore a 30/30. Esso, produrrà, con eventuali piccoli aggiustamenti, il voto finale proposto.
Si consiglia di svolgere gli esercizi nell'ordine proposto, riservando la – eventuale – soluzione dell'Esercizio 4.b ad un'ultima fase, solo a valle di una ben meditata soluzione degli esercizi precedenti.

Soluzioni

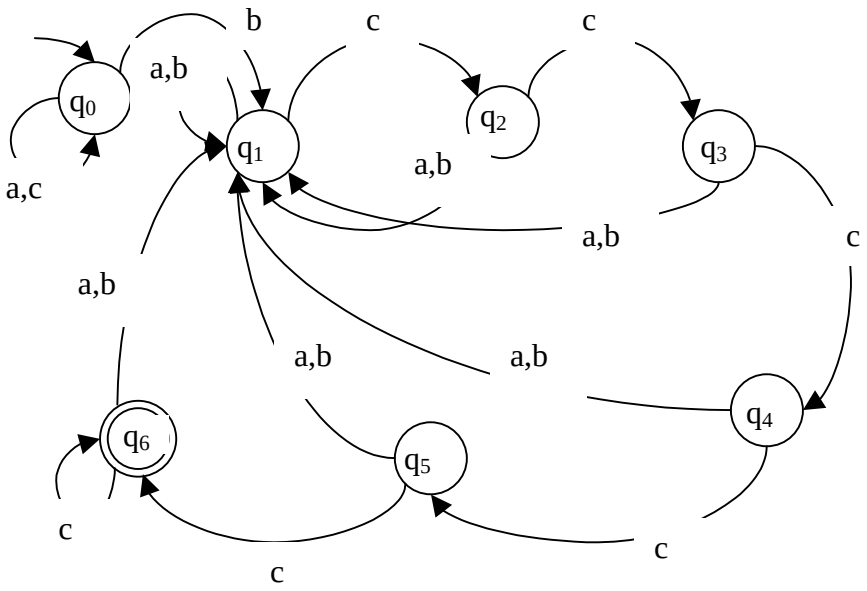
Esercizio 1 (con leggere "abbreviazioni" rispetto alla pura sintassi del prim'ordine)

$x \in L \leftrightarrow x \in \{a,b,c\}^* \wedge (\exists y, z\ ((x = y.b.z.ccccc) \wedge (y, z \in \{a,b,c\}^*)))$

Si omette in quanto ben nota la formula che formalizza l'appartenenza x, y, z ∈ {a,b,c}*.

Esercizio 2

Il linguaggio L è riconosciuto dal seguente automa a stati finiti (deterministico):



Esercizio 3

Essendo L riconosciuto da un automa a stati finiti, esiste sicuramente una grammatica regolare che genera L. Tuttavia la seguente grammatica ha un minor numero di produzioni di una regolare.

S → AbAccccc

A → Aa | Ab | Ac | ε

Esercizio 4.a

Seguendo uno schema tradizionale già adottato per la formalizzazione di altre macchine astratte, l'automa a doppia coda può essere formalizzato nel modo seguente:

ADC: $<Q, I, \Gamma, \delta, q_0, Z_0, F>$, dove i simboli usati hanno lo stesso significato di altri automi.

δ: $(Q \times (I \cup \varepsilon) \times \Gamma \times \Gamma) \rightarrow (Q \times \Gamma^* \times \Gamma^*)$ è la funzione di transizione (δ deve essere soggetta a restrizione analoga a quella adottata per gli automi a pila se si vuole che l'automa ADC sia deterministico).

