Informatica Teorica

Appello del 19 Luglio 2006

Esercizio 1 (Punti 10)
Si consideri il linguaggio seguente:
L = {wcW | w, W ∈ {a, b}*; W essendo una stringa che differisce dalla stringa riflessa, w^R, di w per al più 3 caratteri}
Si scrivano una grammatica e un automa, preferibilmente a potenza minima (ossia appartenenti a una categoria di grammatiche e automi tale che una categoria di minor potenza generativa non possa definire L) che generi e riconosca L, rispettivamente.

Esercizio 2 (Punti 13)
Si consideri il seguente problema: data una *f(x)* qualsiasi (con dominio e codominio l'insieme *N*) di cui sia noto a priori che è computabile, stabilire se esistono un polinomio *P* a coefficienti interi non negativi e un valore $\overline{x} \in N$ tali che $P(\overline{x}) = f(\overline{x})$.
Si dica, giustificando brevemente la risposta, se il problema è decidibile, semidecidibile, o neanche semidecidibile.
Come cambia, se cambia, la risposta assumendo di sapere a priori che *f* non è totalmente indefinita (ossia indefinita per ogni valore del suo dominio)?

Esercizio 3 (Punti 10)
Si consideri il linguaggio L presentato nell'esercizio 1. Si descriva nel dettaglio un algoritmo per una macchina basata sull'architettura di Von Neumann (ad es. la macchina RAM), che restituisca in uscita la stringa wcw^R, se la stringa in ingresso x = wcW appartiene a L, cc altrimenti. Si supponga che la stringa in ingresso NON sia già disponibile in memoria. Se ne valuti la complessità (spaziale e temporale, con entrambi i criteri di costo).

Soluzioni

Esercizio 1
L è evidentemente un linguaggio non contestuale.
Una G che genera L è la seguente:

S → c | aSa | bSb | aAb | bAa
A → c | aAa | bAb | aBb | bBa
B → c | aBa | bBb | aCb | bCa
C → c | aCa | bCb

Similmente, un automa a pila che riconosca L può operare nel modo seguente:
Inizialmente impila i caratteri di w fino ad incontrare c.
Dopo aver letto c inizia a svuotare la pila rimanendo nello stesso stato finché il simbolo letto corrisponde al simbolo in cima alla pila. Se la pila si svuota quando si giunge a fine stringa in questo stato l'ingresso viene accettato.
Non appena viene letto un carattere no corrispondente al carattere in pila si cambia stato una prima volta; indi si prosegue lo svuotamento della pila nel modo usuale.
Sono possibili al più 3 cambiamenti di stato durante la lettura di W.

Esercizio 2
E' noto che per qualsiasi valore y esistono un polinomio P e un valore z tale che P(z) = y (ad esempio il polinomio costante y). Quindi il fatto che esistano un polinomio P a coefficienti interi non negativi e un valore $\overline{x} \in N$ tali che $P(\overline{x}) = f(\overline{x})$ è equivalente a stabilire se esiste un $\overline{x} \in N$ tale che $f(\overline{x})$ sia definito, ossia se *f* sia totalmente indefinita o meno.
Questo è un problema notoriamente indecidibile (tra i tanti modi di dimostrarlo si può utilizzare il teorema di Rice); tuttavia esso è semidecidibile: infatti se un tale $\overline{x}$ esiste, mediante una tipica enumerazione diagonale (simulo x mosse di una macchina che calcola f sul valore y, ecc.) esso viene individuato.
Se invece si sapesse a priori che *f* non è totalmente indefinita, allora l'esistenza di $\overline{x}$ sarebbe già garantita e quindi anche il problema di partenza sarebbe risolto.

Esercizio 3

Un semplice algoritmo che risolve il problema utilizza una lista di caratteri la cui lunghezza massima sarà pari alla lunghezza di w, una struttura dati LIFO (cioè una pila) ed un contatore che memorizza il numero di "errori" commessi dalla stringa W rispetto alla w^R (quest'ultimo è inizializzato a zero). Ogni volta che un carattere viene letto dall'input, si procede in questa maniera:

1. se non si è ancora letto il separatore 'c', si copia il carattere letto in coda alla lista e si impila lo stesso carattere sulla pila
2. se si legge il carattere 'c', si procede al carattere successivo
3. se si è già letto il separatore 'c', allora
    a. se il carattere appena letto corrisponde a quello in cima alla pila, si elimina quest'ultimo dalla pila e si procede alla lettura del carattere successivo
    b. altrimenti, si incrementa il contatore degli errori e
        i. se il contatore errori è maggiore di 3, si scrive 'cc' in output e si termina,
        ii. altrimenti, si procede con la lettura del carattere successivo
4. se la stringa è stata letta completamente dall'input e la pila è vuota, si utilizza la lista riempita al passo 1. per scrivere in output wcw^R. Per fare ciò, si scrive prima w scorrendo la lista a partire dalla testa, si inserisce 'c' , e si ripercorre la lista in direzione opposta, ottenendo quindi w^R in output. Altrimenti, si scrive 'cc' in output.

Chiamando n la lunghezza della stringa in ingresso, l'occupazione in memoria dell'algoritmo proposto è proporzionale a questa lunghezza. Infatti, per ciascun carattere letto, è necessario un numero costante di celle di memoria (nel caso peggiore, una posizione nella lista e una posizione sulla pila). Per cui, la complessità spaziale dell'algoritmo sarà $\Theta(n)$ sia a costo costante che a costo logaritmico. Per quanto riguarda la complessità temporale, i passi 1., 2. e 3. hanno un costo costante per singola esecuzione. Nel caso peggiore, 1. e 3. vengono eseguiti n volte. Analogamente, il passo 4. richiede un numero di operazioni direttamente proporzionale ad n nel caso peggiore. Di conseguenza, la complessità temporale dell'algoritmo valutata a costo costante sarà $\Theta(n)$, mentre valutata a costo logaritmico sarà $\Theta(n \log(n))$.