Informatica Teorica

Prima prova in itinere - 5 Maggio 2005

Esercizio 1 (5/13 punti)

Si consideri il linguaggio:

$L \subseteq \{a,b,c\}^*$ costituito dalle stringhe in cui il numero di a sia uguale al numero di b + 5 e il numero di c sia pari.

Si costruisca una macchina astratta che riconosca L. Tra le diverse macchine astratte che riconoscono L sono preferite macchine "a potenza minima" ossia appartenenti alla categoria di automi a minor potenza riconoscitiva possibile.

Esercizio 2 (4/13 punti)

E' noto che le due seguenti definizioni di grammatica regolare sono equivalenti, ossia generano la stessa classe di linguaggi:

1) $\forall \alpha \rightarrow \beta \in P, |\alpha| = 1, \beta \in V_N.V_T \cup V_T$
2) $\forall \alpha \rightarrow \beta \in P, |\alpha| = 1, \beta \in V_T.V_N \cup V_T$

La seguente ulteriore modifica della definizione è anch'essa equivalente alle precedenti?

3) $\forall \alpha \rightarrow \beta \in P, |\alpha| = 1, \beta \in V_T.V_N \cup V_T \cup V_N.V_T$

Giustificare brevemente la risposta.

Esercizio 3 (5/13 punti)

Si definisca in maniera matematicamente precisa la seguente versione arricchita di automa a pila.

L'automa, oltre a tutte le caratteristiche dei normali automi a pila (nondeterministici), può anche, ad ogni mossa, esaminare il simbolo contenuto nel fondo della pila e sostituirlo con un altro simbolo.

Se ne formalizzino anche le regole di funzionamento (per esempio mediante le normali nozioni di configurazione e transizione tra configurazioni) e di riconoscimento di stringhe.

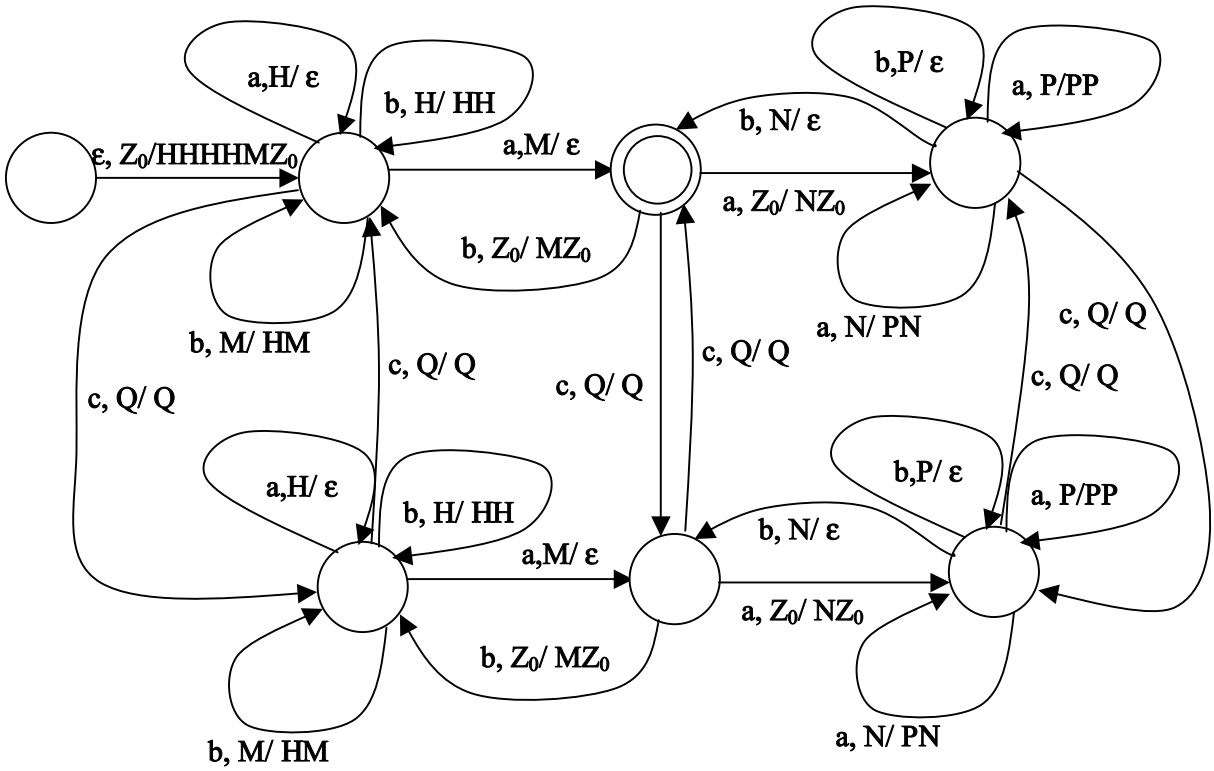
NB: nel caso la precedente definizione informale non risulti sufficientemente precisa in alcuni dettagli, si operino opportune scelte, spiegandone le ragioni, per giungere ad una formalizzazione precisa.

Si dica poi, giustificando brevemente la risposta, se l'automa, così arricchito aumenta la potenza riconoscitiva rispetto agli automi a pila tradizionali. In caso positivo, raggiunge la potenza delle macchine di Turing?

Soluzioni

Esercizio1

Il linguaggio L è riconosciuto dal seguente automa a pila deterministico:



Legenda: il simbolo Q indica un qualsiasi simbolo ∈ Γ

Evidentemente non è possibile riconoscere L con una macchina a stati finiti, a causa della necessità di conteggio illimitato sul numero di a e b.

Esercizio2

No

La grammatica seguente soddisfa la nuova definizione ma genera il linguaggio

$\{a^n b^n \mid n >= 1\}$ che notoriamente non è regolare.

