Informatica Teorica
Prima Semiunità

Secondo compitino – 18 Gennaio 2001

Esercizio 1 (10 punti)

Si dica se è decidibile il seguente problema:

Stabilire se la funzione f, con dominio e codominio N (l'insieme dei numeri naturali), calcolata dalla i-esima macchina di Turing M_i gode della proprietà seguente:

$$\forall x \Big(\big(\neg(x >= 0) \rightarrow f(x) \neq 5 \big) \wedge \big((x >= 0) \rightarrow \big(f(x) > 37 \vee \big((\neg\, (f(x) = \perp) \rightarrow (f(x) < 100)) \big) \big) \big) \Big)$$

Esercizio 2 (15 punti)

Si descriva un algoritmo per la macchina RAM che calcoli la funzione

f: N → N; f(x) = **if** x pari **then** $\sqrt{x^x}$ **else** x^3

e se ne valuti l'ordine di grandezza della complessità temporale sia a criterio di costo costante che a criterio logaritmico, assumendo –***preferibilmente***- come parametro *n* della dimensione del dato *x* in ingresso la lunghezza della stringa che lo codifica in binario.

NB 1

Non è strettamente necessario produrre un algoritmo di complessità ottima, ma certamente la qualità della soluzione verrà valutata anche in base alla complessità ottenuta.

NB 2

Non è necessario codificare in dettaglio l'algoritmo: è sufficiente descriverlo a un livello di precisione sufficiente da permetterne la valutazione corretta e precisa della complessità (a meno della relazione Θ); ad esempio mediante pseudolinguaggio di alto livello del tipo di SMALG o simili.

Soluzioni

Esercizio 1
La formula che deve essere soddisfatta da f è sempre vera. Infatti
$(\neg(x >= 0) \rightarrow f(x) \neq 5)$
è vera perché $\neg(x >= 0)$ è falso per ogni x in N;

$((x >= 0) \rightarrow (f(x) > 37 \vee ((\neg (f(x) = \perp) \rightarrow (f(x) < 100)))))$
è pure vera perché:
x è sempre >= 0
se $f(x) = \perp$) è vero il secondo ramo dell'$\vee$ (perché ne è falso l'antecedente)
se $f(x) \neq \perp$) e f(x) >37 è vero il primo ramo dell'$\vee$;
se invece ($f(x) \neq \perp$) e f(x) <= 37, è vero il secondo ramo dell'$\vee$ perché f(x) <= 37 implica f(x) < 100.
Perciò ogni macchina di Turing calcola una funzione che gode della proprietà specificata ed il problema risulta deciso oltre che decidibile.

Esercizio 2
Un semplice algoritmo che calcola la funzione f è il seguente:

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

Chiaramente il ramo **else** può essere trascurato ai fini della valutazione di complessità (nel caso pessimo).
A criterio di costo costante la complessità temporale é $\Theta(x)$. Se indichiamo con n la lunghezza di x, essa é dunque $\Theta(2^n)$.
A criterio di costo logaritmico invece la complessità della singola operazione
ris := ris*x é $\Theta(\log(x^i))$, ossia $\Theta(i.\log(x))$. La complessità del ciclo è perciò $\Theta(m^2.\log(x)) = \Theta(x^2.\log(x)) = \Theta(n.(2^n)^2) = \Theta(n.(2^{2n}))$.

E' però possibile ottenere una migliore complessità (analizzata qui solo per il criterio logaritmico) mediante l'algoritmo seguente, leggermente più sofisticato (limitandosi al caso x pari e > 1):

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

una singola esecuzione del ciclo costa, per i valori massimi di ris, $\Theta(x.\log(x))$, come per l'algoritmo precedente. In questo caso però il ciclo viene eseguito al più log(m) + 1 volte.
Di conseguenza la complessità totale è $\Theta(x.\log^2(x))$, ossia $\Theta(n^2.(2^n))$.

Informatica Teorica
Prima Semiunità

Appello del 7 Febbraio 2001

Esercizio 1

Si formalizzi mediante una formula del prim'ordine la seguente affermazione:

"Se un problema P è decidibile allora è anche semidecidibile, ma se è semidecidibile e il suo complemento non è decidibile, allora P non è decidibile. Invece, se P non è semidecidibile il suo complemento non è decidibile."

Esercizio 2

Si dica, motivando brevemente la risposta, se l'affermazione di cui all'esercizio 1 è vera o falsa.

Esercizio 3

Si descriva brevemente –senza necessariamente codificarlo in dettaglio- un algoritmo per la macchina RAM (o per qualunque altro linguaggio di alto livello del tipo del Pascal) che riceva in ingresso 3 numeri naturali, x, y, k, codificati come sequenze di cifre decimali, e separati tra loro da opportuni caratteri speciali, e scriva sul nastro di uscita il rapporto x/y codificato anch'esso da una sequenza di cifre decimali facendo uso del carattere speciale ',' per separare la parte intera dalla parte decimale e troncando la parte decimale dopo k cifre. Le k cifre della parte decimale devono essere scritte anche se fossero 0.

Ad esempio, se x = 10, y = 3, k = 12. Il nastro di ingresso potrebbe essere rappresentato come segue:

1	0	#	3	#	1	2	#						

E il risultato a sua volta potrebbe essere rappresentato sul nastro di uscita nella maniera seguente:

3	,	3	3	3	3	3	3	3	3	3	3	3	3

Si valuti poi la complessità temporale dell'algoritmo sia a criterio di costo costante che a criterio logaritmico.

Note:

- Si può assumere che la RAM sia dotata delle operazioni MULT, DIV e MOD che calcolano rispettivamente il prodotto, il quoto e il resto della divisione ***intera*** tra due numeri. Inoltre, per semplicità, si può assumere che il loro costo a criterio logaritmico sia analogo a quello delle operazioni additive.
- Non viene richiesto un algoritmo particolarmente sofisticato: anche il tradizionale algoritmo usato per il calcolo manuale della divisione è da considerarsi più che soddisfacente.

- Si sottolinea che la complessità va calcolata in funzione della lunghezza della stringa di ingresso.

Soluzioni

Esercizio 1

Simboli utilizzati:

P: simbolo di variabile indicante un generico problema

Comp: simbolo di funzione indicante il complemento di un problema

Dec: simbolo di predicato indicante il fatto che un problema sia decidibile

SemDec: simbolo di predicato indicante il fatto che un problema sia semidecidibile

$$\forall P \quad (Dec(P) \rightarrow SemDec(P)) \wedge$$
$$(SemDec(P) \wedge (\neg\ Dec(Comp(P)) \rightarrow \neg\ Dec(P)\ ) \wedge$$
$$(\neg\ SemDec(P) \rightarrow \neg\ Dec(Comp(P)))$$

Esercizio 2

L'affermazione è vera, in quanto sono vere tutte le sue componenti:

Se P è decidibile è anche semidecidibile

Se P è semidecidibile e il suo complemento non è decidibile, P non può essere anche decidibile perché gli insiemi decidibili sono chiusi rispetto al complemento.

Per la stessa ragione non può essere che P sia non semidecidibile e il suo complemento decidibile.

Esercizio 3

```
char in[MAX] = {"1000#3#13#"};

unsigned int c2ui(char c) {
  return ((unsigned int) c)-((unsigned int) '0');
}

void main () {
   unsigned int n=0;
   unsigned int x=0, y=0, k=0;
   unsigned int i = 0;
   unsigned int z, c, r;
   while (in[i] != '#')
     x=x*10+c2ui(in[i++]);
   i++;
   while (in[i] != '#')
     y=y*10+c2ui(in[i++]);
   i++;
   while (in[i] != '#')
     k=k*10+c2ui(in[i++]);
   z = x/y;
   printf("%d", x/y);
   c=0;
   printf(".");
   r = x%y;
   }
   while (c<k) {
     printf("%d", (r*10)/y);
     r=(r*10)%y;
```

```
      c++;
    }
    printf("\n");
}
```

Analisi di complessità a criterio di costo costante:

I primi tre cicli compiono un'operazione per ogni carattere letto dall'ingresso, quindi il loro contributo è $\Theta(n)$.

Per il resto, occorre valutare il numero di caratteri che compongono x, y e k:

k può facilmente essere composto da un numero di caratteri proporzionale ad n (mediamente n/3). In questo caso, l'ultimo ciclo viene eseguito al più $\Theta(10^n)$ volte.

Si può quindi concludere che la complessità del programma è $\Theta(10^n)$.

Analisi di complessità a criterio di costo logaritmico:

I cicli iniziali costruiscono almeno un numero dell'ordine di $\Theta(10^n)$, quindi il costo è $\Theta(n \log 10^n) = \Theta(n^2)$.

Per il resto è facile verificare che il costo è ancora determinato dal ciclo while. Occorre però considerare che le operazioni contenute nel ciclo coinvolgono un termine r che al massimo diventa dell'ordine di 10*y, cioè $\Theta(10^{n+1})$. Il costo di ciascuna operazione è quindi $\Theta(\log 10^{n+1}) = \Theta(n)$. Il costo dell'ultimo ciclo e dell'intero algoritmo risulta perciò $\Theta(n10^n)$.

Informatica Teorica
Prima Semiunità

Appello del 22 Febbraio 2001

Esercizio 1

Si costruisca una grammatica che generi il linguaggio L definito dalla seguente formula del I ordine:

$\forall x\ (x \in L \leftrightarrow \exists y,z,w\ (\ x = y.z.w \wedge y \in L1 \wedge z \in L2 \wedge w \in L3)$

$\wedge$

$\forall x\ (x \in L1 \leftrightarrow (x = aa \vee x = bb \vee \exists y\ (\ x = a.y.a \wedge y \in L1 \vee x = b.y.b \wedge y \in L1)$

$\wedge$

$\forall x\ (x \in L2 \leftrightarrow (x = \varepsilon \vee \exists y\ (\ x = ab.y \wedge y \in L2)$

$\wedge$

$\forall x\ (x \in L3 \leftrightarrow (x = \varepsilon \vee \exists y\ (\ x = a.y.b \wedge y \in L3)$

Esercizio 2

Per ognuna delle affermazioni seguenti si dica, spiegandone brevemente le ragioni, se esse sono vere o false.

1. Data una qualsiasi funzione calcolabile f esiste sempre un algoritmo A che la calcola, codificato nel linguaggio della RAM, la cui complessità, valutata a criterio di costo logaritmico sia <= di un opportuno polinomio applicato alla complessità valutata a criterio di costo costante.

2. Per qualche funzione calcolabile f esiste un algoritmo A che la calcola, codificato nel linguaggio della RAM, tale per cui non esista un algoritmo A1 ad esso equivalente la cui complessità, valutata a criterio di costo logaritmico sia <= di un opportuno polinomio applicato alla complessità di A valutata a criterio di costo costante.

3. **(Facoltativa)** Data una qualsiasi funzione calcolabile f esiste sempre un algoritmo A che la calcola, codificato nel linguaggio della RAM, tale per cui non esista un algoritmo A1 ad esso equivalente la cui complessità, valutata a criterio di costo logaritmico sia <= di un opportuno polinomio applicato alla complessità di A valutata a criterio di costo costante.

Soluzioni

Esercizio 1

L è la concatenazione di L1, L2, L3 con

L1 = $\{x|\ x = w.w^R, w \in \{a,b\}^+\}$, L2 = $\{ab\}^*$, L3 = $\{a^nb^n, n >= 0\}$

Perciò la G seguente genera L:

S → ABC

A → aa | bb | aAa | bAb

B → ε | abB

C → ε | aCb

Esercizio 2

1. Vera: basta che A sia un algoritmo che simuli il comportamento di una MT che calcola f.

2. Vera: è ben noto che la funzione 2^{2^n} è calcolabile in tempo lineare a criterio di costo costante ma ha complessità spaziale -e quindi temporale- almeno esponenziale a criterio logaritmico.

3. Falsa: alcune funzioni (ad esempio la gran parte di quelle calcolate da automi a stati finiti) hanno complessità lineare sia a criterio di costo costante che a criterio di costo logaritmico. Non è quindi possibile trovare un algoritmo che a criterio di costo costante abbia complessità Θ-< che a criterio di costo logaritmico.

Informatica Teorica
Prima Semiunità

Appello del 10 luglio 2001

Esercizio 1

Si consideri il linguaggio seguente:

$L = \{a^n b^m, n>0, m = f(n)\}$.