Una configurazione c di un ADC è definita come:

$c = <q, x, \gamma_1, \gamma_2>$, $q \in Q$, $x \in I^*$, $\gamma_1, \gamma_2 \in \Gamma^*$

La relazione di transizione tra configurazioni è definita nel modo seguente:

$c1 = <q1, x1, \gamma 1_1, \gamma 1_2> \ |- \ c2 = <q2, x2, \gamma 2_1, \gamma 2_2>$ se e solo se

$x1 = x2$, $\gamma 1_1 = A\eta 1_1$, $\gamma 2_1 = \eta 1_1 \lambda 2_1$, $\gamma 1_2 = B\eta 1_2$, $\gamma 2_2 = \eta 1_2 \lambda 2_2$,

e $\delta(q1, \varepsilon, A, B) = <q2, \lambda 2_1, \lambda 2_2>$;

oppure

$x1 = i.x2$, $\gamma 1_1 = A\eta 1_1$, $\gamma 2_1 = \eta 1_1 \lambda 2_1$, $\gamma 1_2 = B\eta 1_2$, $\gamma 2_2 = \eta 1_2 \lambda 2_2$,

e $\delta(q1, i, A, B) = <q2, \lambda 2_1, \lambda 2_2>$

Infine la stringa x è accettata da ADC se e solo se $c_0 = <q_0, x, Z_0, Z_0> \ |-^*- \ c = <q, \varepsilon, \gamma_1, \gamma_2>$ con $q \in F$.

Esercizio 4.b

A grandi linee un ACS (automa a coda singola) può simulare un ADC nel modo seguente:

La coda di ACS contiene, all'inizio e alla fine di una "macro-mossa" che simula una singola mossa di ADC una stringa $\gamma_1\$\gamma_2$ dove, al solito, \$ è un simbolo speciale usato come separatore e γ_1 e γ_2 i contenuti delle due code di ADC.

La macro-mossa di ACS consiste in:

- lettura e memorizzazione mediante memoria a stati finiti del primo carattere di γ_1;
- scorrimento e ricopiatura della parte restante di γ_1 in fondo alla coda, separata da un \$ dalla parte precedente;
- lettura e memorizzazione mediante memoria a stati finiti del primo carattere di γ_2 (subito dopo il \$);

[a questo punto ACS è in possesso (nella sua memoria a stati finiti) di tutte le informazioni necessarie per simulare la mossa di ADC];

- scorrimento e ricopiatura della parte restante di γ_2 in fondo alla coda, separata da un \$ dalla parte precedente;
- scorrimento e ricopiatura della parte restante di γ_1 in fondo alla coda, separata da un \$ dalla parte precedente e seguita dalla stringa λ_1, definita dalla δ di ADC;
- scorrimento e ricopiatura della parte restante di γ_2 in fondo alla coda, separata da un \$ dalla parte precedente e seguita dalla stringa λ_2, definita dalla δ di ADC;
- cancellazione del \$ rimanente sul fronte della coda;
- eventuale avanzamento della testina di lettura;
- cambio dello stato q1 → q2 simulato di ADC.

Informatica Teorica

Seconda prova in itinere e appello del corso del Vecchio ordinamento

29 Giugno 2004

Esercizio 1 (Punti 6)

Si dica, giustificando brevemente la risposta, se le seguenti affermazioni sono vere o false:

a) Il problema di stabilire se un generico programma P, codificato in C, eseguito su un generico dato di ingresso x, non terminerà mai la sua esecuzione è semidecidibile.

b) Il problema di stabilire se un generico programma P, codificato in C, eseguito su un generico dato di ingresso x, "andrà in loop", nel senso stretto del termine, ossia ripeterà indefinitamente una stessa sequenza di "mosse" ritornando ogni volta nello stesso stato di memoria, è semidecidibile.

Esercizio 2 (Punti 4)

Il signor Furbetti sta realizzando un archivio mediante una tabella hash. I record dell'archivio hanno una chiave costituita da sequenze di al più 30 caratteri e la tabella ha una dimensione di 5 MB.
Furbetti è particolarmente soddisfatto perché ha trovato una funzione di hash h "perfetta" (ossia è riuscito a dimostrare che la sua funzione h è tale che $x \neq y$ implica $h(x) \neq h(y)$). Accingendosi però a implementare un algoritmo per il calcolo di h non riesce a trovarne uno. Egli viene allora assalito dal dubbio che la sua h non sia calcolabile.
Potete aiutare Furbetti a risolvere il suo dubbio?