S → aA

A → Sb | b

Esercizio 3

Un automa riconoscitore nondeterministico a pila, che possa accedere e modificare anche il fondo della pila può essere formalizzato mediante una 7-pla
<Q, I, Γ, δ, q_0, Z_0, F >in cui tutti i simboli hanno lo stesso significato di quelli relativi agli automi a pila tradizionali, con l'unica differenza che la funzione δ viene estesa nel modo seguente:

$$\delta : Q \times (I \cup \{\varepsilon\}) \times \Gamma \times \Gamma \to \wp_F(Q \times \Gamma^* \times \Gamma)$$

dove δ (q, i, A, B) = {<q', α, C>} significa che l'automa, trovandosi nello stato q, leggendo i dal nastro di ingresso (oppure ε nel caso di ε –mossa), A dalla cima della pila, B dal fondo della medesima, in maniera nondeterministica tra le diverse terne possibili, si porta in q', scrive α sulla cima della pila al posto di A e C sul fondo al posto di B.
Una configurazione c dell'automa è una 3-pla c = <q, x, γ>, come per l'automa a pila tradizionale; la relazione di transizione è, parzialmente, definita dalle regole seguentise | γ | ≥ 2

< q, i.y, AηH > |-- <q', y, α ηM> se e solo se <q', α, M> ∈ δ (q, i, A, H)

se | γ | = 1

<q, i.y, A> |-- <q', y, αM> se e solo se <q', αM, M> ∈ δ (q, i, A, A)

NB: se <q', αH, M> ∈ δ (q, i, A, A) con H ≠ M, la transizione non è applicabile. Sono però possibili anche altre definizioni: ad esempio che la macchina dia priorità alla riscrittura sulla cima della pila rispetto a quella sul fondo; oppure che comunque la transizione sia possibile solo se il contenuto della pila contiene almeno due caratteri.

(la parte rimanente della definizione della relazione di transizione viene data per scontata).

Infine x è accettata dall'automa se e solo se

< q_0, x, Z_0 > |-- <q', ε, γ>, per qualche γ, con q' ∈ F, come nel caso dell'automa a pila tradizionale.

La potenza dell'automa non aumenta rispetto all'automa a pila tradizionale, poiché l'informazione aggiuntiva che esso può trattare è un'informazione finita (il simbolo in fondo alla pila) e può quindi essere memorizzata attraverso gli stati finiti dell'organo di controllo.

Informatica Teorica

Seconda prova in itinere - 29 Giugno 2005, Sezione Cremona+Como

Esercizio 1 (punti 6/17-esimi)

Si *specifichi,* mediante opportune pre- e post-condizioni il seguente requisito per un frammento di programma FP:

Sia dato un array *a* di *n* < NMAX interi positivi; gli *n* interi sono memorizzati nelle prime posizioni di *a*, a partire da *a*[0]; dopo di essi si trova il valore convenzionale 0 usato come "terminatore".

FP deve produrre un nuovo array *b*, che sia il risultato dell'ordinamento di *a* in ordine crescente e dell'eliminazione di eventuali ripetizioni. FP usa *a* come variabile "read-only". Anche la sequenza dei valori di *b* deve essere terminata da uno 0.

Suggerimento: si consiglia di introdurre la funzione *lungh(z)* definita su array di NMAX interi che fornisce il numero di valori di *z* diversi da 0 che precedono il primo 0 (in caso di assenza di uno 0, *lungh(z)* = -1 per convenzione).

Esercizio 2 (punti 7/17-esimi)

Si consideri il programma seguente, codificato in un ipotetico linguaggio ispirato al C.

```
int g(int p); int f(char t);

main ()

{  char a[20]; int i, z;

   scanf(a); z = 0

   while (i = 0; g(i) != 20; i++) do z = z + f(a[i]);

   printf(z);

}
```

Si assuma che f e g siano due sottoprogrammi "esterni" la cui terminazione sia garantita per ogni valore dei parametri di ingresso.

Si dica, motivando brevemente la risposta, se è decidibile il problema di stabilire se il programma main di cui sopra terminerà o meno la sua esecuzione in corrispondenza di un generico –non fissato!- valore del dato di input a.

Esercizio 3 (punti 7/17-esimi)

Si consideri una sequenza S di elementi che contengano informazione codificabile in un numero limitato a priori di celle di memoria. Si assuma che S si trovi già memorizzata nella memoria di una macchina RAM secondo una delle due tradizionali tecniche:

a) come array di celle consecutive

b) come lista di elementi collegati tra loro mediante puntatori

Si valuti la complessità asintotica sia spaziale che temporale, a meno della relazione Θ, di un algoritmo di ricerca sequenziale nei due casi, facendo uso sia del criterio di costo costante che del criterio di costo logaritmico; si spieghi brevemente come si giunge ai risultati proposti, senza bisogno peraltro di codificare in dettaglio l'algoritmo. Si indichi con precisione il parametro rispetto al quale viene misurata la dimensione dei dati di ingresso, in funzione della quale viene fornita la funzione di complessità.

Esercizio 3-bis, Facoltativo (punti 4/17-esimi, valido solo se preceduto da soluzione corretta della parte obbligatoria)

Come cambiano le valutazioni di complessità di cui sopra immaginando che il medesimo algoritmo venga eseguito da una Macchina di Turing?