A. Si dia un esempio di f(n) per cui L sia generabile da una grammatica regolare.

B. Si dia un esempio di f(n) per cui la grammatica a potenza minima in grado di generare L sia noncontestuale.

C. Si dia un esempio di f(n) per cui la grammatica a potenza minima in grado di generare L sia non ristretta.

Esercizio 2

Si consideri un dispositivo elettrico alimentato da batteria, inizialmente carica e non ricaricabile.

Il dispositivo è inizialmente spento. Il dispositivo può funzionare in due modi (1 e 2) che assorbono quantità diverse di potenza: in modo 1 il consumo è tale da consentire il funzionamento per un periodo T, mentre in modo 2 l'assorbimento di potenza è doppio. L'evento CM provoca il passaggio da un modo all'altro (se il dispositivo stava funzionando in modo 1 lo porta a funzionare in modo 2, e viceversa).

Al tempo t_0 il dispositivo viene acceso, iniziando a funzionare in modo 1. Il dispositivo si spegne solo quando la batteria si scarica. Specificare il funzionamento del dispositivo.

Soluzione Esercizio 1

A. f(n)=k, dove k è un intero costante (non negativo).
B. f(n)=kn, dove k è una costante intera positiva. f(n)=n/k va ugualmente bene.
C. $f(n)=n^k$, dove k è una costante intera > 1.

Soluzione Esercizio 2

Stati: Spento(t), Acceso(t,K,t_c). Nello stato acceso K indica il modo e può valere 1 o 2. t_c indica lo stato di carica in termini di tempo di funzionamento residuo in modo 1.

Evento: CM(t) indica il cambiamento di modo.

$\exists\ t_c \wedge t_c>0 \wedge (\text{Acceso}(t,1,t_c) \vee \text{Acceso}(t,2,t_c)) \rightarrow \neg\text{Spento}(t)$

$\text{Spento}(t) \rightarrow \neg\exists\ t_c \wedge t_c>0 \wedge (\text{Acceso}(t,1,t_c) \vee \text{Acceso}(t,2,t_c))$

$\text{Spento}(t) \vee (\exists\ t_c \wedge t_c>0 \wedge (\text{Acceso}(t,1,t_c) \vee \text{Acceso}(t,2,t_c)))$

$\forall t_c\ (\neg\exists\ K\ (\text{Acceso}(t,K,t_c) \wedge K\neq 1 \wedge K\neq 2)$

$\text{Acceso}(t_0,1,T) \wedge \forall t_1\ (t_1<t \rightarrow \text{Spento}(t))$

$\text{Acceso}(t,1,t_c) \wedge t_c>0 \wedge \neg\text{CM}(t) \rightarrow \exists\ t_2\ (t<t_2\leq t+t_c \wedge (\forall t_3\ (t\leq t_3\leq t_2 \rightarrow \neg\text{CM}(t_2) \wedge \text{Acceso}(t_3,1,t_c-(t_3-t))))$

$\text{Acceso}(t,2,t_c) \wedge t_c>0 \wedge \neg\text{CM}(t) \rightarrow \exists\ t_2\ (t<t_2\leq t+t_c/2 \wedge (\forall t_3\ (t\leq t_3\leq t_2 \rightarrow \neg\text{CM}(t_2) \wedge \text{Acceso}(t_3,2,t_c-2(t_3-t))))$

$\text{Acceso}(t,1,t_c) \wedge t_c>0 \wedge \text{CM}(t) \rightarrow \exists\ t_1\ (t<t_1\leq t+t_c/2 \wedge \forall t_2\ (t<t_2\leq t_1 \rightarrow \neg\text{CM}(t_2) \wedge \text{Acceso}(t_2,2,t_c-2(t_2-t))))$

$\text{Acceso}(t,2,t_c) \wedge t_c>0 \wedge \text{CM}(t) \rightarrow \exists\ t_1\ (t<t_1\leq t+t_c \wedge \forall t_2\ (t<t_2\leq t_1 \rightarrow \neg\text{CM}(t_2) \wedge \text{Acceso}(t_2,1,t_c-(t_2-t))))$

$(\text{Acceso}(t,1,0) \vee \text{Acceso}(t,2,0)) \rightarrow \forall t_1\ (t_1\geq t \rightarrow \text{Spento}(t))$

Informatica Teorica
Prima Semiunità

Appello del 25 luglio 2001

Esercizio 1

Si consideri il linguaggio L definito come segue:

$s \in L \leftrightarrow \exists\, y,m\ (s=c^m{\cdot}y \wedge m>0 \wedge y \in L_Y)$

$s \in L_Y \leftrightarrow s=\varepsilon \vee \exists\, x,m\ (s=a^m{\cdot}x \wedge m>0 \wedge x \in L_X)$

$s \in L_X \leftrightarrow s=\varepsilon \vee \exists\, y,m\ (s=b^m{\cdot}y \wedge m>0 \wedge y \in L_Y)$

D. Si scriva una grammatica che generi il linguaggio L.

E. Si dica (giustificando la risposta) quale automa a potenza minima riconosce L.

F. Si dica (giustificando la risposta) con quale complessità temporale l'automa di cui al punto B riconosce L.

Esercizio 2

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

A. Date due generiche funzioni f1 e f2 definite sui numeri naturali è decidibile il problema di stabilire se $\forall x\ f1(x) \neq f2(x)$. (NB: i dati del problema sono le funzioni f1, f2, non x!)

B. Date due generiche funzioni *calcolabili* f1 e f2 definite sui numeri naturali è decidibile il problema di stabilire se $\forall x\ f1(x) \neq f2(x)$.

C. Dati due generici polinomi P1 e P2 definiti sui numeri naturali è decidibile il problema di stabilire se $\forall x\ P1(x) \neq P2(x)$.

D. Data una funzione computabile f e un polinomio P definiti sui numeri naturali è decidibile il problema di stabilire se $\forall x\ f(x) \neq P(x)$.

Soluzione Esercizio 1

D. $S \rightarrow cS \mid cY$

$Y \rightarrow \varepsilon \mid AX$

$X \rightarrow \varepsilon \mid BY$

$A \rightarrow a \mid aA$

$B \rightarrow b \mid bB$

E. Basta un automa a stati finiti.

F. $\Theta(n)$

Soluzione Esercizio 2

A: falso

B: falso

C: vero

D: falso.

Informatica Teorica
Prima Semiunità

Appello del 20 Settembre 2001

Esercizio 1

Parte a

Si formalizzi mediante un'opportuna macchina astratta il comportamento del sistema seguente:

Un apparato di controllo è predisposto al monitoraggio di un impianto.

Quando il sistema si trova in condizioni normali di funzionamento, esso riceve da un sensore un segnale proveniente dall'impianto ad istanti regolarmente distanziati (ad esempio con un periodo di k secondi). Se il segnale non evidenzia anomalie il sistema rimane in condizioni di funzionamento normali. Se invece il segnale indica la presenza di anomalie l'apparato di controllo può, indifferentemente, decidere di spegnere l'impianto oppure di segnalare l'anomalia ad un operatore umano.

Nel primo caso il sistema rimane spento finché non viene dato un comando di ripartenza dall'esterno. Nel secondo caso:

Se l'operatore esegue un'azione di riparazione il sistema riprende il funzionamento normale

Se invece l'operatore non esegue alcuna azione entro k secondi oppure invia un comando esplicito di spegnimento, l'impianto viene spento e rimane spento finché non viene dato un comando di ripartenza dall'esterno.

Parte b

Si dica se la macchina astratta costruita nella parte a dell'esercizio è deterministica o no. Nel secondo caso, è possibile costruirne una equivalente deterministica? Se la risposta è positiva, la si costruisca.

Esercizio 2

Si formalizzi mediante una formula del primo ordine la seguente proprietà del sistema di cui all'esercizio 1:

"Il sistema non si trova mai in condizioni di funzionamento non normali per più di x secondi"

NB: *non è detto che il sistema, come formalizzato nell'esercizio 1 debba godere di tale proprietà.*

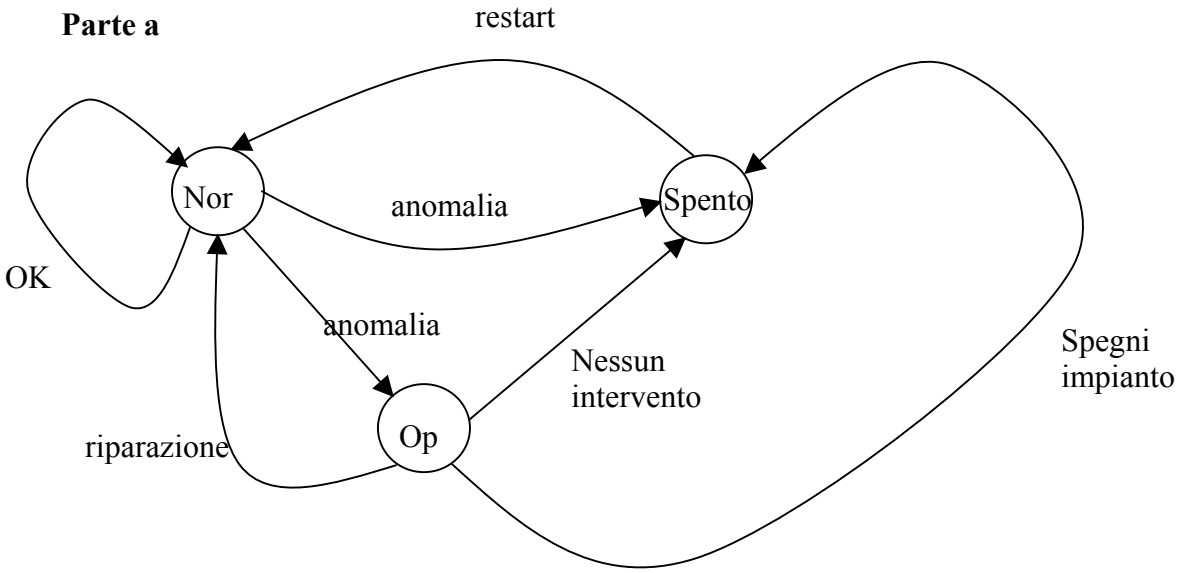
Esercizio 3

Si dica, giustificando brevemente la risposta, se la seguente affermazione è vera o falsa: "Se un problema P è semidecidibile ma non decidibile, allora P non appartiene all'insieme P, ossia l'inisieme dei problemi risolvibili da un algoritmo deterministico in tempo polinomiale".

Soluzioni

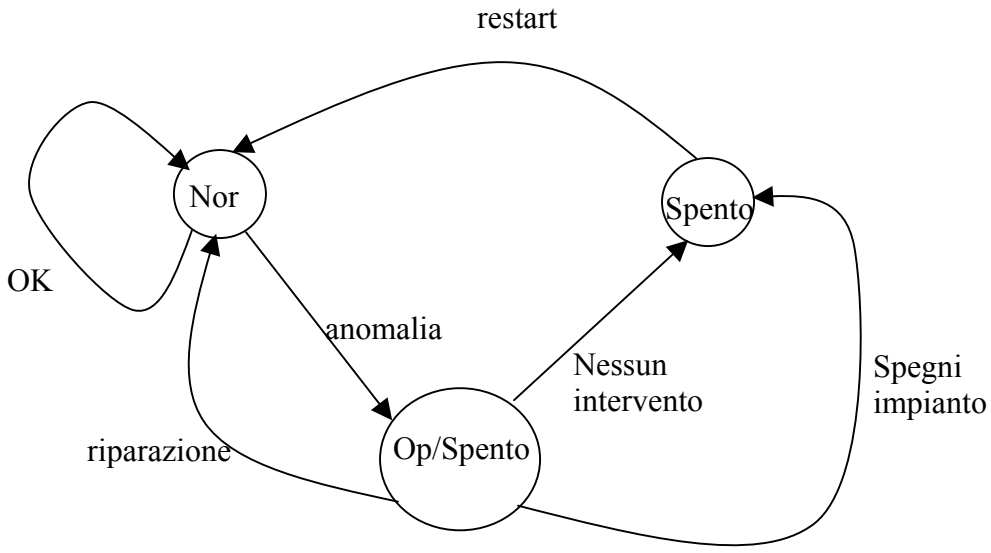
Esercizio 1

Parte a



Parte b

L'automa a stati finiti "naturale" che descrive il funzionamento dell'impianto risulta nondeterministico. Una versione deterministica ad esso equivalente è la seguente:



Esercizio 2

$$\forall h\,(\forall t\,(0 \le t \le h) \rightarrow \neg Nor(t)) \rightarrow h < x$$

Esercizio 3

L'affermazione è vera. Infatti, se un algoritmo deterministico risolve un problema P in tempo polinomiale, allora P è sicuramente decidibile, poiché la computazione dell'algoritmo termina sempre.