Esercizio 3 (Punti 8)

a) Si descriva sinteticamente –senza necessariamente codificarlo in modo completo- un algoritmo per la macchina RAM per riconoscere il linguaggio $L = \{a^n b^n c^n \mid n >= 1\}$. Se ne valutino complessità spaziale e temporale –a meno della relazione Θ-mediante il criterio di costo logaritmico.
b) E' possibile ottenere le stesse complessità (sia spaziale che temporale) mediante una macchina di Turing? Giustificare brevemente la risposta.

Esercizio 4

Si scriva una grammatica che generi il linguaggio $L = \{a^n w \mid w \in \{b,c\}^* \wedge \#(b,w) = \#(c,w) = n, n >= 1\}$, dove la funzione #(i,x) denota il numero di occorrenze del carattere i nella stringa x.
La grammatica così ottenuta è a potenza minima, ossia non esiste una grammatica ad essa equivalente appartenente ad una classe di minor potenza generativa? Giustificare brevemente la risposta.

Soluzioni

Esercizio 1

a) Falsa. Infatti il problema complementare è notoriamente non decidibile ma semidecidibile. Quindi non può essere semidecidibile anche il problema della ***non***-terminazione.

b) Vera. E' concettualmente possibile registrare ogni stato dell'esecuzione del programma e verificare se lo stato attuale è identico a uno già attraversato. Se ciò accade, da quel momento in poi l'esecuzione si ripeterà identicamente all'infinito.

Esercizio 2

h è certamente calcolabile perché ha un dominio finito. Ciò però aiuta solo in parte Furbetti perché non fornisce indicazioni su come trovare l'algoritmo di calcolo di h.

Esercizio 3

a) La RAM aggiorna un contatore per ogni a letto (costo $\Theta(i)$, i = 1, …, n). Salva il valore del contatore al termine della lettura degli a in due celle diverse. Decrementa la prima copia per ogni b letto e la seconda copia per ogni c letto successivamente. Al termine entrambi i contatori devono contenere esattamente 0. Ne deriva una complessità spaziale, a criterio logaritmico, $\Theta(\log(n))$ e una temporale $\Theta(n.\log(n))$

b) Sì. Una MdT può simulare il comportamento della RAM esattamente allo stesso modo, usando un nastro per ogni contatore e codificandone il contenuto in numerazione binaria. Per ogni carattere letto l'incremento o il decremento del nastro contatore richiede un numero di mosse al più proporzionale alla lunghezza del nastro, ossia $\Theta(\log(n))$. Perciò complessità spaziale e temporale coincidono con quelle della RAM.

   Si noti che un MdT potrebbe riconoscere lo stesso linguaggio con complessità $\Theta(n)$ sia spaziale che temporale.

Esercizio 4

S → aSH | aH

H → BC

BC → CB

CB → BC

B → b

C → c

La grammatica di cui sopra non è non-contestuale (appartiene però alla classe delle grammatiche contestuali, una classe non considerata in questo corso di minor potenza delle grammatiche generali e di maggior potenza di quelle non-contestuali; questa classe genera linguaggi decidibili al contrario delle grammatiche generali). Non è possibile generare lo stesso linguaggio mediante una grammatica non-contestuale perché per riconoscere L occorre tenere aggiornati i conteggi sia delle be che delle c per confrontarli con il numero di a letto inizialmente.

Informatica Teorica

Sezione Cremona + Como

Appello del 20 Luglio 2004

Esercizio 1 (punti 9/15)

Si specifichi, mediante una formula del prim'ordine un apparato che funziona nel modo seguente:

All'istante 0 esso emette un segnale s, che può essere uno 0 o un 1. Se, *dopo* l'emissione di s e *prima* dello scadere di 3 secondi, viene ricevuto lo stesso segnale in risposta, allora allo scadere del terzo secondo viene emesso un nuovo segnale invertendone il valore (0 se prima si era emesso 1 e viceversa); in caso contrario viene riemesso lo stesso segnale inviato in precedenza. Il comportamento dell'apparato si ripete poi periodicamente con periodo di 3 secondi applicando sempre la stessa regola.