Soluzioni

Esercizio 1

In primo luogo si definisca la funzione lungh(z) mediante una formula del tipo

$$lungh(z) = n \leftrightarrow ((0 \le n < NMAX \wedge z[n]=0 \wedge \neg\exists i(0 \le i < n \wedge z[i] = 0)) \vee (n = -1 \wedge \neg\exists i(0 \le i < NMAX \wedge z[i] = 0)))$$

A questo punto il requisito richiesto può essere specificato dalla seguente coppia per-post-condizione:s

$$\{\exists n\,(lungh\,(a) = n \quad \wedge \forall i\,(0 \le i < n \rightarrow a[i] > 0))\}$$
$$FP$$
$$\{\exists m\,(lungh\,(b) = m \quad \wedge$$
$$\forall i\,(0 \le i < m-1 \rightarrow b[i] < b[i+1]) \quad \wedge$$
$$\forall i\,(0 \le i < lungh\,(a) \rightarrow \exists j\,((0 \le j < m) \wedge (a[i] = b[j]))) \quad \wedge$$
$$\forall j\,(0 \le j < m \rightarrow \exists i\,((0 \le i < lungh\,(a)) \wedge (a[i] = b[j])))\}$$

Esercizio 2

La risposta è positiva

Premessa. Perché il quesito sia significativo occorre ovviamente assumere una macchina astratta che presupponga un dominio totale di dati infinito: in questo caso, il tipo **int** deve essere l'insieme matematico degli interi. Altrimenti, come noto, la macchina astratta diventa automaticamente un automa a stati finiti e quindi la sua terminazione decidibile.

Il ***dominio*** dei dati di ingresso *del programma in oggetto* è però ***finito***, essendo l'insieme degli array di 20 caratteri. Per ognuno dei possibili valori di ingresso la risposta al quesito è semplicemente un sì o un no; esiste quindi una macchina di Turing che fornisce tale risposta, per ogni valore del dato di ingresso a; di conseguenza esiste un numero finito di macchine di Turing, che, opportunamente combinate tra loro, forniscono la risposta corretta per ogni valore del dato di ingresso a; la combinazione di tali macchine può dar luogo ad un'unica macchina che, sul dominio finito dei possibili valori di a, fornisce l'insieme finito delle risposte corrette. Ovviamente, ciò non garantisce la conoscenza di tale macchina; ne garantisce esclusivamente l'esistenza.

Equivalentemente e più sinteticamente si può osservare che il problema posto (la decidibilità dell'halt del programma rispetto ai diversi valori di ingresso) è formalizzato da una funzione a dominio finito (l'insieme degli array di 20 caratteri) e codominio {T, F}. Tale funzione è quindi riducibile a una tabella finita, e quindi calcolabile.

Si noti anche che la terminazione del programma, essendo garantita la terminazione di una singola esecuzione del corpo del ciclo, dipende solo dalla valutazione della funzione g, che, o non produce mai 20, oppure prima o poi deve produrre il valore 20 per qualche valore di i. Ciò è indipendente dal valore del dato di ingresso a. Quindi la tabella di cui sopra, non dipendendo dai dati di ingresso sarà costituita da tutti T o tutti F.

Esercizio 3

Si indichi con n il numero di elementi della sequenza. Assumiamo per semplicità che ogni elemento contenga informazione codificabile in un numero fissato di byte. In questo caso è del tutto indifferente assumere come parametro di dimensione dei dati in ingresso il numero n di elementi di S o le celle di memoria utilizzate per memorizzare S.

La memorizzazione mediante array richiede una quantità di memoria $\Theta(n)$ sia a criterio di costo costante che a criterio logaritmico. La memorizzazione mediante puntatori invece richiede di memorizzare un intero come puntatore per ogni elemento di S. Essendo il numero di elementi di S illimitato, una quantità di memoria richiesta sarà $\Theta(n)$ a criterio di costo costante ma $\Theta(n \cdot \log(n))$ a criterio logaritmico.

La complessità temporale risulterà invece $\Theta(n)$ a criterio di costo costante e $\Theta(n \cdot \log(n))$ a criterio logaritmico per entrambi i tipi di memorizzazione.

Esercizio 3 bis (parte facoltativa)

Simulare un array mediante un nastro di macchina di Turing richiede ovviamente uno spazio $\Theta(n)$. $\Theta(n)$ risulta pure la complessità temporale di un algoritmo di ricerca sequenziale.

La memorizzazione di un puntatore in un nastro di macchina di Turing richiede la codifica (ad esempio in binario) di un numero i e quindi un numero $\Theta(\log(i))$ di celle. La complessità spaziale risulta perciò $\Theta(n \cdot \log(n))$ e l'accesso a un generico elemento richiede pure $\Theta(n \cdot \log(n))$, rendendo perciò la complessità totale $\Theta(n^2 \cdot \log(n))$

Informatica Teorica

Recupero della prima prova in itinere - 29 Giugno 2005, Sezione Cremona+Como

Esercizio 1 (punti 6+3/13-esimi)

Si scrivano degli automi che riconoscano i seguenti linguaggi (k è fissato e > 1):

$L_1 = \{ x \in \{a,b\}^* \mid x = a^n b^{kn}, n > 0\}$

$L_2 = \{ x \in \{a,b\}^* \mid x = a^n b^{n^k}, n > 0\}$

Parte facoltativa. Si scriva un automa che riconosca il seguente linguaggio (k è fissato e > 1):

$L_3 = \{ x \in \{a,b\}^* \mid x = a^n b^{\log_k(n)}, n > 0\}$

NB: non è necessario che gli automi vengano disegnati completamente. Essi devono però essere descritti con sufficiente dettaglio da poter capire quali sono i loro stati e le loro transizioni.

Esercizio 2 (punti 6/13-esimi)

Si scriva una grammatica che generi il seguente linguaggio:

$L = \{ x \in \{a,b,c\}^* \mid \#x_a = 2n, \#x_b = 4, \#x_c = n, n > 0\}$

dove con la scrittura $\#x_\alpha$ si intende il numero di occorrenze del carattere α nella stringa x.

La grammatica scritta è a potenza minima tra quelle che riconoscono L? Si giustifichi la risposta.

