Prova Scritta di Linguaggi di Programmazione I

17/07/2009

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. Si rappresenti il P-code relativo alla funzione `mmem`, cioè la sezione di codice Pascal all'interno del riquadro.

```
program esercizio;
  const IMAX = 9; UNKNOWN = 0;
  type harray = array[ 0..IMAX, 0..IMAX ] of integer; history = ^harray;
  ... ...

  function mmem( i, j: integer; h: history ) : integer;

  begin
    if h^[i,j] = UNKNOWN then
      begin
        if i*j = 0 then h^[i,j] := 1
                   else h^[i,j] := mmem( i-1, j, h ) + mmem( i, j-1, h )
      end;
    mmem := h^[i,j];
  end;

  function manh( i, j: integer ) : integer;
    var u, v: integer; h: history;
  begin
    new( h );
    for u := 0 to i do
      for v := 0 to j do h^[i,j] := UNKNOWN;
    manh := mmem( i, j, h )
  end;

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

2. Sia $PE_{L_1}^{L_2}$ un valutatore parziale di L_2 scritto in L_1. Sia $[\![P]\!]$ la funzione calcolata dal programma P, $\forall P$. Si stabiliscano un programma Q scritto in L_c ed un R per cui la valutazione $[\![PE_{L_a}^{L_b}]\!](R, Q)$ abbia senso e produca un interprete di L_d.

3. Sia G la grammatica individuata dalle produzioni

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

pensata per generare il linguaggio $L := \{v \mid v \in \{\mathtt{a},\mathtt{b}\}^*,\ v\mathtt{a} = \mathtt{a}v^R,\ \#(\mathtt{a}, v) \leq \#(\mathtt{b}, v)\}$, dove w^R è la stringa w rovesciata e $\#(x, w)$ è il numero di occorrenze del simbolo x dentro w.

La grammatica G è ambigua?

Si diano le stringhe di L di lunghezza ≤ 4; l'albero di parsing di una stringa di $\mathcal{L}(G) \setminus L$; una stringa di $L \setminus \mathcal{L}(G)$. Qualora non fosse possibile si giustifichi il perchè.

$\{w \in L \mid |w| \leq 4\} = \{\varepsilon, \mathtt{ab}, \mathtt{abb}, \mathtt{abab}, \mathtt{abbb}\}$.
$\mathtt{a} \in \mathcal{L}(G) \setminus L$. $\mathtt{abab} \in L \setminus \mathcal{L}(G)$.

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

```
int v[3]={data_di_nascita},  i=1, j=2;
int mess(ref int z, int j) {
 do write(v[j])
 while ( (v[j++] += v[j]--) < 70 );
 v[--j] -= ++v[i++] - z;
 return i+j;
}
write(mess(v[j],j--) + 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. Si scriva una funzione che data una matrice di dimensioni $n \times k$ ed una $k \times m$ restituisca la matrice prodotto corrispondente (di dimensioni $n \times m$). Si assuma di moltiplicare matrici con dimensioni compatibili e (se facesse comodo) matrici non degeneri.

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

   ```
   data (Eq a) => Tree a = V | N a [Tree a]
   ```

   si scriva un predicato `isSymm` che stabilisce se un albero ha una forma simmetrica (cioè è uguale, non considerando il contenuto, al suo trasposto, dove il trasposto di un albero si ottiene prendendo per ogni nodo i trasposti dei figli in ordine inverso).

   Ad esempio

   ```
   isSymm (N 1 [N 2 [N 4 []], N 3 [N 5 []]])
   ```

   restituisce **True**.

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

```
int x = 2, y[] = {3,4,5};
dynamic-scope int H(valres int y) {
  int z = y--;
  x += (y *= 2) + z++;  write(x,y,z);
  return y++;
}
static-scope int F(ref int v, int R(valres int)) {
    static-scope void G(name int v) {
        write(--x);  y[x--] += v-x; write(x,y);
    }
    int x = 1;
    G( R(v) + v++ );  return v++;
}
write(F(y[x++], H)); write(x,y);
```

$0, 19, 10, 7, -1, 19, 4, 11, 11, 19, 19, 4, 12$