L'apparato emette un segnale soltanto in istanti che sono multipli di 3 secondi.

Si suggerisce di usare i predicati emette(s,t) e riceve(s,t) per indicare, rispettivamente l'emissione e la ricezione del segnale s all'istante t.

NB: il domino temporale può essere sia discreto (ad esempio, l'insieme dei naturali) che continuo (ad esempio, l'insieme dei reali).

Esercizio 2 (punti 8/15)

Con riferimento all'esercizio precedente, si assuma un tempo discreto (la cui unità sia il secondo). Si costruisca un'opportuna macchina astratta, preferibilmente a potenza minima, che abbia come alfabeto di ingresso (rispettivamente, di uscita) la ricezione (risp., emissione) di 0, la ricezione (risp., emissione) di 1, oppure l'assenza di segnale e si comporti come specificato nell'esercizio 1.

NB: essendo il comportamento dell'apparato periodico, senza che sia prevista una sua terminazione, la macchina astratta, di conseguenza, non dovrà prevedere una situazione di arresto, a meno che non debba bloccarsi a causa di errori.

Esercizio 3 (punti 7/15)

Si dica, giustificando brevemente la risposta, quali di queste affermazioni sono vere e quali false:

1. La funzione g(y,x) = (1 se $f_y(x)$ è pari, 0 altrimenti) è calcolabile.
2. La funzione g(y,x) = (1 se $f_y(x)$ è pari, $\perp$ altrimenti) è calcolabile
3. La funzione g(y) = (1 se $f_y(x)$ è pari $\forall x$, 0 altrimenti) è calcolabile
4. La funzione g(y) = (1 se $f_y(x)$ è pari $\forall x$, $\perp$ altrimenti) è calcolabile

NB: Nel caso la risposta a qualcuna delle domande precedenti fosse "Falsa" e se ne fornisse una dimostrazione mediante tecnica diagonale, il punteggio acquisito verrebbe maggiorato di punti 3.

Esercizio 4 (punti 8/15)

Si descriva con sufficiente precisione, ma senza necessariamente specificare ogni dettaglio, come una Macchina di Turing a nastro singolo deterministica (si ricordi che una Macchina "a nastro singolo" è diversa da una macchina a k nastri con k = 1) possa riconoscere il linguaggio L = {ww, w $\in$ {a,b}*} analizzandone la complessità spaziale e temporale (a meno dell'ordine di grandezza determinato dalla Θ–equivalenza).