Soluzioni

Esercizio 1

Per riconoscere il linguaggio L_1 è sufficiente un automa a pila deterministico che funziona in questa maniera: per ogni a letta, esso mette k A sulla pila. Quando poi arriva a leggere le b, per ogni b letta cancella una B dalla pila. Se l'automa arriva in fondo alla stringa con solo il carattere Z_0 sulla pila, la stringa viene accettata, altrimenti viene rifiutata.

Per riconoscere il linguaggio L_2, invece, è necessaria una MT. Una MT (a 2 nastri) che riconosce L2 può essere fatta in questa maniera.
Essa innanzi tutto legge le a e memorizza nel primo nastro un numero di A pari al numero di a lette, mette nel secondo nastro altrettante B. Quindi ripete le seguenti operazioni k-1 volte:

- ad ogni B del secondo nastro sostituisce un numero di B pari al numero di A nel primo nastro (creando per ogni B da "moltiplicare" ogni volta lo spazio necessario nel secondo nastro, spostando le celle di n posti a partire dal fondo).

A questo punto legge le b dal nastro di ingresso, cancellando una B dal terzo nastro per ogni b letta. Se alla fine, quando si è letta l'ultima b, il terzo nastro è vuoto, la MT accetta la stringa, portandosi in uno stato finale.

Parte facoltativa.

Anche per riconoscere il linguaggio L_3 è necessaria una MT. La MT (a 2 nastri) può essere fatta in questa maniera.
Essa innanzi tutto legge le a e memorizza nel primo nastro un numero di A pari al numero di a lette. Quindi esegue passate successive sul primo nastro fino a che il primo nastro rimane vuoto, e ad ogni passata fa quanto segue:

- ad ogni k A del primo nastro sostituisce una sola A; se in fondo al nastro rimane un numero di A minore di k, le ultime A vengono eliminate e basta (di fatto ciò corrisponde a dividere il numero di A per k);
- scrive una B nel secondo nastro (a meno che nel primo nastro ci fossero meno di k A).

A questo punto legge le b dal nastro di ingresso, cancellando una B dal terzo nastro per ogni b letta. Se alla fine, quando si è letta l'ultima b, il terzo nastro è vuoto, la MT accetta la stringa, portandosi in uno stato finale.

Una maniera alternativa di risolvere il problema poteva essere di memorizzare sul primo nastro il numero di a lette codificando il numero in base k. $\log_k(n)$ a questo punto è banalmente il numero di cifre necessarie per codificare n in base k (meno uno), e quindi per verificare che ci siano $\log_k(n)$ b basta cancellare una cifra della codifica in base k per ogni b letta.

Esercizio 2

Una grammatica che genera il linguaggio L è la seguente:

$S \rightarrow AACS \mid X$

$X \rightarrow BBBB$

$AC \rightarrow CA$

$CA \rightarrow AC$

$AB \rightarrow BA$

$BA \rightarrow AB$

$CB \rightarrow BC$

$BC \rightarrow CB$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

Tale grammatica non è a potenza minima. Il linguaggio può essere riconosciuto da un automa a pila (deterministico), per cui la grammatica a potenza minima che lo genera è una grammatica non contestuale.

Una possibile grammatica non contestuale che genera il linguaggio L è la seguente:

$S \rightarrow Saac \mid Saca \mid Scaa \mid aSac \mid aSca \mid cSaa \mid aaSc \mid acSa \mid caSa \mid aacS \mid acaS \mid caaS \mid bS_1$

$S_1 \rightarrow S_1aac \mid S_1aca \mid S_1caa \mid aS_1ac \mid aS_1ca \mid cS_1aa \mid aaS_1c \mid acS_1a \mid caS_1a \mid aacS_1 \mid acaS_1 \mid caaS_1 \mid bS_2$

$S_2 \rightarrow S_2aac \mid S_2aca \mid S_2caa \mid aS_2ac \mid aS_2ca \mid cS_2aa \mid aaS_2c \mid acS_2a \mid caS_2a \mid aacS_2 \mid acaS_2 \mid caaS_2 \mid bS_3$

$S_3 \rightarrow S_3aac \mid S_3aca \mid S_3caa \mid aS_3ac \mid aS_3ca \mid cS_3aa \mid aaS_3c \mid acS_3a \mid caS_3a \mid aacS_3 \mid acaS_3 \mid caaS_3 \mid b$

Informatica Teorica

Vecchio ordinamento

Appello del 5 Luglio 2005

Esercizio 1 (punti 15)

Si consideri il seguente linguaggio L_{stud} formato dalle parole *p* così formate: $\{p = N^6V^k \mid N \in 0..9, V \in \{a, b, c, d\}, k <= 30\}$ (cioè composte da 6 cifre, seguite da al massimo 30 caratteri, in cui ogni carattere è o *a*, o *b*, o *c*, o *d*), in cui le 6 cifre rappresentato i numeri di matricola degli studenti immatricolati al Politecnico di Milano nell'anno 2001/2002, e i caratteri seguenti i voti conseguiti negli esami superati (*d* per un voto da 18 a 21, *c* per uno da 22 a 24, *b* per uno da 25 a 27, *a* per uno da 28 a 30).

Si dica, giustificando brevemente la risposta:

a. quale è l'automa a potenza minima in grado di riconoscere il linguaggio L_{stud};

b. se è decidibile il problema di stabilire se ci sono 2 studenti diversi con gli stessi voti;

c. quale è la complessità minima per risolvere il problema di appartenenza di una stringa al linguaggio L_{stud}.

Esercizio 2 (punti 8)

Si specifichi mediante opportune formule del prim'ordine il seguente comportamento di un ipotetico sistema.

