Fondamenti dell'Informatica

13 luglio 2016

Esercizio 1

Si dica se il linguaggio

$$L = \left\{ x \in \{0,1\}^* \;\middle|\; \exists y \in \{0,1\}^* \,.\, xxx = yy \right\}$$

è regolare o meno, dimostrando formalmente ogni affermazione.

Esercizio 2

Negli automi a pila non deterministici è perfettamente possibile scambiare gli stati accettanti con quelli non accettanti. Questo implica che i linguaggi acontestuali sono chiusi per complementazione? Oppure no? Perché?

Esercizio 3

Si enunci e si dimostri il Teorema di Kleene sugli insiemi ricorsivamente enumerabili.

Esercizio 4

Si dica, motivando adeguatamente, se il seguente problema è decidibile:

> Dato un programma P scritto nel linguaggio While, un input per P, e due variabili x e y debitamente inizializzate all'inizio di P, dire se è vero o falso che, durante l'intera esecuzione di P successiva all'inizializzazione di x e y, si ha che $x \neq y$.

Esercizio 5

Sia $G = \langle V, T, P, S \rangle$ una grammatica CF. Sia $\Gamma \colon \wp(V) \to \wp(V)$ definito, per ogni $W \in \wp(V)$, da

$$\Gamma(W) = \left\{ A \in V \;\middle|\; \exists \alpha \in (T \cup W)^* \,.\, (A \to \alpha) \in P \right\}.$$

Si dimostri che, per ogni $i \geq 1$ e per ogni $A \in V$, $A \in \Gamma^i(\emptyset)$ se e solo se esiste un albero di derivazione per G che descrive $w \in T^*$, con radice etichettata con A e di altezza minore o uguale a i.

Esercizio 6

Si dica cosa stampa il seguente frammento in uno pseudolinguaggio con passaggio con scope statico, nei quattro casi: passaggio per valore (`value`), passaggio per riferimento (`reference`), passaggio per valore-risultato (`value_result`), passaggio per nome (`name`):

```
int x = 10;
void foo(value/reference/value_result/name int& y){
  x = x + 1;
  x = x + 1;
  y = y + 10;
  y = y + 1;
  x = x + y;
  write(x);
}

{
  int x = 50;
  foo(x);
  write(x);
}
```

Esercizio 7

In un certo linguaggio di programmazione con blocchi e funzioni c'è la possibilità di dichiarare una variabile locale ad una funzione come `own`, con la sintassi

```
own <tipo> <nome>;
```

Una variabile `own` dichiarata nella funzione `foo` è locale a `foo` ma la sua vita si estende anche dopo la terminazione dell'attivazione di `foo` nella quale è stata creata; in particolare, nel caso di una seconda attivazione di `foo` la variabile mantiene il valore che aveva nell'attivazione precedente. Ad esempio,

```
void foo(int y){
  own int x;
  if (y == 0) x = 0;
  ++x;
  print(x);
}
foo (0);
foo (1);
foo (1);
```

stamperà 1, 2, 3. Si descriva, dettagliando, un possibile modo di implementare le variabili `own`. Cosa succede in caso di ricorsione?