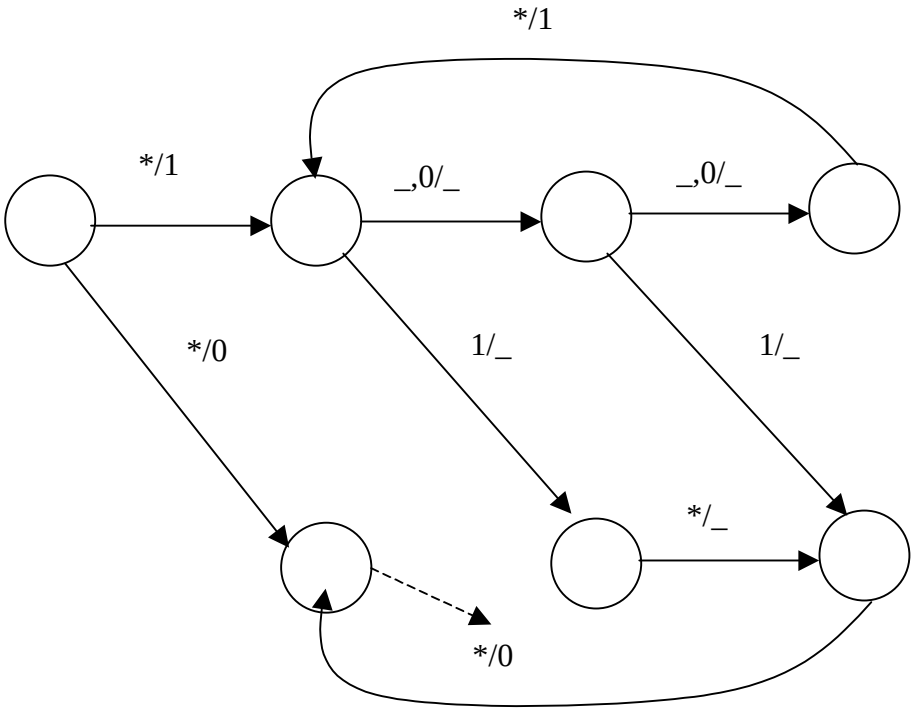
Soluzioni

Esercizio 1

$$
\begin{aligned}
&\forall s \\
&(s \in \{0,1\} \\
&\wedge \\
&emette(0,0) \vee emette(1,0) \wedge \\
&\forall t,k(k \in N \wedge k > 0 \wedge t = 3.k \rightarrow \\
&\quad (emette(s,(k-1).t) \rightarrow \\
&\quad\quad (\exists t_1((k-1).t < t_1 < k.t \wedge riceve(s,t_1)) \rightarrow emette((s+1) \bmod 2, t)) \wedge \\
&\quad\quad (\neg\exists t_1((k-1).t < t_1 < k.t \wedge riceve(s,t_1)) \rightarrow emette(s,t)) \\
&\quad)) \\
&\quad) \\
&\wedge \\
&emette(s,t) \rightarrow \exists k \in N \wedge t = 3.k \\
&)
\end{aligned}
$$

Esercizio 2

La figura sottostante descrive parzialmente un automa a stati finiti che si comporta secondo le specifiche. La parte omessa, cui si accede attraverso la freccia tratteggiata, tratta il caso dell'emissione iniziale di uno 0 in modo simmetrico.



Legenda: * indica un qualsiasi simbolo; _ indica assenza di segnale; a, b/ etichetta una transizione che avviene in conseguenza dll'input a o dell'input b.

Esercizio 3

1. Falsa. E' possibile ridurre il problema della terminazione al problema dato assegnando al risultato della funzione il valore 2 a valle del calcolo di una funzione generica $f_z(w)$di cui si voglia stabilire la terminazione.

   Si può sfruttare per la dimostrazione anche il teorema di Rice nel seguente modo: si fissi un certo $\underline{x}$ (qualunque, per esempio 7) e si definisca la funzione h(y) = (1 se $f_y(\underline{x})$ è pari, 0 altrimenti). h(y) è la funzione caratteristica dell'insieme delle macchine di Turing che calcolano funzioni che danno risultato pari quando hanno come input il numero $\underline{x}$. Questo insieme non è né l'insieme vuoto, né l'insieme universo, e non è dunque decidibile, quindi la sua funzione caratteristica non è calcolabile. Se g(x,y) fosse calcolabile, lo sarebbe anche h(y),il che porterebbe a una contraddizione.

   La dimostrazione può anche essere costruita in forma diretta usando la tecnica diagonale nel modo seguente:
   Si definisca h(x) = Se $f_x(x)$ non è pari allora 2, $\perp$ altrimenti. Se g fosse calcolabile lo sarebbe anche h. Quindi h dovrebbe essere = f_{x0} per qualche x0.Ma h(x0) non pari implicherebbe h(x0) = 2 e h(x0) = 2 implicherebbe h(x0) non pari.

2. Vera: basta fare il run di $f_y(x)$

3. Falsa grazie al teorema di Rice: infatti la g(y) è la funzione caratteristica dell'insieme di tutte (e sole) le macchine di Turing che computano funzioni totali con valore sempre pari. Questo insieme non è né l'insieme vuoto, né l'insieme universo, non è quindi decidibile, e la sua funzione caratteristica non è computabile.