Tutte le volte che, in un certo istante, si riceve il segnale *Start*, per i 10 istanti istanti successivi viene emesso un segnale intermittente (con intermittenza di 1 istante) *Blink* (per esempio, se viene ricevuto il segnale *Start* all'istante 6, il segnale *Blink* viene emesso agli istanti 8, 10, 12, 14 e 16). Due segnali di *Start* successivi sono a distanza l'uno dall'altro di almeno 15 istanti di tempo.

Esercizio 3 (punti 7)

Si costruisca una grammatica G che generi il seguente linguaggio L:

$L = \{a^n(bc)^m \mid n, m >= 1, m < n/2\}$

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.

Soluzioni

Esercizio 1

a. Il inguaggio è formato da un numero *finito* di parole, quindi può essere riconosciuto da un automa a stati finiti (addirittura, da un automa senza cicli).

b. Essendo il linguaggio finito, il problema è decidibile.

c. Il problema del riconoscimento di può risolvere con complessità O(1) (cioè costante) in quanto il linguaggio è finito.

Esercizio 2

Codificando la ricezione/emissione di un segnale S all'istante t con il predicato S(t), le formule seguenti specificano il comportamento desiderato:

$$\forall t(Start(t) \rightarrow \forall t_1(1 \leq t_1 \leq 5 \rightarrow Blink(t + 2t_1)))$$
$$\wedge$$
$$\forall t_2, t_3(Start(t_2) \wedge Start(t_3) \rightarrow |t_2 - t_3| \geq 15)$$

Esercizio 3

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

S → aaaabc | ASB

A → aa

B → bc | ε

Informatica Teorica

Prova d'esame - 13 Luglio 2005, Sezione Cremona+Como

Esercizio 1 (punti 8/30-esimi)

Si scriva un automa che riconosca il linguaggio L le cui stringhe sono costruite sull'alfabeto A={0, 1} e sono fatte nella seguente maniera: le stringhe hanno lunghezza dispari; se il primo e l'ultimo carattere della stringa sono entrambi uguali ad '1', il carattere di centro è anch'esso uguale ad '1', altrimenti il carattere di centro è uguale a '0'.

NB: il punteggio massimo verrà assegnato solo se l'automa ideato sarà a potenza riconoscitiva minima tra quelli che riconoscono il linguaggio desiderato.

Esercizio 2 (punti 8/30-esimi)

Descrivere (senza necessariamente codificarla nei dettagli) una macchina RAM che riconosce il linguaggio L descritto nell'esercizio 1, e se ne dia la complessità spaziale e temporale a costo costante e a costo logaritmico.

Esercizio 3 (punti 8/30-esimi)

Scriva una formula logica che descrive un segnale fatto nella seguente maniera: dal momento in cui il segnale viene emesso la prima volta (che potrebbe anche non essere l'istante 0), esso viene emesso ad intervalli che si raddoppiano continuamente. L'intervallo tra i primi due istanti di emissione può essere qualunque.

In altre parole, se la distanza fra la (k-1)-esima emissione e la k-esima emissione è d, la distanza tra la k-esima e la (k+1)-esima emissione è 2d.

Un esempio possibile tra i tanti di evoluzione temporale del segnale è la seguente (vengono indicati gli istanti in cui il segnale viene emesso; si noti che l'intervallo tra le prime due emissioni è pari a 3 istanti di tempo):

2, 5, 11, 23, 47, 95, 191, ...

Esercizio 4 (punti 8/30-esimi)

parte a.

Siano date le seguenti pre- e post- condizioni di un metodo Java che ha un parametro in ingresso *arg* di tipo *String*, ritorna un *int*, e può sollevare un'eccezione *ComputationException*:

Pre: $\text{arg} \neq \textbf{null} \wedge 1 \leq \textbf{length}(\text{arg}) \leq 50 \wedge$
$\forall\, i, j (1 \leq i < j \leq \textbf{length}(\text{arg}) \rightarrow \text{arg}[i] \neq \text{arg}[j])$

Post: $\textbf{result} > 0 \vee \textbf{raise}$ ComputationException

laddove *result* indica il valore ritornato dal metodo (se questo ritorna senza sollevare eccezioni) e *raise ComputationException* indica il fatto che viene sollevata un'eccezione *ComputationException*; inoltre, *length*(*arg*) indica la lunghezza della stringa *arg*, e *arg*[*i*] indica l'i-esimo carattere della stringa.

E' decidibile il problema di stabilire se un'implementazione I soddisfa la specifica data sopra?

Si motivi adeguatamente la risposta.

parte b.