Informatica Teorica
Prima Semiunità

Primo compitino - 22 Novembre 2001

Esercizio 1 (11 punti)

Parte a (3 punti)

Si definisca in maniera formale la condizione di accettazione di una stringa da parte di un automa a pila che riconosca a pila vuota *e* a stato finale: la stringa viene accettata se e solo se, al termine della scansione della stringa, la pila dell'automa è vuota (non contiene *nessun* carattere) e lo stato dell'organo di controllo appartiene all'insieme di stati di accettazione.

Parte b (4 punti)

Si dica, giustificando brevemente la risposta, se gli automi a pila nondeterministici che riconoscono a pila vuota e a stato finale sono più, meno, o ugualmente potenti degli automi nondeterministici che riconoscono solo sulla base dello stato finale.

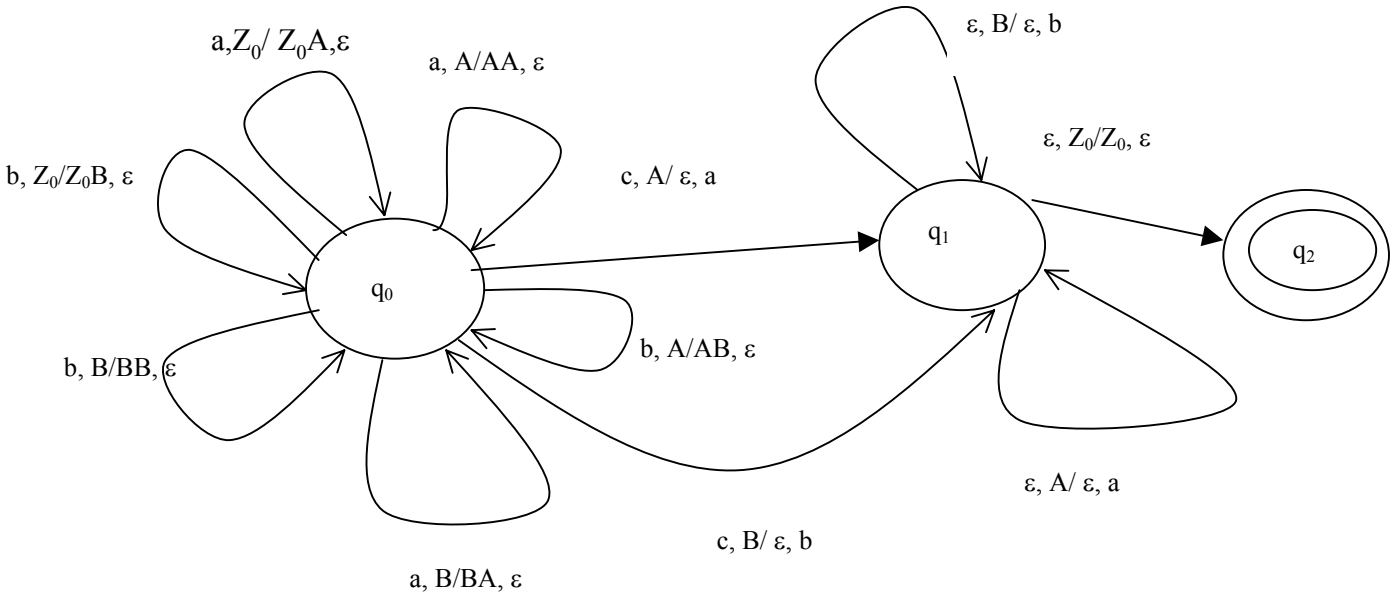
Parte c (4 punti)

Si dica, giustificando brevemente la risposta, se gli automi a pila nondeterministici che riconoscono a pila vuota e a stato finale sono più, meno, o ugualmente potenti degli automi nondeterministici che riconoscono solo a pila vuota (ossia che accettano la stringa di ingresso se e solo se al termine della sua scansione la pila è vuota).

Esercizio 2 (5 punti)

Sia $\tau(x)$, $x \in \{a,b,c\}^*$ la traduzione definita dal traduttore a pila della figura sottostante (**NB**: la figura adotta la convenzione che il carattere più a destra di una stringa sia quello che viene depositato sulla cima della pila mentre quello a sinistra rimane ad esso sottostante). Si costruisca una grammatica che genera il linguaggio definito nel modo seguente:

$L = \{x\$\tau(x), x \in \{a,b,c\}^*\}$



Soluzioni

Esercizio1

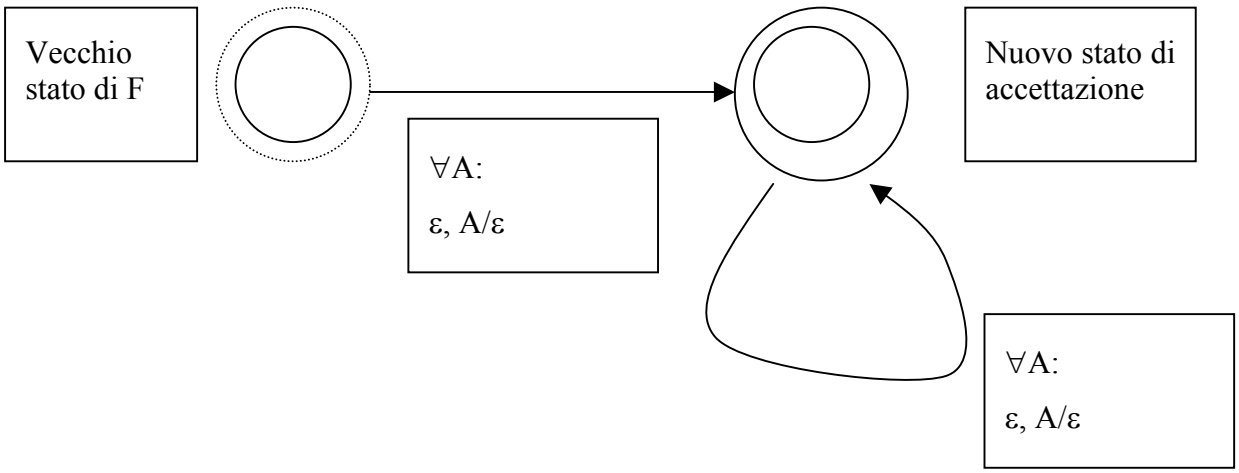
Parte a

Adottando le usuali convenzioni, x viene accettata se e solo se

$c_0 = <q_0, x, Z_0> |---* <q, \varepsilon, \varepsilon> \wedge q \in F$

Parte b

La nuova famiglia di automi è ugualmente potente di quella che riconosce a solo stato finale. Infatti, dato un automa che riconosce a solo stato finale, esso può essere trasformato in un automa che si comporta esattamente come quello originario ma, una volta giunto a uno stato di accettazione di quello originario (che non sarà più di accettazione per il nuovo automa), si porti in un nuovo stato di accetazione mediante una ε–mossa e successivamente svuoti la pila.



Si noti tuttavia che una tale trasformazione potrebbe trasformare un automa originariamente deterministico in uno nondeterministico, poiché gli stati di accettazione non comportano necessariamente l'arresto dell'automa.

Viceversa, dato un automa che riconosce a stato finale e pila vuota, lo si può trasformare, se non lo è già, in primo luogo in un automa che usa il simbolo Z_0 solo come simbolo di "fondo pila" e che quindi lo elimina solo all'ultima mossa (ciò può essere sempre fatto eventualmente introducendo un nuovo simbolo Z_{00} se Z_0 non godesse già di questa proprietà); successivamente si trasforma ogni mossa che cancella il simbolo di fondo pila andando in uno stato di accettazione in una mossa che lascia inalterato il simbolo di pila e porta l'automa in un nuovo unico stato di accettazione in cui l'automa viene bloccato.

Parte c

La nuova famiglia di automi è ugualmente potente anche di quella che riconosce a sola pila vuota. Infatti, mediante costruzioni simili alle precedenti, è sempre possibile costruire un automa che riconosca a stato finale modificando uno che riconosca a pila vuota e viceversa.

Esercizo 2

S → aAa |bAb|
A → aAa| bAb | c$

Informatica Teorica
Prima Semiunità

Secondo compitino – 18 Gennaio 2002

Esercizio 1
Si formalizzi mediante una formula del prim'ordine ***una sola*** delle seguenti regole (nel caso si formalizzino entrambe verrà tenuto conto del punteggio ***minore*** ottenuto!):

Regola 1 (punti 5)
Se un automobilista commette un'infrazione, entro 150 giorni dalla data dell'infrazione gli verrà notificata la contravvenzione, ed entro 30 giorni dalla data della notifica egli provvederà al relativo pagamento.
Si suggerisce di fare uso di predicati del tipo Infr(A, t) per indicare il fatto che l'automobilista A ha commesso un'infrazione nel giorno t. Per semplicità si può assumere che l'automobilista commetta al più un'infrazione.

Regola 2 (punti 9)
Se un automobilista commette un'infrazione e la contravvenzione gli viene notificata entro 150 giorni dalla data dell'infrazione, egli deve pagare la multa entro 30 giorni dalla data della notifica. Qualora non paghi entro 30 giorni egli verrà processato.
Vale lo stesso suggerimento fornito per la regola 1.

Esercizio 2 (punti 4)
Si dica, giustificando brevemente la risposta, se il seguente problema è decidibile:
Siano S1 la sequenza di numeri interi {2, 28, 999, 34, 1278, 7649, 100000} e S2 la sequenza di caratteri {a, a, c, y, e, s, t, u, d, a, b, x, o, p, w}
Dato un generico programma P, codificato in un qualsiasi linguaggio di programmazione (C, C++, Java, Modula-2, ...) stabilire se P, ricevendo S1 in ingresso, produce S2 come risultato.

Esercizio 3
Si dica, spiegandone brevemente le ragioni, se le seguenti affermazioni sono vere o false.
NB1: se, nella spiegazione delle risposte, si fa uso di fatti ben noti (ad esempio enunciati nei libri di testo) non è necessario dimostrare a loro volta tali fatti.
NB2: le funzioni di complessità relative alla RAM si intendono valutate con criterio di costo logaritmico.

1) (1 punto)
Il riconoscimento di un linguaggio non-contestuale deterministico può essere effettuato da una Macchina di Turing a k nastri con complessità temporale $\Theta(n)$.

2) (1 punto)
Il riconoscimento di un linguaggio non-contestuale deterministico può essere effettuato da una Macchina di Turing a nastro singolo con complessità temporale $\Theta(n)$.

3) (2 punti)
Il riconoscimento di un linguaggio non-contestuale deterministico può essere effettuato da una RAM con complessità temporale $\Theta(n)$.

4) (1 punto)
Il riconoscimento di un linguaggio regolare può essere effettuato da una Macchina di Turing a nastro singolo con complessità temporale $\Theta(n)$.

5) (1 punto)
Il riconoscimento di un linguaggio regolare può essere effettuato da una RAM con complessità temporale $\Theta(n)$.

6) (3 punti)

Nessun linguaggio non-contestuale può essere riconosciuto da una Macchina di Turing a k nastri con complessità ***spaziale*** Θ-< di Θ (log(n)).

Soluzioni

Esercizio1

Regola 1

Sia il predicato Infr definito come nel testo dell'esercizio. Similmente siano definiti i predicati Notif(A,t) e Paga(A,t). La regola è allora formalizzata dalla formula seguente:

$\forall A, t ((Infr(A,t) \rightarrow \exists t1, t2 ((Notif(A, t1) \wedge t \leq t1 \leq t + 150) \wedge (Paga(A, t2) \wedge t1 \leq t2 \leq t1 + 30))$

Regola 2

$\forall A, t, t1, t2$

$((Infr(A,t) \wedge Notif(A, t1) \wedge t \leq t1 \leq t + 150 \wedge \forall t2 (t1 \leq t2 \leq t1 + 30 \rightarrow \neg Paga(A, t2))) \rightarrow$

$\exists t3 (Process(A, t3) \wedge t1+30 < t3))$

con l'ovvio significato del nuovo predicato Process.