4. Falsa. Infatti questa è la funzione semicaratteristica dell'insieme di tutte (e sole) le macchine di Turing che computano funzioni totali con valore sempre pari. Se tale funzione fosse calcolabile, ciò vorrebbe dire che tale insieme di funzioni è RE. Questo è tuttavia impossibile, Si prenda infatti una funzione calcolabile f(x) generica; si può facilmente calcolare la funzione f'(x) = 2f(x), la quale ha lo stesso dominio di f(x) ed ha, laddove è definita, valore pari. Di conseguenza, a partire dall'insieme di tutte le funzioni computabili a valori solo pari è possibile costruire l'insieme di tutte le funzioni computabili semplicemente dividendo per 2 le immagini. Da questo deriva che se l'insieme di tutte e sole le funzioni totali a valori pari fosse RE, anche l'insieme di tutte e sole le funzioni totali lo sarebbe. E' tuttavia ben noto (da lezione) che l'insieme di tutte e sole le funzioni totali non è RE, di conseguenza non lo è neanche l'insieme di funzioni di partenza.

Esercizio 4

La macchina M deve in primo luogo individuare la metà della stringa in ingresso. Per ottenere ciò, mediante una prima passata memorizza –ad esempio in unario- la lunghezza della stringa, alla sua destra (durante questa passata la macchina dovrà tenere traccia del carattere fino a dove si è contato, per esempio cambiandolo da a ad a', e da b a b', e riconvertendolo nel carattere originale prima di passare a contare il prossimo carattere). Ciò richiede un tempo $\Theta(n)$ per ogni carattere letto e quindi $\Theta(n^2)$ per l'intera stringa. Indi, per ogni coppia di caratteri della stringa che memorizza la lunghezza dell'input, sposta di una posizione un'opportuna marca all'interno della stringa di input (ad esempio sostituendo il carattere a

con il carattere a’ e b con b’). Al termine la marca si troverà a metà della stringa di ingresso. Anche questa macro-operazione richiede un tempo $\Theta(n^2)$ (Θ(n) per ogni coppia di caratteri).

A questo punto M può procedere a confrontare, uno per uno il primo carattere dell’input con il carattere in corrispondenza della marca che segna la metà; il secondo con il successivo. Anche questo confronto richiede un tempo Θ(n) per ogni carattere e quindi $\Theta(n^2)$ per l’intera stringa.

In conclusione l’intero procedimento ha una complessità temporale $\Theta(n^2)$ e spaziale Θ(n).

Informatica Teorica

Sezione Cremona + Como

Appello del 15 Settembre 2004

Esercizio 1 (punti 8)

Si specifichi, mediante una formula del prim'ordine il ripetersi periodico di un segnale istantaneo (ossia che si verifica in un istante di tempo isolato) con periodo 2 unità di tempo a partire dall'istante 5. Prima dell'istante 5 e in tutti gli altri istanti il segnale non si verifica.

Esercizio 2 (punti 9)

Si costruisca un automa a pila equivalente alla grammatica seguente (ossia che riconosca il linguaggio da essa generato):

S → aAB

A → aAA | bAB | c

B → bBB | aBA | c

L'automa così costruito è deterministico? In caso negativo, è possibile costruirne uno equivalente deterministico?

Esercizio 3 (punti 7)

Si dica, giustificando brevemente la risposta, quali delle seguenti affermazioni sono vere e quali false:

1. L'insieme dei programmi C la cui terminazione è garantita a priori per ogni valore dei dati in ingresso è ricorsivamente enumerabile.
2. L'insieme dei programmi C la cui terminazione è garantita a priori per ogni valore dei dati in ingresso è ricorsivo.

Esercizio 4 (punti 7)

In passato alcuni studenti, nello svolgimento di prove d'esame, hanno costruito una valutazione di complessità per macchine di Turing dichiarando di utilizzare il criterio di costo logaritmico. Si spieghi brevemente perché ciò è un errore e perché invece tale criterio è utile e spesso necessario nella valutazione di complessità relativa a un modello di calcolo come la RAM.