Sia data la seguente implementazione

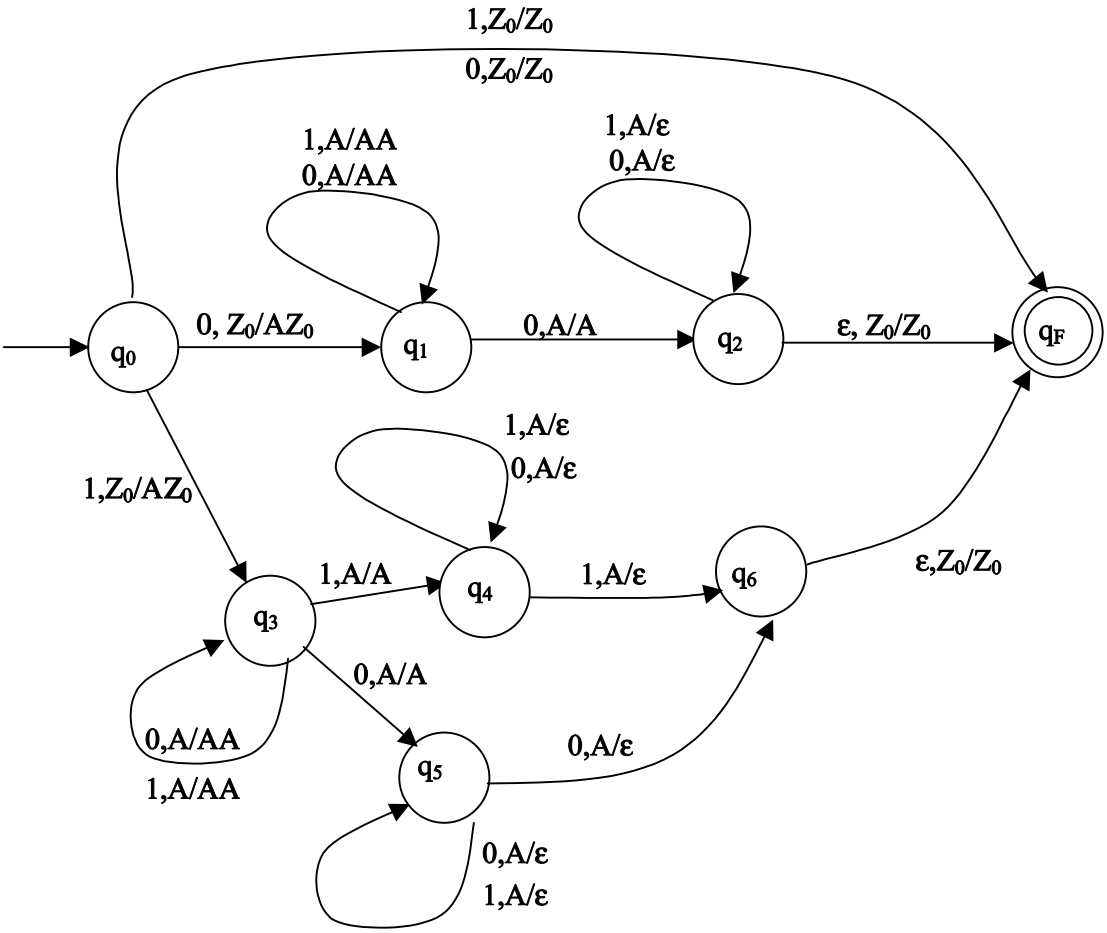
```
int myImpl(String arg) throws ComputationException {
  int sum = 0, i = 1;
  try {
     while(sum < (int)Math.pow(23.5, 4.21)){
       int diff = (int)(arg.charAt(i-1)-arg.charAt(0));
       sum += (int)(diff*Math.pow(4.67, 2.9));
       if (i == arg.length()) i = 1;
       else i++;
     }
  } catch (Exception e) {
    throw new ComputationException();
  }
  return sum;
}
```

E' decidibile il problema di stabilire se *myImpl* soddisfa la specifica descritta sopra?

Soluzioni

Esercizio 1

Il linguaggio L è riconosciuto dal seguente automa a pila nondeterministico:



Evidentemente non è possibile riconoscere L con una macchina a stati finiti, visto che le stringhe di L possono essere arbitrariamente lunghe, nè con un automa a pila deterministico, che non e' in grado di "indovinare" il punto centrale della stringa.

Esercizio 2

Una semplice macchina RAM che riconosce il linguaggio è fatta nella seguente maniera.

Innanzi tutto, essa legge una volta i caratteri in ingresso; ogni carattere letto viene memorizzato, e contemporaneamente si incrementa un contatore che mi dice quanti caratteri ho letto fino a quel momento. Alla fine della lettura, se ho contato un numero di caratteri pari, segnalo su nastro di uscita che la stringa non è accettata. Se invece ho contato un numero di caratteri dispari, vado a leggere il primo carattere, l'ultimo, e quello di mezzo (per recuperare l'offset del quale rispetto al primo carattere basta dividere per 2 il numero di caratteri letti). Se questi sono tutti uguali ad 1, oppure se quello di centro ed almeno uno tra il primo e l'ultimo sono uguali a 0, accetto la stringa, segnalandolo sul nastro di uscita, altrimenti rifiuto.

A costo costante, sia la complessità spaziale che quella temporale sono $\Theta(n)$.

A costo logaritmico, la complessità spaziale è ancora $\Theta(n)$, mentre quella temporale è $\Theta(n \cdot \log(n))$.

Esercizio 3

Indicando, come al solito, col predicato e(t) che il segnale viene emesso al tempo t, una possibile soluzione è:

$$\forall\, t_1, t_2\, (\, t_1 < t_2 \,\wedge\, e(t_1) \,\wedge\, e(t_2) \,\wedge\, \forall\, t'\, (\, t_1 < t' < t_2 \,\rightarrow\, \neg\, e(t')\,)$$

$$\rightarrow$$

$$e(t_2 + 2\,(t_2 - t_1)) \,\wedge\, \forall\, t'\, (\, t_2 < t' < t_2 + 2\,(t_2 - t_1) \,\rightarrow\, \neg\, e(t')\,)\,)$$

Esercizio 4

parte a.

Non è decidibile.

Una maniera per dimostrarlo potrebbe essere tramite il Teorema di Rice: l'insieme delle implementazioni (cioè delle MT, visto che si può tradurre un programma Java in una MT equivalente) che soddisfano la specifica non è certamente l'insieme vuoto (esiste almeno una funzione che soddisfa la specifica), e non è certamente neanche l'insieme universo (è banale pensare ad una funzione che non soddisfa la specifica), quindi il problema di stabilire se una data MT appartiene all'insieme non è decidibile.

Una maniera alternativa potrebbe riducendo il problema della terminazione al problema del soddisfacimento della specifica. Preso un programma qualunque P (facendo attenzione che in P non compaia il parametro *arg*) basta in effetti costruire un metodo siffatto:

```
int Pridotto(String arg) throws ComputationException {
  P
  return 1;
}
```

Tale metodo soddisfa la specifica se e solo se P termina.