Esercizio 2

Il problema è indecidibile. Infatti la proprietà di P che si vuole decidere è sicuramente posseduta da alcune funzioni calcolabili definite sull'insieme dei possibili file di Input ma non da tutte. In base al teorema di Rice allora non è possibile stabilire se un generico algoritmo (comunque codificato) calcoli una funzione dotata della proprietà suddetta.

Esercizio 3

1) Vero: la MT a k nastri può simulare l'automa a pila deterministico senza alterarne al complessità ed è noto che ogni automa a pila deterministico ha complessità lineare.

2) Falso: è stato dimostrato che il linguaggio wcw^R richiede una complessità almeno $\Theta (n^2)$ per una MT a nastro singolo.

3) Falso: il linguaggio wcw^R richiede almeno Θ (n.log(n)).

4) Vero: la MT a nastro singolo può comportarsi come un automa regolare senza alterarne la complessità (ameno dell'ultima mossa)

5) Vero: anche la RAM può simulare facilmente l'automa a stati finiti con una quantità di memoria finita e quindi in modo tale che ogni singola mossa costi un valore costante anche a criterio di costo logaritmico.

6) Falso: un linguaggio regolare può essere riconosciuto usando una quantità di memoria costante.

Informatica Teorica

Prima Semiunità

Appello del 23 Gennaio 2002

Esercizio 1
Un compilatore a più passate legge il file sorgente dall'input e produce il codice oggetto in un file di output dopo aver eseguito un certo numero di passate, che producono file contenenti elaborazioni intermedie: l'input di ogni passata è l'output della precedente.

a) **(punti 3)**
Si formalizzi il concetto di *automa a pila deterministico a k passate*: in analogia al compilatore a più passate tale automa riceve una stringa sul nastro di ingresso e la traduce come un normale trasduttore a pila deterministico; terminata la traduzione l'automa opera nuovamente sulla stringa traduzione da esso stesso prodotta e così per k volte.

b) **(punti 5)**
Si dica, giustificando brevemente la risposta, se l'automa a k passate, usato come *riconoscitore* di linguaggi (esso esegue k-1 traduzioni e alla k-esima passata opera esculsivamente come riconoscitore accettando la stringa originaria se e solo se la stringa prodotta dalla (k-1)-esima passata è accettata dalla k-esima passata) è più potente o no del normale automa a pila deterministico (a una passata).

Esercizio 2
Si formalizzi mediante una formula del prim'ordine ***almeno la prima*** delle seguenti regole:

Regola 1 **(punti 4)** **[3]**
Uno studente non può frequentare un corso insegnato da un suo parente.

Regola 2 **(ulteriori punti 5)** **[4]**
La precedente regola 1 viene *modificata e precisata* nel modo seguente:
Uno studente non può frequentare un corso insegnato da un suo parente con grado di parentela < 7.
Il grado di parentela è definito nel modo seguente:
Per qualsiasi grado di parentela la relazione di parentela di grado g è simmetrica.
Se l'individuo x è genitore dell'individuo y, allora x e y sono parenti di grado 1. Se x è parente di grado g di y e x è genitore di z allora y è parente di grado g + 1 di z .

Esercizio 3 **(punti 9)** **[8]**
Si consideri il seguente problema:
Dato un programma P che calcola la funzione f_P: Z → Z (Z essendo l'insieme dei numeri interi) stabilire se $f_P(x) > 0$ per tutti gli x per cui è definita.
Si dica, giustificando brevemente la risposta, se il problema suddetto è

- decidibile,
- semidecidibile ma non decidibile,
- neanche semidecidibile.

Esercizio 4

Si fornisca una maggiorazione –a meno della relazione Θ - della relazione tra le complessità temporali dei seguenti modelli di calcolo qualora si usi uno di essi per simulare l'altro:
Si spieghino brevemente le ragioni delle risposte fornite.

1) **(punti 4)** **[3]**
Una Macchina di Turing (MT) a k nastri simula una MT a k+1 nastri.

2) **(punti 5)** **[4]**
Una Macchina di Turing a nastro singolo simula una RAM la cui complessità temporale sia misurata a criterio di costo logaritmico.

Soluzioni

Esercizio 1

a)

Gli elementi che costituiscono l'automa (insieme di stato, alfabeti, transizioni, ...) sono identici a quelli del normale traduttore deterministico, con la sola differenza che gli alfabeti I e O devono coincidere.

L'accettazione e la traduzione di una stringa da parte dell'automa a più passate sono definite nel modo seguente:

$x \in L(T) \wedge \tau(x) = z \leftrightarrow$

$<x, q_0, Z_0, \varepsilon> \vdash\!\!-\!\!* <\varepsilon, q, \gamma, z_1> \wedge q \in F \wedge$

$< z_1, q_0, Z_0, \varepsilon> \vdash\!\!-\!\!* <\varepsilon, q, \gamma, z_2> \wedge q \in F \wedge$

...

$< z_{k-1}, q_0, Z_0, \varepsilon> \vdash\!\!-\!\!* <\varepsilon, q, \gamma, z_k> \wedge q \in F \wedge z_k = z$

NB: i simboli q e γ utilizzati nella definizione non denotano necessariamente lo stesso valore.

b)

Gli automi a k passate sono più potenti di quelli normali (a una passata). Infatti è facile riconoscere il linguaggio $\{a^n b^n c^n\}$ in due passate: la prima passata verifica che il numero di a sia uguale al numero di b e copia i b e i c sul nastro di uscita. La seconda passata verifica che il numero di b sia uguale al numero di c.

Esercizio2

Regola 1

Si faccia uso dei predicati seguenti:

Freq (stud, c) (Lo studente stud frequenta il corso c)

Par (x, y) (La persona x è parente della persona y; la relazione definita dal predicato è simmetrica)

Ins(prof, c) (Il professor prof insegna il corso c)

Ovviamente studenti e professori sono persone.

La seguente formula formalizza allora la regola 1:

$\forall$stud, prof, c ((Ins(prof, c) $\wedge$ Par(prof, stud) $\rightarrow \neg$ Freq(Stud, c))

Regola 2

Rispetto alla regola 1 occorre modificare il predicato Par: Par(x, y, g) indica che x e y sono parenti di grado g.

La semantica di Par(x, y, g) viene definita dalle seguenti regole aggiuntive, dopo aver introdotto il nuovo predicato elementare Gen(x, y) (x è genitore di y):

$\forall$x, y, z, g (

Gen(x,y) $\rightarrow$ Par(x, y, 1) $\wedge$

Par(x,y,g) $\rightarrow$ Par(y, x, g) $\wedge$

Par(x,y,g) $\wedge$ Gen(x,z) $\rightarrow$ Par(y, z, g+1)

)

La precedente regola 1 viene poi modificata nel modo ovvio.

Esercizio 3

Il problema non è decidibile. Infatti l'insieme delle funzioni per cui vale la proprietà specificata evidentemente non è né vuoto né l'insieme universo. Si applica perciò il teorema di Rice.

E' però semidecidibile il complemento del problema. Infatti, con tipica tecnica di enumerazione diagonale è possibile individuare un x tale per cui $f_P(x) \leq 0$, se un tale x esiste.

Di conseguenza il problema originario non può essere semidecidibile a sua volta, altrimenti sarebbe anche decidibile.

Esercizio 4

7) E' noto che una (MT) a 1 nastro può simulare una MT a k nastri con complessità $T_1(n) \leq T_k^2(n)$ (la tecnica dimostrativa è quella usuale: la memoria della macchina a k nastri viene rappresentata nell'unico nastro e a questo punto simulare la singola mossa della macchina originaria costa al più una completa scansione dell'intera memoria, che ha una dimensione $\leq T_k(n)$).

Il risultato vale quindi a maggior ragione anche quando il numero complessivo di nastri diminuisce di un'unità.

8) In prima istanza si può osservare che una MT a nastro singolo può simulare una MT a k nastri in tempo quadratico rispetto alla macchina originale. Questa a sua volta può simulare una RAM in tempo quadratico rispetto ad essa; e quindi una MT a nastro singolo può simulare una RAM in tempo limitato dalla quarta potenza della complessità della RAM valuatata a criterio di costo logaritmico.

Tuttavia, si può anche applicare direttamente la tecnica dimostrativa uitlizzata per calcolare la relazione di complessità tra la RAM e la MT a k nastri e constatare che utilizzando il nastro singolo della MT occorre solamente effettuare un numero superiore di "passate" sul nastro per simulare una mossa della RAM. Ma tale numero, essendo comunque limitato, non altera l'ordine di grandezza della relazione di complessità che rimane perciò quadratico.

Informatica Teorica
Prima Semiunità

Appello del 21 Febbraio 2002

Esercizio 1

Si formalizzino mediante formule del prim'ordine i seguenti requisiti riguardanti il traffico di un aeroporto:

A) Due velivoli non devono mai trovarsi a distanza tra loro inferiore a k metri.

B) Un aereo può atterrare o decollare solo dopo almeno z minuti dall'ultimo decollo o atterraggio.

Facoltativamente

Si modifichi il requisito A) nel modo seguente:

Abis) Due velivoli non devono mai trovarsi a distanza tra loro inferiore a k metri, a meno che non si trovino nella zona di parcheggio. In tal caso devono rispettare una distanza di almeno h metri ($h < k$).

Suggerimento

Si può (non si deve!) far uso di predicati del tipo Pos(V,P) per denotare il fatto che il velivolo V si trovi nel punto P dell'aeroporto.

Esercizio 2

Si consideri il seguente linguaggio:

$L = \{a^n b^n c^n \mid n \geq 1\}$

Si descriva a grandi linee ma in maniera sufficientemente precisa come una macchina di Turing a *nastro singolo* possa riconoscere L e se ne analizzi la complessità in termini della relazione Θ.

NB: la soluzione proposta sarà considerata tanto migliore quanto minore sarà la sua complessità.

Soluzione dell'Esercizio 1

Si definiscano i seguenti predicati:

Pos(V,P): il velivolo V si trova nel punto P dell'aeroporto

Att(V,t): il velivolo V atterra all'istante t

Dec(V,t): il velivolo V decolla all'istante t

Dist(P1, P2): funzione che indica la distanza tra il punto P1 e il punto P2 dell'aeroporto

I requisiti A e B possono allora essere formalizzati nel modo seguente:

A) $\forall V1, V2, P1, P2 ((Pos(V1,P1) \wedge Pos(V2, P2) \wedge (V1 \neq V2)) \rightarrow Dist(P1, P2) \geq k)$

B) $\forall V1, V2, t1, t2 ((Att(V1,t1) \vee Dec(V1,t1)) \wedge (Att(V2,t2) \vee Dec(V2,t2)) \wedge (V1 \neq V2)) \rightarrow |t1 - t2| \geq z)$

Parte facoltativa

Si definisca il seguente ulteriore predicato:

Parch(P): il punto P appartiene all'area di parcheggio

Allora il requisito Abis) può essere formalizzato come segue:

Abis) $\forall V1, V2, P1, P2$

$((Pos(V1,P1) \wedge Pos(V2, P2) \wedge (V1 \neq V2) \wedge (\neg Parch(P1) \vee \neg Parch(P2)) \rightarrow Dist(P1, P2) \geq k)$

$\wedge$

$((Pos(V1,P1) \wedge Pos(V2, P2) \wedge (V1 \neq V2) \wedge (Parch(P1) \wedge Parch(P2)) \rightarrow Dist(P1, P2) \geq h))$

o, equivalentemente:

Abis) $\forall V1, V2, P1, P2$

$((Pos(V1,P1) \wedge Pos(V2, P2) \wedge (V1 \neq V2) \rightarrow$

$((\neg Parch(P1) \vee \neg Parch(P2)) \rightarrow Dist(P1, P2) \geq k)$

$\wedge$

$((Parch(P1) \wedge Parch(P2)) \rightarrow Dist(P1, P2) \geq h)))$

NB.