Soluzioni

Esercizio 1

Codificando l'occorrenza del segnale all'istante t mediante il predicato s(t), la formula seguente specifica il comportamento desiderato:

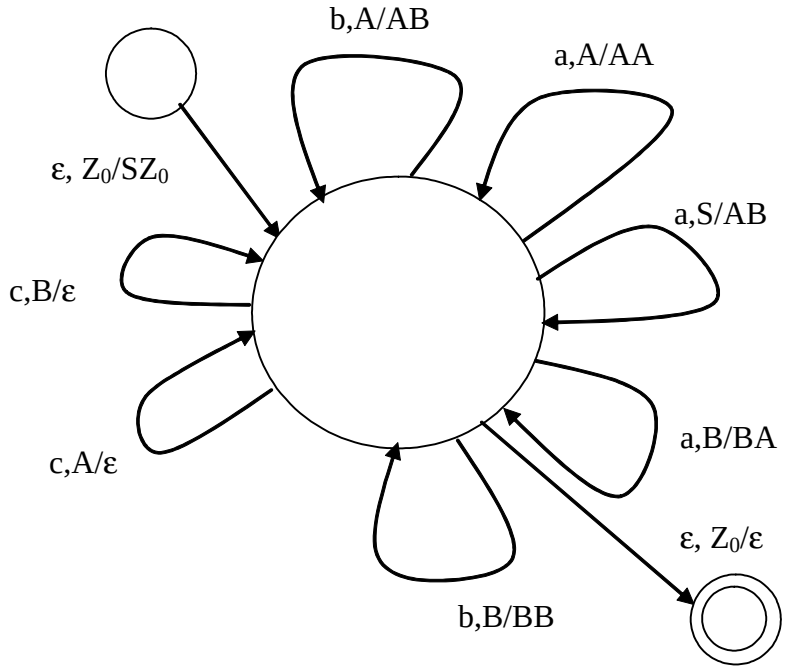
$$
\begin{aligned}
&s(5) \wedge \\
&\forall t((t < 5) \rightarrow \neg s(t) \wedge \\
&\quad (t \geq 5 \rightarrow (s(t) \leftrightarrow (\exists k(k \in N \wedge t = 5 + 2k)))))
\end{aligned}
$$

Si noti che possiamo omettere i primi termini, dal momento che sono implicati dall'ultimo e scrivere più sinteticamente solo:

$\forall t\ (\mathrm{s}(t) \ \leftrightarrow\ (\exists k\ (k \in \mathrm{N} \wedge t = 2k + 5)))$

Esercizio 2

La figura sottostante descrive un automa a pila (deterministico) equivalente alla grammatica fornita.



Legenda: la stringa a destra del carattere / viene depositata sulla pila mettendo il carattere più a sinistra in alto.

Esercizio 3

Entrambe le affermazioni sono false. Infatti è noto (Teorema 2.10 del testo) che un insieme di indici di Macchine di Turing che calcolino tutte e sole le funzioni calcolabili e totali non è ricorsivamente enumerabile. Se perciò l'insieme suddetto di programmi C fosse ricorsivamente enumerabile, potremmo automaticamente calcolare per ogni tale programma un corrispondente indice di MT che calcola la stessa funzione del programma, e quindi ricavare da una enumerazione ricorsiva dell'insieme dei programmi una corrispondente enumerazione ricorsiva di ***un*** insieme di indici di MT che calcolano tutte e sole le funzioni calcolabili e totali.

A maggior ragione, se tale insieme non è ricorsivamente enumerabile, non può essere ricorsivo.

Esercizio 4

Le operazioni astratte della MT (transizioni) sono effettivamente elementari e possono essere realizzate a "risorse costanti" con qualunque tecnologia di base. Al contrario le operazioni astratte della RAM, più di alto livello, sono realizzate mediante opportuno HW la cui complessità, intermini di circuiteria e utilizzata e tempo impiegato è funzione –appunto logaritmica- della dimensione dei dati in ingresso.

