Algoritmi e Principi dell'Informatica

Prova in itinere del 24 novembre 2016

Il tempo per completare la prova è 1 h e 45 minuti.

Esercizio 1 (punti 7/15)

Si consideri un semplice linguaggio di programmazione LP1 che tratta solo interi, ed è dotato delle normali istruzioni di assegnamento (variabile = espressione), dell'istruzione condizionale **if-then-else** e del ciclo **while**. Si consideri poi un linguaggio LP2 che differisce da LP1 in quanto invece del ciclo **while** ha il costrutto **for** definito da questa sintassi:

for <variabile di tipo intero> = <costante intera1> **to** <costante intera2> **do {**<corpo del ciclo>**}**

in cui <costante intera2> ≥ <costante intera1>

e la cui semantica consiste in:

- Inizializzare la variabile intera, diciamo *i*, con il valore <costante intera1>,
- eseguire il corpo del ciclo,
- incrementare di un'unità *i*,
- uscire dal ciclo quando *i* è maggiore di <costante intera2>

NB: il corpo del ciclo non può contenere assegnamenti ad *i*.

E' decidibile il problema di stabilire se, dato un generico programma scritto in LP1 ne esiste un altro ad esso equivalente in LP2?

Spiegare brevemente la risposta.

Esercizio 2 (punti 5/15)

Specificare in logica del prim'ordine un predicato unario *pal* che indica che il suo argomento è una stringa palindroma (cioè che non cambia se letta da sinistra verso destra o da destra verso sinistra) servendosi <u>esclusivamente</u> dei seguenti predicati, funzioni e costanti:

- il predicato unario *char* che è vero se e solo se il suo argomento è una stringa di lunghezza 1;
- il predicato (binario) = che rappresenta l'uguaglianza tra stringhe;
- la funzione binaria • che rappresenta la concatenazione di stringhe;
- la costante ε, che rappresenta la stringa vuota.

Esercizio 3 (punti 4/15 per le parti a) e b), + 2 per la parte c); NB: la parte c) verrà valutata solo per chi abbia svolto correttamente le due parti precedenti).

Si consideri il linguaggio L delle stringhe palindrome costruite sull'alfabeto {a,b} (per la definizione di "palindromo" si veda l'esercizio precedente).

a) Definire una grammatica, a *potenza generativa minima*, che generi L.
b) Costruire un automa, a *potenza riconoscitiva minima*, che riconosca L.
c) Definire una grammatica, a *potenza generativa minima*, che riconosca il complemento di L.

Soluzioni

Esercizio 1

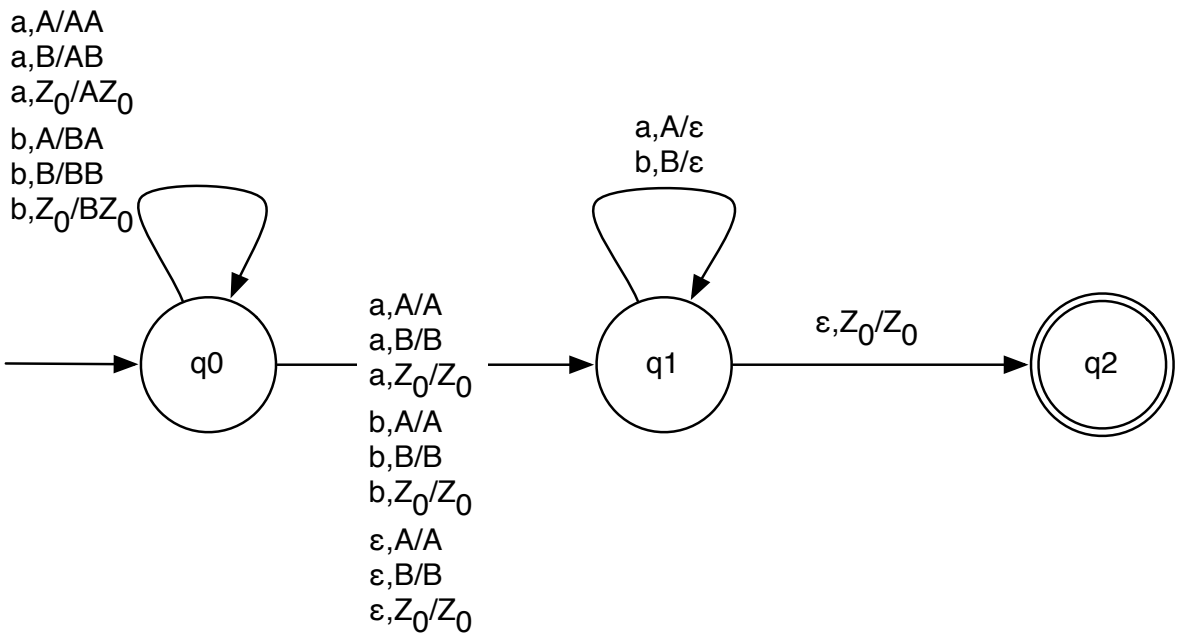
E' ben noto che i costrutti base di LP1 sono sufficienti per dare al linguaggio la potenza della MT (è un facile esercizio codificare una qualsiasi MT in LP1). Al contrario LP2 non ha tutta la potenza della MT (tra l'altro i suoi programmi terminano sempre). Quindi i problemi (in questo caso funzioni definite su numeri interi) risolvibili da LP2 sono un sottoinsieme proprio non vuoto dei problemi ricorsivamente enumerabili. Se fosse possibile decidere il problema suddetto sarebbe perciò anche possibile decidere se una generica MT risolve un problema della famiglia dei problemi risolvibili con LP2, violando il teorema di Rice.

Esercizio 2

$\forall x\ (pal(x) \leftrightarrow x=\varepsilon \vee char(x) \vee \exists y \exists z\ (char(y) \wedge (y \bullet z) \bullet y = x \wedge pal(z)))$

Esercizio 3

a) S → aSa | bSb | a | b | ε
b) E' sufficiente un NDPDA che, nondeterministicamente, individua il punto di mezzo del palindromo (di un carattere, se la stringa è di lunghezza dispari, o di zero caratteri se la stringa è di lunghezza pari). Non è possibile individuare tale punto deterministicamente prima di aver letto interamente la stringa, quando è ormai troppo tardi per determinare se si tratti di un palindromo.



c)

S → aSa | bSb | aTb | bTa

T → Ta | Tb | a | b | ε