Se sapessi risolvere il problema di partenza, saprei anche risolvere il problema della terminazione di un programma qualunque, che è un assurdo.

parte b.

In questo caso, il problema è banalmente decidibile.

Infatti, l'implementazione è in questo caso fissata a *myImpl*, quindi si danno 2 soli casi: o *myImpl* soddisfa le specifiche, oppure non le soddisfa. La risposta è quindi in questo caso binaria, o sì o no, e la MT che risolve il problema o è quella costante uguale ad 1, oppure è quella costante uguale a 0. In entrambi i casi essa è banalmente calcolabile.

Informatica Teorica (Sez. Cremona+Como)

Appello del 13 Settembre 2005

Esercizio 1 (punti 10/30-esimi)

Si scrivano un automa e una grammatica che, rispettivamente, riconosca e generi il linguaggio $L = \{a^n a^m a^m a^n \mid n, m >= 1, n \text{ pari}, m \text{ dispari}\}$.

NB: il punteggio massimo verrà assegnato solo se l'automa ideato sarà a potenza riconoscitiva minima tra quelli che riconoscono il linguaggio desiderato e la grammatica avrà il minimo numero di produzioni.

Esercizio 2 (punti 10/30-esimi)

Si scriva una formula logica che descrive la seguente regola per controllare atterraggi e decolli in un aeroporto:

- Se c'è un aereo in fase di decollo nessun altro aereo può decollare entro 3 minuti dall'inizio del decollo e nessun altro aereo può atterrare entro 5 minuti dall'inizio del decollo dell'aereo precedente;
- Se c'è un aereo in fase di atterraggio nessun altro aereo può atterrare entro 3 minuti dall'inizio dell'atterraggio e nessun altro aereo può decollare entro 5 minuti dall'inizio dell'atterraggio dell'aereo precedente.

Esercizio 3 (punti 10/30-esimi)

Dire, giustificando brevemente la risposta, se la seguente funzione $f: N \rightarrow N$ è:

a) Computabile

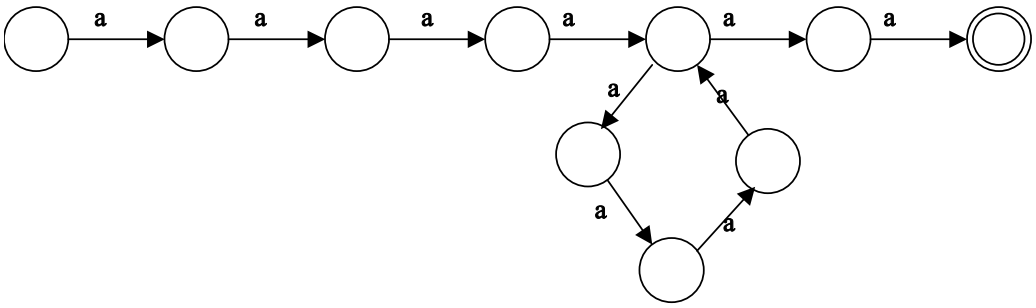
b) Totale.

f(n) = n-esimo numero primo. Per convenzione, si assuma f(0) = 1, e, di conseguenza, f(1) = 2, f(2) = 3, f(3) = 5, …

Soluzioni

Esercizio 1

La definizione di L equivale a L = $\{a^{2n} | \text{ n dispari e } >= 2\}$, ossia L = $\{a^6, a^{10}, a^{14}, a^{18}, \ldots\}$, ossia, detto in altra maniera L = $\{ a^{2(2l+1)} \mid l \geq 1 \} = \{ a^{4l+2} \mid l \geq 1 \}$. Quindi L può essere riconosciuto mediante il seguente semplice automa a stati finiti:



Da esso si può ricavare immediatamente una grammatica regolare equivalente, che può essere "compattata" nella seguente grammatica (non contestuale!) per minimizzare il numero di produzioni:

$S \rightarrow a^4A$

$A \rightarrow a^4A \mid a^2$

Esercizio 2

Si definscano i predicati StartTakeOff (x, t) e StartLand(x, t) per indicare, rispettivamente, che l'aereo x ha iniziato il decollo o l'atterraggio all'istante t.

Il requisito richiesto è allora formalizzato dalla formula seguente:

$$\forall t, x(\mathit{StartTakeOff}(x,t) \rightarrow$$
$$\neg\exists y, t_1 (y \neq x \wedge (t \leq t_1 \leq t+3) \wedge \mathit{StartTakeOff}(y,t_1)) \wedge$$
$$\neg\exists y, t_1 (y \neq x \wedge (t \leq t_1 \leq t+5) \wedge \mathit{StartLand}(y,t_1)))$$
$$\wedge$$
$$\forall t, x(\mathit{StartLand}(x,t) \rightarrow$$
$$\neg\exists y, t_1 (y \neq x \wedge (t \leq t_1 \leq t+3) \wedge \mathit{StartLand}(y,t_1)) \wedge$$
$$\neg\exists y, t_1 (y \neq x \wedge (t \leq t_1 \leq t+5) \wedge \mathit{StartTakeOff}(y,t_1)))$$

Altri predicati e formule, non strettamente necessari ai fini della formalizzazione del requisito ma utili a chiarire il significato dei termini potrebbero definire la fase di atterraggio/decollo dell'aereo x come quella compresa tra i relativi inizio (Start) e fine (End).

Esercizio 3

La funzione f è computabile: essendo infatti decidibile se un numero sia primo o no, dopo aver costruito l'n-esimo numero primo si può ottenere l'n+1-esimo enumerando tutti i numeri ad esso successivi fino ad ottenere il prossimo numero primo.

f è anche totale perché i numeri primi sono infiniti. Quindi, per ogni n, esiste l'n+1-esimo numero primo che verrà individuato dall'algoritmo suddetto.

