Prova Scritta di Linguaggi di Programmazione I

02/02/2009

Si noti che quanto messo nei riquadri è una *bozza* fornita solo a *titolo indicativo*.
Quindi **non è** un modello di soluzione completa che ci si aspetta ad un esame.

1. Con riferimento al seguente programma in Pascal, si rappresenti il P-code relativo alla funzione `vprod`, cioè la sezione di codice all'interno del riquadro. (il simbolo *cappelletto*, in sostituzione di $\uparrow$, si riferisce ai puntatori.)

```
program esercizio;

  type  v3 = array[ 0 .. 2 ] of real;  vect = ^v3;  ...

  function cmp( u, v: vect;  j, k: integer ) : real;
  begin  cmp := u^[j]*v^[k] - u^[k]*v^[j]  end;

  function vprod( u, v: vect ) : vect;

    var  i: integer;  z: vect;
  begin
    new( z );
    for  i := 0  to  2  do
      z^[i] := cmp( u, v,  (i+1) mod 3, (i+2) mod 3 );
    vprod := z
  end;

  begin  ...  ...  end.
```

2. Sia $I^{L_2}_{L_1}$ un interprete di L_2 scritto in L_1 e $PE^{L_2}_{L_1}$ un valutatore parziale di L_2 scritto in L_1. Sia $[\![P]\!]$ la funzione calcolata dal programma P, $\forall P$. Si stabilisca un programma Q per cui la valutazione $[\![PE^{L_1}_{L_0}]\!](Q, I^{L_3}_{L_2})$ abbia senso e produca un compilatore.

Scegliendo $Q = PE^{L_2}_{L_1}$

$$[\![PE^{L_1}_{L_0}]\!](PE^{L_2}_{L_1}, I^{L_3}_{L_2}) = P \in L_1$$

con P tale che, $\forall T \in L_3$

$$[\![P]\!](T) = [\![PE^{L_2}_{L_1}]\!](I^{L_3}_{L_2}, T) = R \in L_2$$

con R tale che, $\forall X$

$$[\![R]\!](X) = [\![I^{L_3}_{L_2}]\!](T, X) = [\![T]\!](X)$$

il risultato P è quindi un compilatore da L_3 a L_2 scritto in L_1 ($C^{L_3 \to L_2}_{L_1}$).

3. Sia G la grammatica individuata dalle produzioni

$$S ::= \mathtt{a}\ A\ \mathtt{b}\ S\ \mathtt{a} \mid \mathtt{d} \qquad\qquad A ::= \mathtt{b}\ S \mid \mathtt{h}\ A$$

e sia L il linguaggio $\{xyw \mid xw = w^R y,\ x, y \in \{\mathtt{a}, \mathtt{b}\},\ w \in \{\mathtt{b}, \mathtt{c}\}^*\}$, dove w^R è la stringa w rovesciata.

La grammatica G è ambigua? In ogni stringa di $\mathcal{L}(G)$ quante `b` si possono avere? Si dia l'albero di parsing di una stringa contenente esattamente 2 `b`.

Si diano le stringhe di L di lunghezza ≤ 5 e tutte quelle di $\mathcal{L}(G) \cap L$.

Prova Scritta di Linguaggi di Programmazione I

02/02/2009

> G non è ambigua.
>
> Ogni stringa di $\mathcal{L}(G)$ ha un numero pari di `b`.
>
> $\{w \in L \mid |w| \leq 5\} = \{\texttt{aa}, \texttt{bb}, \texttt{bbb}, \texttt{bbbb}, \texttt{bbcb}, \texttt{bbbbb}, \texttt{bbccb}\}$.
>
> $\mathcal{L}(G) \cap L = \emptyset$

4. Si mostri l'evoluzione delle variabili e l'output del seguente frammento di programma in un linguaggio C-like con assegnamento che calcola *l*-value *prima* di *r*-value, valutazione argomenti chiamate *da destra a sinistra* e indici vettori inizianti da 1:

```
int v[3] = {data_di_nascita};
int  i=1, j=2;
int mess(int i, ref int z) {
 while ( (v[i++] += v[i]--) < 70 );
 v[i--] -= v[++j] - z;
 return i;
}
write(mess(j--, v[j--]));
write(v[i], i++);
```

> Attenzione che l'ordine di valutazione degli argomenti delle chiamate *non ha nulla a che vedere* con quello che le procedure poi fanno con gli argomenti. In particolare una `write`(e_1, e_2) stamperà sempre prima (il valore di) e_1 e poi e_2.

5. Assumendo di utilizzare nell'Esercizio 8 la tecnica di implementazione del "display" si mostri (schematicamente) con dei diagrammi la situazione sul display e sullo stack di sistema quando entra in esecuzione H. La situazione incontrata è consistente con quanto dovrebbe succedere?

> La situazione è (casualmente) consistente.

6. Rappresentando Alberi "generici" con il tipo di dato

```
data (Eq a,Show a) => Tree a = Void | Node a [Tree a]
  deriving (Eq,Show)
```

si scriva una funzione `diameter` che determina il diametro di un albero. Il diametro di un albero è la lunghezza del massimo cammino fra due nodi, indipendentemente dall'orientamento degli archi.

7. Mediante la codifica ad albero chiamata "Quad Tree" si codificano immagini quadrate il cui lato sia una potenza di 2. Se l'immagine è omogenea (stesso colore) la si codifica, indipendentemente dalle sue dimensioni, con una foglia contenente il colore. Se l'immagine è eterogenea allora si utilizza un nodo i cui figli contengono le codifiche dei quadranti superiore-sinistro, superiore-destro, inferiore-sinistro, inferiore-destro, rispettivamente. Usando il tipo di dato

```
data (Eq a,Show a) => QT a = C a | Q (QT a) (QT a) (QT a) (QT a)
  deriving (Eq,Show)
```

si scriva una funzione Haskell `insertPict` che dati i QuadTrees di due immagini q_t, q_f ed un QuadTree "maschera" a valori booleani, costruisce il QuadTree dell'immagine risultante mantenendo i pixel di q_t in corrispondenza del valore `True` (della maschera) oppure di q_f in corrispondenza del valore `False`. Ad esempio

```
let q2 = Q z u u u; z = C 0; u = C 1
    mask = Q ct cf ct cf; cf = C False; ct = C True
in insertPict (C 4) q2 mask
```

restituisce Q (C 4) (C 1) (C 4) (C 1).

8. Si mostri l'evoluzione delle variabili e l'output del seguente frammento di programma espresso in un linguaggio C-like con scoping *statico*, *deep binding*, assegnamento che calcola *r*-value *dopo* *l*-value, valutazione delle espressioni da sinistra a destra e indici vettori inizianti da 0:

```
int x = 1, y[] = {3,5,7};
int F(valres int v, int R(ref int)) {
    void G(name int v) {
        write(--x);  y[x++] += v-x; write(x,y);
    }
    int x=1;
    G( v-- + R(v) );  return v--;
}
int H(ref int y) {
  int z = y--;
  x += z++;  write(x,y,z);
  return z--;
}
write(F(y[x++], H));
write(x,y);
```

$0, 6, 3, 5, 1, 12, 5, 7, 3, 6, 12, 2, 7$