Al solo scopo della formalizzazione dei requisiti A, Abis, B, di cui sopra è sufficiente usare i predicati suddetti. Qualora si volesse formalizzare un insieme più completo di requisiti relativi al funzionamento del traffico di velivoli in un aeroporto, sarebbe probabilmente più opportuno usare predicati più "ricchi" di informazione, ad esempio, Pos(V, P, t) per stabilire che il velivolo V si trova nel punto P all'istante t.

Soluzione dell'esercizio 2

Una soluzione semplice consiste nello scandire diverse volte il nastro: ad ogni "passata" la macchina conta un a, un b, e un c, ad esempio rimpiazzandoli con un asterisco: se alla fine tutti gli a, b, e c sono stati rimpiazzati da asterischi contemporaneamente la stringa viene accettata. In tal modo si fanno n passate e si ottiene quindi una complessità $\Theta(n^2)$.

Una soluzione più sofisticata consiste invece nel "dividere per 2" il numero di caratteri da contare ad ogni passata. Illustriamo prima come si possa fare ciò nell'ipotesi semplificativa che n sia una potenza di 2. Successivamente rilasseremo questa ipotesi.

Alla prima passata la macchina trasforma la stringa $a^n b^n c^n$ nella stringa $(*a)^{n/2}$ $(*b)^{n/2}$ $(*c)^{n/2}$

Alla seconda passata la stringa viene trasformata in $(***a)^{n/4}$ $(***b)^{n/4}$ $(***c)^{n/4}$

e così via: se alla fine tutte le a, b e c spariscono contemporaneamente la stringa viene accettata.

Il numero di passate è $\Theta(\log(n))$ e quindi la complessità complessiva è $\Theta(n.\log(n))$.

Consideriamo ora il caso che n non sia una potenza di 2.

In tal caso ad ogni passata la macchina deve anche ricordarsi, mediante gli stati, se la divisione per 2 operata ha lasciato un resto o no. Se il confronto tra i 3 esponenti non dà luogo a tre pari o tre dispari la stringa viene immediatamente rigettata. Altrimenti essa viene trasformata nel modo seguente:

Alla prima passata la macchina trasforma la stringa $a^n b^n c^n$ nella stringa

$(*a)^{n/2}$ [$](*b)^{n/2}$ [$] $(*c)^{n/2}$[$]

Dove [$] indica la presenza o meno del nuovo simbolo $ a seconda che n fosse dispari o pari, rispettivamente. Si noti che per decidere se scrivere o no il simbolo $ la macchina può necessitare di un passo indietro.

Alla seconda passata la stringa viene trasformata in

$(***a)^{n/4}$ [$$][$]$(***b)^{n/4}$ [$$][$] $(***c)^{n/4}$[$$][$]

Con il medesimo significato delle parentesi quadre.

In generale alla k-esima passata la stringa sarà trasformata in

$(*^{2^{k-1}}a)^{n/(2^k)}$ [$^{2^{k-1}}] [$^{2^{k-2}}] …[$]$(*^{2^{k-1}}b)^{n/(2^k)}$ [$^{2^{k-1}}] …[$]$(*^{2^{k-1}}c)^{n/(2^k)}$ [$^{2^{k-1}}] ...[$]

Ad ogni passata può essere necessario un numero di passi indietro $\Theta(2^k) = \Theta(n)$ per registrare il resto della divisione mediante i caratteri $. In ogni caso ogni passata richiede non più di $\Theta(n)$ passi. Il numero di passate rimane $\Theta(\log(n))$ e quindi la complessità complessiva è ancora $\Theta(n.\log(n))$.

Informatica Teorica

Prima Semiunità

Appello del 19 Giugno 2002

Esercizio 1 (20 punti)

Si consideri la seguente grammatica:
<Exp> → x | (<Exp> + <Exp>) | (<Exp> - <Exp>) | (<Exp> * <Exp>) | <Cost>*<Exp>
<Cost> → 0 | 1 | 2 | … 9
Si consideri poi il seguente problema:
Date due espressioni E1 ed E2 generate dalla grammatica suddetta, stabilire se $\forall x(E1 = E2)$ (interpretando x come variabile reale e i simboli di operazione come le usuali operazioni aritmetiche).
Si dica, giustificando brevemente la risposta, se il problema suddetto è

- decidibile,
- semidecidibile ma non decidibile,
- neanche semidecidibile.

Esercizio 2 (15 punti)

Si costruisca un automa a pila che riconosca le stringhe generate dalla grammatica di cui all'esercizio 1.

Esercizio 2 (versione semplificata: 10 punti)

Si costruisca un automa a pila che riconosca le stringhe generate dalla grammatica di cui all'esercizio 1, privata delle regole
<Exp> → <Cost>*<Exp>
<Cost> → 0 | 1 | 2 | … 9.

Soluzioni

Esercizio 1

La grammatica in questione genera espressioni del tipo

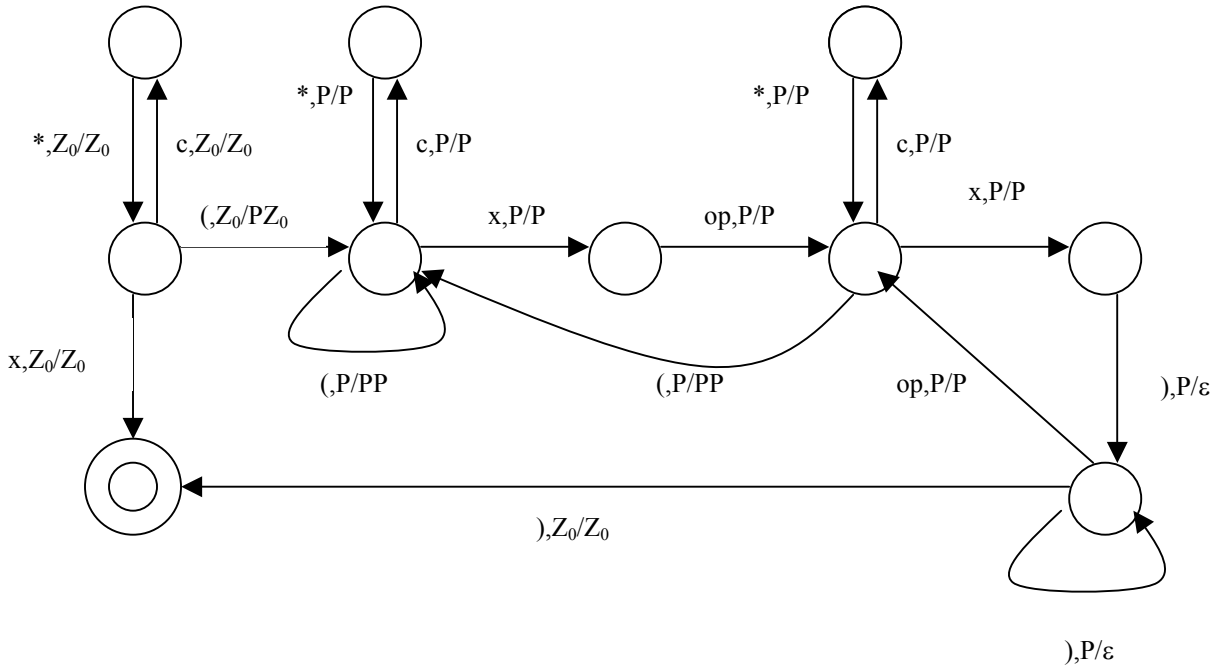
c_1* x*(c_2*x – c_3*x)*(…) - … + (…), dove ci denotano coefficienti numerici compresi tra 0 e 9.

Sviluppando (anche algoritmicamente) tali espressioni, si ottengono polinomi a coefficienti interi nella variabile x: ad esempio $5x^5 + 18x^4 – 45x$ (**NB**: non si ottengono *tutti* i possibili polinomi in x: ad esempio x + 5 e 17x non sono ottenibili a partire dalle espressioni generate dalla grammatica).

In generale, due polinomi a coefficienti interi sono identici (cioè hanno uguale valore per ogni x) se e solo se i coefficienti di tutti i monomi che li compongono coincidono per ogni grado dei monomi. Di conseguenza il problema è decidibile.

Esercizio 2 (versione completa)

Un automa a pila che riconosca le espressioni generate dalla grammatica dell'esercizio 1 è il seguente.



Legenda: op è un'abbreviazione per l'insieme di etichette {+,-,*}

c è un'abbreviazione per l'insieme di etichette {0, 1, …9}

Informatica Teorica
Prima Semiunità

Appello del 4 Luglio 2002

Esercizio 1

L'operazione merge(L1,L2) è così definita per due linguaggi L1, L2:

merge(L1,L2) = $\{z \mid z = x_1.y_1.x_2.y_2. \ldots x_n y_n$, con $x_1. x_2. \ldots x_n = x \in L1$, $y_1.y_2. \ldots y_n = y \in L2\}$

NB: le singole stringhe $x_i\, y_j$, potrebbero essere vuote.

Ad esempio, se L1 = $\{a^n b^n \mid n \geq 1\}$, L2 = $\{c^n d^n \mid n \geq 1\}$, merge(L1,L2) = {acbd, abcd, acacddbb, ccddaabb, …}; non appartiene invece a merge(L1,L2) la stringa bacd.

Si illustri brevemente come sia possibile, date due grammatiche G1 e G2 che generino rispettivamente L1 e L2, costruirne una che generi merge(L1,L2). Per semplicità si puoò assumere che gli alfabeti di L1 e L2 siano disgiunti.

Si fornisca una grammatica che generi merge($\{a^n b^n\}$, $\{c^n d^n\}$).

Attenzione: non è detto che la grammatica che genera merge(L1,L2) appartenga alla stessa categoria di G1 e G2.

Esercizio 2

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

1. Condizione *necessaria* perché un problema P sia indecidibile è che esso sia formalizzabile come il problema del calcolo di una funzione f_P il cui dominio sia infinito.
2. Condizione *sufficiente* perché un problema P sia indecidibile è che esso sia formalizzabile come il problema del calcolo di una funzione f_P il cui dominio sia infinito.

Soluzione Esercizio 1

Sia *ad esempio* {a,b} l'alfabeto di L1 e {c, d} l'alfabeto di L2. Si aggiungano ai loro alfabeti gli insiemi {A,B} e {C,D} rispettivamente. Per una generica stringa x di {a,b}* si definisca X come la stringa ottenuta da x rimpiazzando minuscole con maiuscole. Similmente per y x di {c,d}*.

Date G1 e G2 si costruisca una grammatica che generi tutte le stringhe del tipo XY$, $ essendo un nuovo nonterminale.

Si aggiungano le regole

AC → CA, BC → CB, AD → DA, BD → DB

(esse permettono di scambiare caratteri di L1 con caratteri di L2 senza alterare il loro ordine relativo);

Infine si aggiungano le regole A$ → a, C → $c, …, $ → ε

Per trasformare le stringhe in sequenze di caratteri terminali.

Soluzione Esercizio 2

a. L'affermazione 1 è vera: infatti se un problema fosse formalizzabile come il calcolo di una f_P a dominio finito, la f_P sarebbe sicuramente calcolabile (ad esempio mediante una tabella che ne definisca i valori). Ovviamente ciò non significa che un algoritmo per il calcolo della f_P sia disponibile.

b. L'affermazione 2 è falsa: infatti, ad esempio, la funzione f(x) = x+2 definta sui numeri naturali ha dominio infinito ma è ovviamente calcolabile.

Informatica Teorica
Prima Semiunità

Appello del 18 Luglio 2002

Esercizio 1

Sia A l'alfabeto consistente nei caratteri alfabetici minuscoli (A = {a, b, …, z}). Il linguaggio L contiene tutte le stringhe x di caratteri di A che soddisfano la proprietà seguente:

- Se x inizia con il carattere 'a', allora deve contenere un numero di 'b' uguale al numero di 'z';
- Se x inizia con il carattere 'b', allora la somma del numero di 'c' e del numero di 'h' deve essere pari.

Ad es. appartengono a L le seguenti stringhe: abbzzzb, ace, aaaaaaaaaaaaaaazb, hgsdfahgsfdhaf, xxxxx, bch, bccch, bhh.

Non appartengono a L le seguenti stringhe: abbzzb, aceb, bchc, bhbbbbb.

a) Si costruiscano una grammatica e una macchina astratta a potenza minima che rispettivamente generino e riconoscano L.

b) Si scriva poi una formula del prim'ordine che definisca L. Per semplicità si può fare uso in tale formula delle normale operazioni definite su stringhe e linguaggi (concatenazione, *, operazioni insiemistiche, …) senza formalizzarle esplicitamente.

