Prova Scritta di Linguaggi di Programmazione II

17/09/2008

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.

```
pln( [] ).

pln( [_] ).

pln( [S|X] ) :- lst( X, S, Y ), pln( Y ).

lst( [S], S, [] ).

lst( [P,Q|X], S, [P|Y] ) :- lst( [Q|X], S, Y ).
```

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

```
f [A x]                    = x : f []
f [A x, y] | y =:= B x = h x x
g (x:xs) (f (y:A z:_)  = xs
```

	sx	dx	dup	el	col
R1	•	•			
R2	•		•		
R3	•	•		•	•
TRS	sì	no	sì	sì	sì

Il TRS *non* è constructor based (f sta nei pattern di R3).

IL TRS non è ortogonale, R2 ed R3 si sovrappongono (su f [A v,A w]).

3. Sia dato il seguente programma Curry

```
data T a b = A | B a | C b a

f [x] A           = []
f xs (C True y)   = y:xs
f [] (B Nothing) = x where x free
```

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

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

Le risposte (delle implementazioni di Curry) sono

```
{y = C True v} v:v:x
{x = [], y = B Nothing} z
```

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

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

```
badins x V = N x V V
badins x t@(N y _ _) | x=:=y = t
badins x (N y l r)
  | le x y = N y (badins x l) r
badins x (N y l r)
  | le y x = N y l (badins x r)

le Z _         = success
le (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 `badins Z (N Z V V)` e `badins (S Z) (N Z V V)`?

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

5. Si scriva una funzione in Prolog `maxDiameter/2` che data una lista l di BST determina il massimo dei diametri dei BST di l. Il diametro di un BST è la lunghezza del massimo cammino fra due nodi, indipendentemente dall'orientamento degli archi.

   A titolo di esempio, `maxDiameter([node(f, node(c, node(b, node(a, void, void), void), node(d, void, node(e, void, void))), void)], D)` restituisce `D = 4`.

6. Abbiamo 2 recipienti di capacità B e S (interi), con $B > S$. I due recipienti possono essere riempiti da, o svuotati in, una riserva illimitata d'acqua. Un recipiente può essere versato nell'altro fino a che lui è vuoto o l'altro è pieno. Data la seguente dichiarazione di tipo

   data Action = FillSmall | FillBig | PourSmall | PourBig | SmallInBig | BigInSmall

   si scriva in Curry una funzione (non-deterministica) *leaveInSmall*(B, S, C) che data una capacità C determina una sequenza di azioni che lasciano una quantità C nel recipiente piccolo.

7. Si calcoli $T_P^{ca} \uparrow 4$ per il programma P dell'Esercizio 1

$$
\begin{aligned}
T_P^{ca} \uparrow 1 &= \{lst([A], A, []), pln([A]), pln([])\} \\
T_P^{ca} \uparrow 2 &= T_P^{ca} \uparrow 1 \cup \{lst([A, B], B, [A]), pln([A, A])\} \\
T_P^{ca} \uparrow 3 &= T_P^{ca} \uparrow 2 \cup \{lst([A, B, C], C, [A, B]), pln([A, B, A])\} \\
T_P^{ca} \uparrow 4 &= T_P^{ca} \uparrow 3 \cup \{lst([A, B, C, D], D, [A, B, C]), pln([A, B, B, A])\}
\end{aligned}
$$