Informatica Teorica (Sez. Cremona+Como)

Appello del 6 Febbraio 2006

Esercizio 1 (Punti 15)

Si formalizzino mediante formule logiche del prim'ordine le *regole* del gioco "Sudoku".
Informalmente le regole sono le seguenti:
E' dato un quadrato di 9 * 9 caselle, ognuna delle quali deve contenere un numero intero compreso tra 1 e 9. Il quadrato è suddiviso in 9 "sottoquadrati" di 3 * 3 caselle (il primo "copre" le caselle [1..3 * 1..3]; il secondo [4..6* 1..3], ecc.)
Occorre riempire le caselle del quadrato in modo che:

- Ogni riga e ogni colonna contenga tutti i numeri tra 1 e 9
- Ogni "sottoquadrato" contenga tutti i numeri tra 1 e 9.

Facoltativamente (ulteriori punti 1): potrebbe il problema essere generalizzato rispetto alle dimensioni del quadrato? Se sì, come?

Esercizio 2 (punti 7)

Si dica, giustificando brevemente la risposta, se il suddetto problema del Sudoku, ossia stabilire se, assegnati alcuni valori ad alcune caselle esiste un modo di riempire le altre in modo da soddisfare le regole, è decidibile o no.
Come cambia la risposta (se cambia), se il problema è espresso nella sua forma generalizzata?

Esercizio 3 (punti 8)

Si descrivano brevemente (senza necessariamente "codificarli" in dettaglio) algoritmi per il riconoscimento del linguaggio $\{a^n b^n \mid n >= 1\}$ con le seguenti caratteristiche:

- Una macchina di Turing che minimizzi la complessità temporale
- Una macchina di Turing che minimizzi la complessità spaziale
- Una RAM che minimizzi la complessità temporale
- Una RAM che minimizzi la complessità spaziale

E si confrontino le funzioni di complessità così ottenute.

Soluzioni

Esercizio 1

Si formalizzi la tabella del Sudoku mediante un array (si ricordi che, in termini matematici, un array altro non è che una funzione). Le regole del gioco sono allora formalizzate dalla formula seguente:

$$\forall i,j(1 \le i,j \le 9 \rightarrow 1 \le a[i,j] \le 9)$$
$$\wedge$$
$$\forall i,j,k(1 \le i,j,k \le 9 \rightarrow a[i,j] \ne a[i,k]$$
$$\wedge$$
$$\forall i,j,k(1 \le i,j,k \le 9 \rightarrow a[i,j] \ne a[k,j]$$
$$\wedge$$
$$\forall i,j,k,h((1 \le i,j,k,h \le 9 \wedge i/3 = k/3 \wedge j/3 = h/3 \wedge (i \ne k \vee j \ne h)) \rightarrow a[i,j] \ne a[k,h]$$

NB. il simbolo '/' indica la divisione a risultato intero.

La generalizzazione al caso di un quadrato k*k è banale se si ha l'accortezza di scegliere un valore di k che sia un quadrato perfetto. In tal caso k rimpiazza il numero 9 e $\sqrt{k}$ rimpiazza il numero 3 nella formula precedente.

Esercizio 2

Il problema è chiaramente decidibile in quanto è riconducibile del tutto a un problema finito. Infatti, ogni griglia 9 * 9 può essere riempita con numeri da 1 a 9 (rispettando le regole) in un numero finito di modi diversi. (Contare il numero esatto di schemi diversi non è banale, ma è sufficiente capire che esso è finito. Per curiosità, tale numero è stato calcolato essere uguale a 6 670 903 752 021 072 936 960). Pertanto un algoritmo che provi esaustivamente tutte le possibili soluzioni termina sicuramente.

Anche nal caso generalizzato, il problema rimane decidibile. Per ogni *n* fissato, ogni schema di n^2 * n^2 celle ha un numero finito di possibili completamenti (sicuramente meno di n^{2n^4}), per cui possono essere provati esaustivamente tutti.
Il problema è però computazionalmente oneroso, NP-completo per la precisione, come mostrato in: T. Yato, "Complexity and completeness of finding another solution and its application to puzzles". Master's thesis, University of Tokyo, Department of Information Sciences, 2003.

Esercizio 3

1. Macchina di Turing che minimizzi la complessità temporale.
   E' sufficiente "simulare" con la macchina di Turing un automa a pila, usando un nastro di memoria come una pila. La complessità temporale risultante sarebbe $\Theta(n)$, che è facile capire essere il meglio ottenibile.
2. Macchina di Turing che minimizzi la complessità spaziale.
   In questo caso conviene contare in codifica binaria (o equivalentemente, in altra base > 1) il numero di 'a' ricevute e poi decrementare il contatore binario per ogni 'b' ricevuta. La complessità spaziale sarebbe così $\Theta(\log n)$, ottenuta però al prezzo di un peggioramento di un fattore logaritmico della complessità temporale.
3. RAM che minimizzi la complessità temporale.
   Si può replicare la soluzione della macchina di Turing al punto 1 con una RAM, scrivendo un marker in n celle adiacenti. La complessità temporale risultante è la stessa $\Theta(n)$, con criterio di costo costante. Con criterio di costo logaritmico essa sale invece a $\Theta(n \log n)$, dal momento che anche se scrivo un valore costante (il marker) nelle celle adiacenti, ad ogni accesso devo manipolare un puntatore all'elemento corrente, che è un numero maggiorato da n.
4. RAM che minimizzi la complessità spaziale.
   In questo caso conviene utilizzare un singolo contatore intero in una singola cella di memoria. In questo modo il costo a criterio di costo costante è semplicemente $\Theta(1)$ (uso di un numero costante di celle di memoria), mentre a costo logaritmico diventa $\Theta(\log n)$, che è una caratterizzazione del fatto che la RAM di fatto "implementa" una codifica binaria del dato.