Esercizio 2

Si considerino i due programmi seguenti:

```
#include <stdio.h>
main()
{
        int  a, b, somma;
        scanf("%d%d", &a, &b);
        somma = a + b;
        printf("La somma di a+b è:\n%d \nArrivederci!\n", somma);
}

#include <stdio.h>
main()
{
        int  a, b, somma;
        scanf("%d%d", &a, &b);
        somma = ( (a + b) / 3)*3;
        printf("La somma di a+b è:\n%d \nArrivederci!\n", somma);
}
```

Si dica, giustificando brevemente la risposta, se il problema di stabilire se i due programmi siano equivalenti è decidibile o no.

Parte facoltativa

Cambierebbe la risposta se le variabili a, b, somma di entrambi i programmi fossero dichiarate **float** invece che **int**? Perché?

Soluzione Esercizio 1

Una grammatica noncontestuale che genera L è la seguente

$S \rightarrow A \mid B \mid C$

$A \rightarrow aH$

$H \rightarrow bHzH \mid zHbH \mid \varepsilon \mid \$H$ (Dove $ denota una qualsiasi carattere di A diverso da 'b' e 'z')

$B \rightarrow bK$

$K \rightarrow cJ \mid hJ \mid \mid \varepsilon \mid £K$ (Dove £ denota una qualsiasi carattere di A diverso da 'c' e 'h')

$J \rightarrow cK \mid hK \mid \mid \varepsilon \mid £J$

$C \rightarrow \varepsilon \mid \&D$ (Dove & denota una qualsiasi carattere di A diverso da 'a' e 'b')

$D \rightarrow \varepsilon \mid @D$ (Dove @ denota una qualsiasi carattere di A)

Un automa a pila deterministico può riconoscere L nel modo seguente:

Se la stringa di ingresso è vuota o non comincia per 'a' o 'b', viene comunque accettata;

Se la stringa comincia per 'a' tutti i caratteri diversi da 'b' e 'z' vengono ignorati –con semplici autoanelli dell'automa– ; la pila viene usata per contare e decantare i 'b' e 'z' in eccesso rispettivamente. Ovviamente la stringa viene accettata solo se al termine della scansione la pila non segnala eccesso né di 'b' né di 'z'.

Se la stringa comincia per 'b' un semplice riconoscimento a stati finiti verifica la parità di occorrenze di 'c' e 'h' (identificando tra loro i due caratteri).

Una formula del prim'ordine che definisca le stringhe di L è la seguente:

$s \in L \leftrightarrow (\forall x$

$(s= a.x \rightarrow \#x_b = \#x_z) \wedge$

$(s= b.x \rightarrow \exists n (\#x_c + \#x_h = n \wedge \exists m (n = 2.m))$

$)$

Dove l'operazione # "numero di occorrenze del carattere a in x" è definita, come al solito, nel modo seguente (dove b denota un qualsiasi carattere diverso da 'a'):

$$x = \varepsilon \rightarrow \#x_a = 0 \wedge x = ay \rightarrow \#x_a = \#y_a + 1 \wedge x = by \rightarrow \#x_a = \#y_a$$

Soluzione Esercizio 2

I due programmi non sono equivalenti. Infatti la divisione intera e successiva moltiplicazione per 3 producono un risultato diverso da a+b se questo valore non è multiplo di 3. Essendo il problema risolto, ossia deciso, esso è necessariamente anche decidibile.

Se le variabili in gioco fossero reali invece che intere la soluzione del problema sarebbe più delicata. Infatti la divisione per 3 potrebbe produrre approssimazioni nel risultato che determinerebbero diversi output per certi valori di ingresso e quindi la non equivalenza dei programmi. Tuttavia alcuni compilatori sofisticati potrebbero automaticamente semplificare la divisione e successiva moltiplicazione per 3 rendendo i due programmi del tutto equivalenti. Ad ogni modo il problema rimane comune decidibile, poiché esso fa riferimento a due programmi fissati e quindi la risposta alla domanda sulla loro equivalenza non può che essere SI o NO.

Informatica Teorica

Prima Semiunità

Appello del 19 Settembre 2002

Esercizio 1
Si dica, giustificando brevemente la risposta, se esistono automi appartenenti alle famiglie seguenti che riconoscano il linguaggio L così definito:
$L = \{a^n |, n \text{ numero primo} > 12\}$
Le famiglie di automi da prendere in considerazione sono:

1) Automi a stati finiti deterministici
2) Automi a stati finiti non deterministici
3) Macchine di Turing deterministiche
4) Macchine di Turing non deterministiche

NB: l'esercizio è considerato risolto in maniera soddisfacente se viene fornita la risposta corretta (ben giustificata) ad almeno 2 dei casi richiesti.

Esercizio 2

Si scriva una formula del prim'ordine che formalizzi la proprietà che due numeri siano primi tra loro.

Esercizio 3
Si dica, giustificando brevemente la risposta, l'affermazione seguente è vera o falsa:
Esistono linguaggi riconoscibili da automi a stati finiti in tempo Θ-< (ovvero con ordine di grandezza della complessità temporale inferiore) di quanto necessario con una macchina di Turing a nastro singolo.

Soluzioni

Esercizio 1

1) e 2): NO: Si supponga il contrario e si applichi il pumping Lemma. Si prenda un numero n primo sufficientemente grande (> della cardinalità di Q): per il pumping lemma esiste un k, con 1 <= k <= n, tale che a^{n+rk} appartiene ad L per ogni r. Ma, sicuramente, per ogni k esiste un r tale che n + rk non sia primo.

3) e 4): SI: il problema della primalità di un numero intero è notoriamente decidibile e quindi esistono macchine di Turing, sia deterministiche che non deterministiche che lo risolvono.

NB: il fatto che n debba essere > 12 è evidentemente inessenziale.

Esercizio2

Primi_fra_loro(n,m) $\leftrightarrow$ ($\neg$($\exists$ x, y , z(x > 1 $\wedge$ n = x*y $\wedge$ m = x*z)))

Esercizio 3

L'affermazione è falsa. Infatti una macchina di Turing a nastro singolo può simulare un automa a stati finiti comportandosi, mediante il suo organo di controllo, in maniera identica all'automa. Al più sarà necessaria una mossa finale per riconoscere il fatto che la stringa di ingresso è terminata. Ma ciò non altera l'ordine di grandezza della complessità.

Informatica Teorica

Prima Semiunità

Appello del 22 Gennaio 2003

Esercizio 1

Si scriva una formula del prim'ordine che definisca il linguaggio costituito da tutte le stringhe definite sull'alfabeto {a,b,c} che iniziano con la lettera 'a' e terminano con la lettera 'b' oppure contengono almeno 3 'c' consecutive.

Esercizio 2

Si consideri il seguente problema:
Dato un programma P che calcola la funzione f_P: N → N (N essendo l'insieme dei numeri naturali) stabilire se $f_P(x) \neq 10$ per tutti gli x per cui è definita.
Si dica, giustificando brevemente la risposta, se il problema suddetto è

- decidibile,
- semidecidibile ma non decidibile,
- neanche semidecidibile.

Soluzioni

Esercizio 1

$\forall$ x(

$x \in L \leftrightarrow ((\exists y\ (y \in V_T^*) \wedge (x = ayb)) \vee (\exists\ z, y\ (z \in V_T^*) \wedge (y \in V_T^*) \wedge (x = zcccy)))$

)

Esercizio 2

Il problema non è decidibile. Infatti l'insieme delle funzioni per cui vale la proprietà specificata evidentemente non è né vuoto né l'insieme universo. Si applica perciò il teorema di Rice.

E' però semidecidibile il complemento del problema. Infatti, con tipica tecnica di enumerazione diagonale è possibile individuare un x tale per cui f_P (x) = 0, se un tale x esiste.

Di conseguenza il problema originario non può essere semidecidibile a sua volta, altrimenti sarebbe anche decidibile.

Informatica Teorica

Prima Semiunità

Appello del 19 Febbraio 2003

Esercizio 1

Si dica, giustificando brevemente le risposte, quali delle seguenti affermazioni sono vere:

1. Il complemento dell'intersezione di due linguaggi non contestuali è un linguaggio non contestuale.
2. Il complemento dell'intersezione di due linguaggi non contestuali deterministici è un linguaggio non contestuale.
3. Il complemento dell'intersezione di due linguaggi non contestuali deterministici è un linguaggio non contestuale deterministico.
4. Il complemento dell'intersezione di due linguaggi regolari deterministici è un linguaggio decidibile. (NB: i linguaggi decidibili sono definiti come gli insiemi decidibili o ricorsivi).

Esercizio 2

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

1 La classe dei linguaggi riconoscibili da Macchine di Turing *deterministiche* che terminano sempre la propria computazione è chiusa rispetto al complemento.

2 La classe dei linguaggi riconoscibili da Macchine di Turing *nondeterministiche* che terminano sempre la propria computazione è chiusa rispetto al complemento.

Soluzione dell'Esercizio 1

1. Il complemento dell'intersezione di due linguaggi non contestuali è un linguaggio non contestuale.
   Falsa: I linguaggi non contestuali non sono chiusi rispetto al complemento: esiste quindi un L non contestuale tale che il suo complemento L^ non sia non contestuale. Intersecando L con l'insieme universo, che è un linguaggio non contestuale, otteniamo ancora L , il cui complemento L^ non è non contestuale.
2. Il complemento dell'intersezione di due linguaggi non contestuali deterministici è un linguaggio non contestuale.
   Vera: : I linguaggi non contestuali deterministici sono chiusi rispetto al complemento. Quindi ilcomplemento dell'intersezione di due linguaggi non contestuali deterministici è l'unione dei loro complementi che sono due linguaggi non contestuali deterministici; l'unione di linguaggi non contestuali è non contestuale.
3. Il complemento dell'intersezione di due linguaggi non contestuali deterministici è un linguaggio non contestuale deterministico.
   Falsa: Sulla base del ragionamento precedente il complemento dell'intersezione di due linguaggi non contestuali deterministici è l'unione di due linguaggi non contestuali deterministici, che però potrebbe dar luogo a un linguaggio non contestuale ma non deterministico.
4. Il complemento dell'intersezione di due linguaggi regolari deterministici è un linguaggio decidibile.
   Vera: Grazie alla chiusura dei linguaggi regolari rispetto a tutte le operazioni insiemistiche, il linguaggio ottenuto è regolare e quindi ovviamente decidibile.

Soluzione dell'esercizio 2

1. **Vera**: una MT di Turing che riconosce un linguaggio L terminando sempre la propria computazione può essere sempre trasformata in una MT che riconosce L^ semplicemente scambiando accettazione con non accettazione all'ultima mossa che comporta l'halt della MT. NB: la classe dei linguaggi riconosciuti da Macchine di Turing *deterministiche* che terminano sempre la propria computazione coincide con la classe dei linguaggi ricorsivi.
2. **Vera:** Una MT nondeterministica che termina sempre la propria computazione può essere trasformata in una MT deterministica che gode della stessa proprietà: basta applicare la normale costruzione di trasformazione da nondeterministica a deterministica e constatare che se tutte le computazioni della macchina originaria terminano sempre, così accade alle computazione della corrispondente macchina deterministica. A questo punto si applica la proprietà precedente.

Informatica Teorica

Primo compitino – 28 Aprile 2003

Si consideri il linguaggio L definito sull'alfabeto {a, b, c, f} e consistente di tutte e sole le stringhe appartenenti ad {a,b,c}*.f contenenti un numero di b uguale al numero di c se contengono almeno un a; contenenti invece un numero di b uguale al numero di c + 10 se non vi compare neanche un a.

Esercizio 1 (punti 5/15)
Si fornisca una formula del prim'ordine che caratterizzi tutte e sole le stringhe di L.

Esercizio 2 (punti 6/15)

Si descriva, senza necessariamente entrare nei minimi dettagli, 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 6/15)

Si costruisca una grammatica G che generi L. Contrariamente all'esercizio precedente, G non deve essere necessariamente a potenza minima. Sono invece auspicate per G le –pur soggettive- qualità della semplicità e naturalezza nella sua costruzione e comprensione.