Informatica Teorica

Sezione Cremona + Como

Appello del 5 Febbraio 2005

Esercizio 1 (punti 8)

Si specifichi, mediante una formula del prim'ordine il seguente comportamento di un ipotetico sistema:

Tutte le volte che, in un certo istante, si verifica l'evento A, dopo esattamente k unità di tempo si verifica l'evento B, a meno che, contemporaneamente ad A, non si verifichi anche l'evento C. In tal caso, B non si verifica dopo k unità di tempo, bensì dopo k+1.

Esercizio 2 (punti 8)

Si considerino i seguenti linguaggi:

$L1 = \{a^n b^{2m} | \ n, m >= 1\}$, $L2 = \{a^n b^{3n} | \ n >= 0\}$

Si costruisca una grammatica G che generi il linguaggio $L = L1 \cap L2$. E' preferibile una grammatica a potenza minima, ossia regolare se ne esiste una, non contestuale se ne esiste una ma non ne esiste una regolare, ecc. Nel caso, si spieghi brevemente perché non esistono grammatiche appartenenti a classi inferiori a quella proposta.

Esercizio 3 (punti 6)

Si dica, giustificando brevemente la risposta, se è decidibile il problema di stabilire se il linguaggio L di cui all'esercizio 2 è vuoto o no.

Esercizio 4 (punti 8)

Si dica, giustificando brevemente la risposta, qual è l'ordine di grandezza minimo per il riconoscimento del linguaggio L di cui all'esercizio 2 nei due casi in cui si usi

a) Una macchina di Turing.

b) Una RAM, assumendo il criterio di costo logaritmico.

Soluzioni

Esercizio 1

Codificando l'occorrenza di un generico evento E all'istante t mediante il predicato E(t), la formula seguente specifica il comportamento desiderato:

$$\forall t((A(t) \wedge \neg C(t)) \rightarrow B(t+k)) \wedge$$
$$((A(t) \wedge C(t)) \rightarrow (\neg B(t+k) \wedge B(t+k+1)))$$

Si noti che la formula di cui sopra porta ad una contraddizione se non solo all'istante *t*, ma anche all'istante *t-1* accadono sia A che C (cosa che la specifica a parole in principio non vieta). In questo caso, infatti, il secondo congiunto applicato a *t-1* vincola B ad accadere a *t+k*, mentre lo stesso congiunto riferito all'istante *t* proibisce che B accada a *t+k*. Una formula che evita questo fenomeno, reinterpretando la specifica a parole è la seguente:

$$\forall t((A(t) \wedge \neg C(t)) \rightarrow B(t+k)) \wedge$$
$$((A(t) \wedge C(t)) \rightarrow B(t+k+1))$$

Esercizio 2

Il linguaggio L = L1 $\cap$ L2 è l'insieme $\{a^{2n}b^{6n} | n >= 1\}$. Infatti, L1 è il linguaggio delle stringhe con un numero *pari* di 'b' che seguono un numero qualunque di 'a'. Tra le stringhe di L2, del tipo '$a^n b^{3n}$', quelle che appartengono anche a L1 sono esattamente quelle con un numero pari di 'b', ossia quelle nella forma '$a^{2n}b^{3\cdot(2n)} = a^{2n}b^{6n}$'.

L è generato dalla seguente grammatica non contestuale (e da nessuna grammatica regolare essendo necessaria una pila per il suo riconoscimento):

S → AB | ASB

A → aa

B → bbbbbb

Esercizio 3

L non è vuoto: contiene aabbbbbb, ecc. Quindi il problema L = ∅ è *deciso* e perciò *decidibile*.

Esercizio 4

Una MT che simuli un automa a pila deterministico può facilmente riconoscere L in tempo lineare. Una RAM deve necessariamente aggiornare un contatore (il cui contenuto è un numero proporzionale ad n) per ogni carattere letto. Perciò, a criterio di costo logaritmico, la sua complessità è necessariamente $\Theta(n.\log(n))$.