Prova Scritta di Linguaggi di Programmazione II

10/04/2008

> 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 codice della macchina di Warren relativo al seguente programma in Prolog.

```
fibtree( [n], z ).

fibtree( [n,[n]], s(z) ).

fibtree( [n,L,R], s(s(H)) ) :- fibtree( L, s(H) ), fibtree( R, H ).
```

2. Si stabilisca (argomentando opportunamente) se il seguente programma è lineare destro/sinistro, duplicante, eliminante, collassante; se è constructor-based e se è ortogonale.

```
p n (x:xs) = f xs (x:x:xs) > n
f (x:xs) (y:_) | xs=:=(y:xs)++[x] = x+y
[]     ++ ys = ys
(x:xs) ++ ys = x:(xs++ys)
```

3. Sia dato il seguente programma Curry

```
f _ [y] (Just zs) = Just (y:zs)
f x [] Nothing = Just [x]
f True (x:xs@(_:_)) z = z
```

   Dopo averlo esplicitamente tipato, si stabilisca se è induttivamente sequenziale e si calcolino tutte le derivazioni di *needed* narrowing per il termine `f x y (f x y z) where x,y,z free`.

   Si dica inoltre se le posizioni selezionate nei vari passi delle derivazioni calcolate sono *Basic*.

> Le risposte (delle implementazioni di Curry) sono
>
> ```
> {y = [v], z = Just vs} Just (v:v:vs)
> {x = True, y = (_:_:_)} z
> ```

4. Si consideri il seguente frammento di codice sintatticamente ammissibile sia in Haskell che in Curry.

```
data P = Z | S P
data BST a = V | N a (BST a) (BST a)

bstElem x V = False
bstElem x (N y _ _) | x=:=y  = True
bstElem x (N y l r) | lt x y = bstElem x l
bstElem x (N y l r) | lt y x = bstElem x r

lt Z (S _)     = success
lt (S x) (S y) = lt x y
```

   Si assuma di aver esteso il Prelude di Haskell con le definizioni (=:=) = (==) e success = **True**.

   Si indichi il tipo di dato inferito nei due linguaggi.

   Come si comportano i due linguaggi per le queries `bstElem Z (N Z V V)` e `bstElem (S Z) (N Z V V)`?

   Si possono usare in input termini non-ground? E in caso quali risultati vengono forniti?

5. Sia $\max_n t$ il più grande dei valori strettamente minori di n di un BST di interi t (se ne esistono), si scriva una funzione in Prolog `boundedMaximum/3` che dato un numero n e una lista l di BST determina la lista dei $\max_n t$ che esistono, al variare di t in l.

   A titolo di esempio, `boundedMaximum(3, [node(3,void void), node(5,node(2,void,void),void)]` vale `[2]`

Prova Scritta di Linguaggi di Programmazione II

10/04/2008

6. Preso un gruppo $r_1, \ldots, r_n$ di rettangoli si vuole determinare la massima sequenza di miglior incapsulamento, cioè la più lunga sequenza $r_{x_1}, \ldots, r_{x_k}$ per cui il rettangolo r_{x_i} è strettamente contenuto nel rettangolo $r_{x_{i+1}}$, eventualmente ruotando i rettangoli di $\pi/2$ e *a parità di lunghezza* quella che minimizza la somma $\sum_i (h_{x_{i+1}} - h_{x_i})(l_{x_{i+1}} - l_{x_i})$, dove h e l sono le altezze e larghezze dei rettangoli.

   Si scriva una funzione Curry `maxEncapsulation` che data una lista di coppie che rappresenta i rettangoli determina la (una) massima sequenza di miglior incapsulamento. Ad esempio

```
maxEncapsulation [(6,4),(5,7),(8,4),(2,2)] = [(2,2),(6,4),(7,5)]
```

7. Si calcoli $T_P^{ca} \uparrow 3$ per il programma

```
avl(v,z).
avl(n(N,L,R), s(H)) :- avl(L, H),avl(R, H).
avl(n(N,L,R), s(s(H))) :- avl(L, s(H)), avl(R, H).
avl(n(N,L,R), s(s(H))) :- avl(L, H), avl(R, s(H)).

p(n(X,L,R)) :- R=n(X,L,_), avl(n(X,L,_),N), avl(R,N).
```

$$
\begin{aligned}
T_P^{ca} \uparrow 1 &= \{avl(v,z)\} \\
T_P^{ca} \uparrow 2 &= T_P^{ca} \uparrow 1 \cup \{avl(n(A,v,v),s(z))\} \\
T_P^{ca} \uparrow 3 &= T_P^{ca} \uparrow 2 \cup \{p(n(A,v,n(A,v,v))), avl(n(A,v,n(B,v,v)),s(s(z))), \\
&\qquad avl(n(A,n(B,v,v),v),s(s(z))), \\
&\qquad avl(n(A,n(B,v,v),n(C,v,v)),s(s(z)))\}
\end{aligned}
$$