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.

Soluzioni

Esercizio1 (con leggere "abbreviazioni" rispetto alla pura sintassi del prim'ordine)

$x \in L \leftrightarrow x \in \{a,b,c\}^*.f \wedge$

$(((\exists y, z\ (x = y.a.z) \rightarrow \#(x,b) = \#(x,c)) \wedge ((\neg\ (\exists y, z\ (x = y.a.z)) \rightarrow \#(x,b) = \#(x,c) + 10))))$

dove la funzione #(x,b) indica il numero di occorrenze del carattere b nella stringa x ed è a sua volta definita dalla formula

$$x = \varepsilon \rightarrow (\#(x,b) = 0) \wedge (x = by \rightarrow \#(x,b) = \#(y,b) + 1) \wedge ((x = ay \vee x = cy) \rightarrow \#(x,b) = \#(y,b))$$

Si omette in quanto ben nota la formula che formalizza l'appartenenza $x \in \{a,b,c\}^*.f$.

Esercizio2

Il linguaggio L può essere riconosciuto da un automa a pila deterministico A che operi, a grandi linee, nel modo seguente:

A "tiene conto" mediante la propria pila della differenza tra il numero di b e di c letti (la pila contiene Z0 se e solo se tale differenza è 0). Se durante la lettura incontra almeno un a, quando legge f verifica semplicemente che in pila ci sia Z0; se invece non ha letto alcun a, usa la memoria a stati finiti per verificare che in pila sia rimasto un eccesso di esattamente 10 b.

Esercizio 3

Essendo L riconosciuto da un automa a pila (per di più deterministico), esiste sicuramente una grammatica non contestuale che genera L. Tuttavia è forse più semplice generare L con la seguente grammatica anche se di tipo generale.

S → Hf

H→ AM | MBBBBBBBBBB

M → BCM | ε

AB → BA

BA → AB

CB → BC

BC → CB

AC → CA

CA → AC

A → a

B → b

C → c

Informatica Teorica
Prima Semiunità

Appello del 18 Giugno 2003

Esercizio 1

Si formalizzi mediante un'opportuna formula del primo ordine un apparecchio A definito informalmente come segue:

A può ricevere da due sensori diversi un segnale che può essere il carattere 'a' o il carattere 'b'. I sensori possono anche non trasmettere alcun segnale. Se A riceve dai due sensori lo stesso segnale 'a' o 'b' entro 2 secondi (i segnali possono essere anche contemporanei), allora dopo due secondi dalla ricezione del secondo segnale A emette in uscita il segnale 'c'. Dopo l'emissione di un 'c', incluso l'istante stesso in cui viene emesso 'c', A "perde memoria" degli eventi precedenti, ossia ricomincia il proprio funzionamento come se non avesse ancora ricevuto alcun segnale in ingresso. L'emissione di 'c' avviene solo nelle condizioni specificate.

Alternativamente

Si definisca a grandi linee ma con sufficiente precisione un'opportuna macchina astratta che modelli l'apparato definito nella parte a dell'esercizio. Si assuma in tal caso che il tempo di A sia discreto e che la macchina astratta esegua una transizione in maniera sincrona ogni secondo.

Esercizio 2

Si dica, giustificando brevemente la risposta, quale delle seguenti affermazioni sono vere e quali false:
"Sia P un problema formalizzato come una funzione f_P con dominio e condominio l'insieme N dei numeri naturali. Sia f_P non calcolabile. Se il dominio di f_P viene ristretto a un sottoinsieme finito di N allora:

1. f_P diventa certamente calcolabile
2. f_P rimane certamente non calcolabile
3. f_P potrebbe diventare calcolabile ma potrebbe anche rimanere incalcolabile".

Soluzioni

Esercizio 1

Formalizzaione mediante formula logica

Si introducano i seguenti predicati:

in_1(x, t): viene ricevuto il segnale *x* al tempo *t* sul sensore 1, $x \in \{a, b\}$

in_2(x, t): viene ricevuto il segnale *x* al tempo *t* sul sensore 2

out_c(t): viene emesso il segnale *c* al tempo *t*.

Il funzionamento di A è allora descritto dalla formula seguente:

$$\forall t(out\_c(t) \leftrightarrow (\exists t_1, t_2(in\_1(x, t_1) \wedge (in\_2(y, t_2) \wedge |t_1 - t_2| \leq 2 \wedge (x = y) \wedge (t - \max(t_1, t_2) = 2) \\ \wedge \\ \forall t_3((\min(t_1, t_2) \leq t_3 \leq t) \rightarrow \neg out\_c(t_3))$$

Soluzione alternativa

La formalizzazione mediante macchina astratta dell'apparecchio A è concettualmente semplice ma richiede un notevole numero di dettagli. Date le caratteristiche di A la machina astratta può essere anche un automa a stati finiti. In tal caso i suoi elementi essenziali potrebbero essere i seguenti.

1 L'alfabeto di ingresso deve descrivere le seguenti condizioni:

   a. nessun segnale su entrambi i sensori
   b. segnale a sul sensore 1 e nessun segnale sul sensore 2
   c. segnale a sul sensore 1 e segnale b sul sensore 2
   d. segnale a sul sensore 1 e segnale a sul sensore 2
   e. ….

2 L'alfabeto di uscita formalizza l'emissione di c e la non emissione di alcun segnale (più eventuali altri segnali non considerati qui)

3 A si trova inizialmente in uno stato q0 in cui rimane finché non riceve alcun segnale da ambo i sensori (ingresso a. della lista degli ingressi)

4 In qualsiasi stato (tranne qA0 e qA1, specificati in questo paragrafo) se A riceve due segnali uguali da ambo i sensori si porta in uno stato di attesa qA0. Da qA0 passa a qA1 dopo un'unità di tempo (transizione obbligata qualsiasi sia l'ingresso) e da qA1 passa obbligatoriamente in q0 emettendo c.

5 Da q0 se riceve un a sul sensore 1 e nessun segnale sul sensore 2 si porta in uno stato qan da cui "conta" mediante opportune transizioni gli istanti di tempo successivi: se dopo due di tali transizioni non sono stati ricevuti segnali torna in q0; se entro due transizioni è ricevuto un a sul sensore 2 si porta in qA0 da cui procede come sopra; se viene ricevuto un b sul sensore 1 si porta in un nuovo stato che "ricordi" il precedente ricevimento di a e l'attuale di b, e così via per tutte le –numerose- possibili combinazioni di ricevimento di segnali.

6 In ogni caso, le diversi possibili situazioni da memorizzare sono finite, anche se varie e numerose.

NB.

Una formulazione più compatta si sarebbe potuta ottenere con una macchina di Turing. In ogni caso molto più astratta risulta la formulazione mediante formula logica.

Esercizio 2

E' vera l'affermazione 1 (e sono quindi false le altre). Infatti, una funzione definita su un dominio finito è sempre calcolabile essendo descrivibile mediante una tabella finita. Esiste ovviamente sempre una Macchina di Turing che "calcola" una tabella finita di valori: essa altro non è che … la codifica di un array.

Informatica Teorica

Prima Semiunità

Appello dell'1 Luglio 2003

Esercizio 1

Parte a (punti 5)

Si consideri il seguente problema: dati 3 generici programmi P1, P2, P3, che calcolano, rispettivamente, le 3 funzioni f1, f2, f3, definite sul dominio dei numeri naturali, N, risulta $f1(x) = f2(x) + f3(x)\ \forall x \in N$?

Si dica, giustificando brevemente la risposta, se il problema suddetto è decidibile o no.

NB. P1, P2, P3 sono *dati* del problema, non sono fissati a priori.

Parte b (punti 5)

Che cosa cambia se P2 e P3 sono fissati a priori, ossia sono due programmi noti e non più variabili da caso a caso e, dato invece P1, si vuole sapere se la funzione f1 da esso calcolata è costante e identicamente uguale a f2 + f3 (ossia $\exists\ k \in N\ (f1(x) = k, \forall x \in N)$, e $f1(x) = f2(x) + f3(x)\ \forall x \in N$)?

Esercizio 2 (punti 8)

Si descriva brevemente ma con sufficiente precisione un semplice algoritmo per il calcolo della funzione $2^{(n!)}$.

Si valuti poi l'ordine di grandezza, secondo la relazione Θ, della complessità spaziale e temporale dell'esecuzione dell'algoritmo mediante una RAM, sia a criterio di costo costante che a criterio di costo logaritmico, **in funzione della lunghezza del dato di ingresso**, assumendo che esso sia codificato in binario.

NB.

1. L'algoritmo non deve essere necessariamente ottimale relativamente alla complessità di esecuzione.
2. Non è necessaria una codifica dettagliata dell'algoritmo.
3. E' possibile usare le operazioni MULT a DIV della RAM con i relativi valori di costo, anche se è ben noto che tali valori sono in realtà ottimistici.

Esercizio 3 (punti 9)

Si formalizzi mediante un'opportuna formula del primo ordine un apparecchio A definito informalmente come segue:

A può ricevere ad ogni secondo da un sensore un segnale che può essere il carattere 'a' o il carattere 'b'. Il sensore però potrebbe anche non trasmettere alcun segnale. Se A riceve dal sensore lo stesso segnale, sia 'a' che 'b', entro un intervallo di k (>= 2) secondi, estremi inclusi, allora dopo due secondi dalla ricezione del secondo segnale, A emette in uscita il segnale 'c'. In caso contrario A non emette alcun segnale. Ad ogni istante la condizione per l'emissione del segnale di uscita viene ricalcolata indipendentemente da quanto fatto in precedenza, ossia c viene emesso se e solo se 2 secondi prima è stata verificata la condizione seguente: nell'intervallo costituito dai k secondi precedenti l'istante corrente, estremi inclusi ed incluso l'istante corrente, si sia ricevuto un segnale uguale a quello dell'istante corrente **(NB**: il termine "l'istante corrente" è riferito al momento della valutazione della condizione, non al momento dell'eventuale emissione di 'c'): ad esempio, per k = 3, se viene ricevuta in ingresso la sequenza "aab-ab-a" viene emessa in uscita la sequenza "- - - c - - c c - c" dove il simbolo '–' indica l'assenza di segnale.

(**NB** k è la lunghezza dell'intervallo, che quindi consiste di k+1 istanti perché gli estremi dell'intervallo sono inclusi.)

Esercizio 4 (punti 8)

Si definisca un'opportuna macchina astratta che modelli l'apparato definito nell'esercizio 3, assumendo k = 2.

Suggerimento

Una possibile macchina astratta potrebbe consistere in due "automi in cascata", essendo l'alfabeto di uscita del primo coincidente con l'alfabeto di ingresso del secondo. Il primo automa verifica in ogni istante se è soddisfatta la condizione richiesta per l'emissione **futura** del segnale 'c'. In caso positivo invia **immediatamente** un opportuno segnale al secondo. Il secondo altro non fa che emettere il segnale 'c' con due secondi di ritardo rispetto al momento in cui riceve il segnale dal primo automa.

Ovviamente sono possibili e accettabili anche soluzioni diverse da quanto proposto.

Soluzioni

Esercizio 1

Parte a

Il problema è indecidibile, in virtù del classico Teorema di Rice: un suo caso particolare è quello in cui f2 e f3 siano fissate (ad esempio f2(x) = x, f3(x) = 5.x). In tal caso il problema si riduce a stabilire se un generico programma P calcola la funzione f(x) = 6.x; {f} non è né l'insieme vuoto né l'insieme universo; ergo …

Parte b

In questo caso il problema diventa decidibile o meno a seconda dei valori di P2 e P3. Infatti, se la funzione f2 + f3 non è una funzione costante, l'insieme delle funzioni calcolate dai possibili valori di P1 che siano costanti e identicamente uguali a f2(x) + f3(x) è l'insieme vuoto e allora il problema è decidibile. Altrimenti tale insieme non è vuoto (è l'insieme delle funzioni f1 tali che f1(x) = f2(x) + f3(x) = H, $\forall x \in N$, per un certo valore di H) e il problema rimane perciò indecidibile.

Esercizio 2

Un semplice algoritmo consiste in un primo ciclo che, letto n, calcola il valore m = n!; successivamente, un ulteriore ciclo in cui viene fatto variare l'indice i da 1 a m calcola 2^m mediante successive moltiplicazioni per 2.

NB: si potrebbe ottimizzare tale ciclo mediante successivi elevamenti al quadrato della variabile locale x, inizializzata a 2; tuttavia se m non fosse una potenza di 2, occorrerebbe poi una fase finale di ulteriori moltiplicazioni per 2 che nel caso pessimo sarebbero dello stesso ordine di grandezza di m (in un albero bilanciato il numero di foglie è proporzionale al numero dei nodi interni).

A criterio di costo costante la complessità spaziale dell'algoritmo è $\Theta(1)$ perché è sufficiente un numero finito di celle per la sua esecuzione.

La complessità temporale, calcolata in funzione del valore del dato di ingresso n, è $\Theta(n)+ \Theta(m) = \Theta(n!)$. Tenuto poi conto che la lunghezza della rappresentazione di n in binario, detta ll, è $\Theta(\log(n))$, la complessità temporale espressa in funzione di ll è $\Theta(2^{ll}!)$.

Con il criterio di costo logaritmico la complessità spaziale è Θ del logaritmo del risultato, ossia, $\Theta(\log(2^{n!}))$, ossia $\Theta(n!)$ e $\Theta(2^{ll}!)$.

La complessità temporale è determinata dalla ripetizione m volte di un ciclo la cui singola esecuzione costa in maniera proporzionale alla complessità spaziale, ossia al contenuto della cella destinata a contenere il risultato finale. Essa è perciò $\Theta(m.\log(2^{n!}))$, ossia $\Theta((n!)^2)$ e $\Theta((2^{ll}!)^2)$

Esercizio 3

Si introducano i seguenti predicati:

in(x, t): viene ricevuto dal sensore il segnale *x* al tempo *t*, $x \in \{a, b, -\}$. '–' indica l'assenza di segnale.

out_c(t): viene emesso il segnale *c* al tempo *t*.

Il funzionamento di A è allora descritto dalla formula seguente:

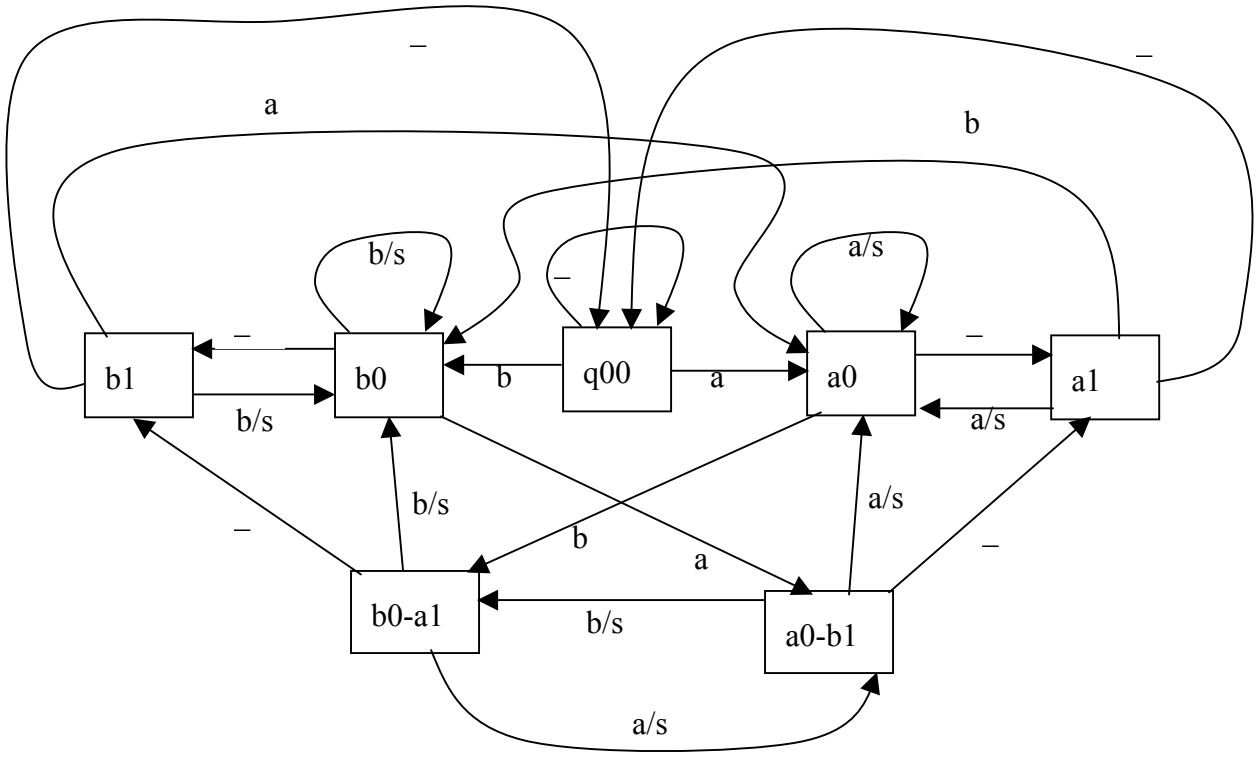
$$\forall t(out\_c(t) \leftrightarrow (\exists t_1, t_2 (in(x, t_1) \wedge in(y, t_2) \wedge |t_1 - t_2| \leq k \wedge (x = y) \wedge (x \neq -) \wedge (t_1 \neq t_2) \wedge (t - \max(t_1, t_2) = 2)))$$

)

Esercizio 4

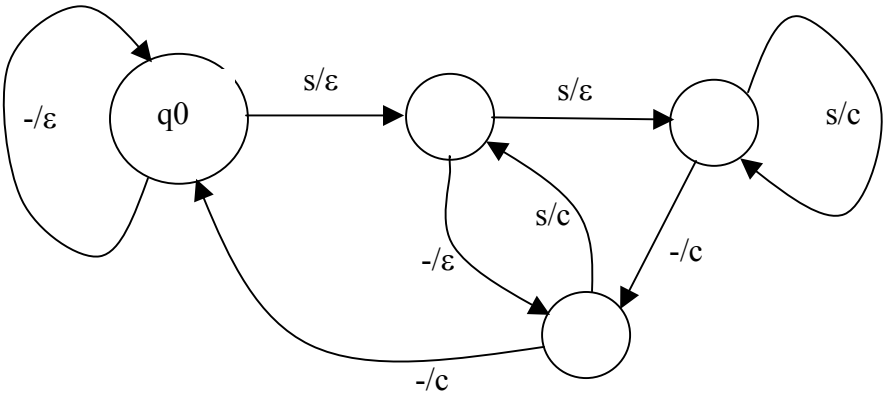
La coppia di automi seguenti opera nel modo indicato nel suggerimento. Diverse altre soluzioni sono possibili.

Primo automa



Legenda: s indica il segnale che viene inviato al secondo automa; dove non è indicata alcuna uscita si sottintende la stringa nulla, ossia appunto l'assenza di segnale.

Secondo automa



Informatica Teorica

Prima Semiunità

Appello dell'17 Luglio 2003

Esercizio 1 (punti 8)

Si specifichi, mediante una formula del prim'ordine un apparato che emette un segnale luminoso istantaneo tutte e sole le volte che un bottone è stato tenuto premuto esattamente e continuativamente per k secondi. La pressione del bottone per meno di k secondi viene ignorata. Se il bottone rimane premuto per più di k secondi, dopo avere emesso il segnale il conteggio ricomincia da capo.

Esercizio 2 (punti 6 senza parte facoltativa, 9 con la parte facoltativa)

Con riferimento all'esercizio precedente, si assuma un tempo discreto (la cui unità sia il secondo) e k = 4. Si costruisca una grammatica che generi il linguaggio L di alfabeto {p, n, s}così definito: L contiene tutte le sequenze di caratteri p, n e s tali che p indichi la pressione del bottone, n l'assenza di pressione sul bottone, s l'emissione del segnale e s compaia immediatamente dopo una sequenza di k p consecutive e solo in tali casi.

Parte facoltativa

Due pregi della grammatica richiesta sono:

- L'essere a potenza minima
- L'essere semplice (avere pochi nonterminali e poche produzioni)

Se è possibile fornire una grammatica che li possieda entrambi, meglio, altrimenti si forniscano due diverse grammatiche perseguendo separatamente i due diversi pregi.

Esercizio 3 (punti 8)

Siano A(x) e B(x) due qualsiasi formule del prim'ordine contenenti la variabile libera x (più altre eventuali variabili non libere). La frase "A(x) è decidibile" è un'abbreviazione per "Esiste un algoritmo in grado di decidere se, dato un valore di x, A(x) è vera o falsa".

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

Comunque presi A(x) e B(x)

1. Se A(x) è decidibile e B(x) è decidibile $A(x) \wedge B(x)$ è decidibile.
2. Se A(x) è indecidibile e B(x) è decidibile $A(x) \vee B(x)$ è decidibile.
3. Se A(x) è indecidibile e B(x) è indecidibile $A(x) \wedge B(x)$ è indecidibile.
4. Se A(x) è indecidibile e B(x) è decidibile $A(x) \vee B(x)$ è indecidibile.
5. Se A(x) è indecidibile $\forall x A(x)$ è indecidibile.
6. Se A(x) è indecidibile $\forall x A(x)$ è decidibile.
7. Se A(x) è indecidibile e B(x) è indecidibile $\forall x (A(x) \wedge B(x))$ è indecidibile.
8. Se A(x) è indecidibile e B(x) è indecidibile $\forall x (A(x)) \wedge \forall y (B(y))$ è indecidibile.

Esercizio 4 (punti 8)

Si descriva con sufficiente precisione, ma senza necessariamente specificare ogni dettaglio, come una Macchina di Turing a k nastri possa riconoscere il linguaggio $L = \{ww, w \in \{a,b\}^*\}$ analizzandole la complessità spaziale e temporale (a meno dell'ordine di grandezza determinato dalla Θ–equivalenza). Sono preferite soluzioni che minimizzino la complessità spaziale.

Soluzioni

Esercizio 1

Si definisca con il predicato PB(t) la pressione del bottone all'istante t; con S(t), l'emissione del segnale all'istante t. La specifica è allora formalizzata dalla formula seguente:

$$\forall t(S(t) \leftrightarrow (\forall t_1((t-k \le t_1 \le t) \rightarrow PB(t_1)) \wedge \forall t_2((t-k < t_2 < t) \rightarrow \neg S(t_2)))$$

E' anche opportuno specificare l'esistenza di un istante iniziale t_0 prima del quale il bottone non è mai stato premuto e nessun segnale è mai stato emesso.

Esercizio 2

Una semplice grammatica noncontestuale è la seguente:

S → AS | BS | A | B

A → pppps

B → n | pn | ppn | pppn

Una grammatica regolare è invece la seguente:

S → pA | nS | p | n

A → pB | nS

B → pC | nS

C → pD | nS

D → sS | s

Esercizio 3

1. Vera
2. Falsa: se B fosse falsa, la decidibilità di $A(x) \vee B(x)$ si ridurrebbe alla decidibilità di $A(x)$.
3. Falsa: se B fosse $\neg A$, $A \wedge B \equiv$ False e quindi decidibile
4. Falsa: se B fosse sempre vera, anche $A(x) \vee B(x)$ lo sarebbe
5. Falsa: una formula chiusa è sempre o vera o falsa, quindi decidibile, anche se non è detto che si sappia il suo valore di verità.
6. Vera, per quanto asserito al punto 5.
7. e 8. sono false in base alle stesse considerazioni del punto 5.

Esercizio 4

Seguendo la stessa traccia che permette di riconoscere il linguaggio $\{wcw^R, w \in \{a,b\}^*\}$, si può memorizzare in un nastro di memoria il valore n della lunghezza della stringa di ingresso, codificandolo in binario; indi lo si divide per 2. A questo punto si memorizza in un altro nastro di memoria con il valore i inizializzato a 1, sempre codificato in binario. Si verifica l'uguaglianza dei caratteri in posizione i e n/2 + i e si procede facendo variare i fino a n/2.

La complessità spaziale risulta perciò $\Theta(\log(n))$. Quella temporale risulta $\Theta(n^2.\log(n))$: dopo l'inizializzazione, sono infatti necessari n/2 confronti; ognuno dei quali richiede n/2 passi;ognuno dei quali a sua volta comporta l'aggiornamento del contatore memorizzato in binario e quindi di lunghezza logaritmica rispetto ad n.