# Prova Scritta di Linguaggi di Programmazione II

23/03/2007

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.

```
short( [], [] ).
short( [X], [X] ).
short( [X, X | L], S ) :- short( [X | L], S ).
short( [X, Y | L], [X | S] ) :- X < Y, short( [Y | L], S ).
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| short/2: | try_me_else | | $short_1$ | $short_3$: | trust_me | | |
| | get_str | null/0 | $A_1$ | | alloc | | 3 |
| | get_str | null/0 | $A_2$ | | get_str | cons/2 | $A_1$ |
| | proceed | | | (X) | uni_var | | $X_3$ |
| $short_1$: | retry_me_else | | $short_2$ | | uni_var | | $X_4$ |
| | get_str | cons/2 | $A_1$ | | get_str | cons/2 | $X_4$ |
| (X) | uni_var | | $X_3$ | (Y) | uni_var | | $Y_1$ |
| | uni_var | | $X_4$ | (L) | uni_var | | $Y_2$ |
| | get_str | null/0 | $X_4$ | | get_str | cons/2 | $A_2$ |
| | get_str | cons/2 | $A_2$ | | uni_val | | $X_3$ |
| | uni_val | | $X_3$ | (S) | uni_var | | $Y_3$ |
| | uni_var | | $X_5$ | | put_val | $X_3$ | $A_1$ |
| | get_str | null/0 | $X_5$ | | put_val | $Y_1$ | $A_2$ |
| | preceed | | | | call | | </2 |
| $short_2$: | retry_me_else | | $short_3$ | | put_str | cons/2 | $A_1$ |
| | alloc | | 0 | | set_val | | $Y_1$ |
| | get_str | cons/2 | $A_1$ | | set_val | | $Y_2$ |
| (X) | uni_var | | $X_3$ | | put_val | $Y_3$ | $A_2$ |
| | uni_var | | $X_4$ | | call | | short/2 |
| | get_str | cons/2 | $X_4$ | | dealloc | | |
| | uni_val | | $X_3$ | | | | |
| (L) | uni_var | | $X_5$ | | | | |
| (S) | get_var | $X_6$ | $A_2$ | | | | |
| | put_str | cons/2 | $A_1$ | | | | |
| | set_val | | $X_3$ | | | | |
| | set_val | | $X_5$ | | | | |
| | put_val | $X_6$ | $A_2$ | | | | |
| | call | | short/2 | | | | |
| | dealloc | | | | | | |

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

$$f(x, y, B(z)) \to f(x, C(y), C(y))$$
$$f(A(x), g(y, E), z) \to g(z, z)$$
$$g(D(A(x), E), y)) \to B(y)$$

3. Sia dato il seguente programma Curry

```
f []      = Nothing
f (x:_) = Just x
```

```
g _       Nothing      = False
g []      (Just b)     = b
g (x:_)   (Just True)  = x
```

Si stabilisca se è induttivamente sequenziale e si calcolino tutte le derivazioni di *needed* narrowing per il termine `g x (f x) where x free`.

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

```
data P = Z | S P

f (S x) y = S (f x y)
f Z x     = x
f x Z     = x
```

Il tipo di dato inferito nei due linguaggi è lo stesso?

Ci si può aspettare di ottenere i medesimi risultati nei 2 linguaggi?

**Se sì** sebbene si abbiano gli stessi risultati, potrebbe avere un qualche effetto pratico una eventuale riscrittura del codice?

**Se no** si mostri come andrebbe riscritta una delle 2 versioni per ottenere gli stessi risultati o si spieghi perché ciò non sia possibile.

5. Si costruiscano degli alberi con le costanti `void/0`, `bal/3`, `left/3` e `right/3`. L'idea è di usare left, right o bal a seconda che il sottoalbero sinistro sia più profondo del destro, meno o uguale. Un albero è detto AVL se la differenza fra le profondità dei sottoalberi destro e sinistro di un qualunque nodo è al massimo 1. Si scriva un predicato PROLOG `isAVL/1` che dato un albero determina se è AVL e se i costruttori dei nodi sono consistenti con lo (s)bilanciamento.

   A titolo di esempio,

```
isAVL( left(3, right(2,void,bal(5,void,void)), bal(7,void,void) ) )
```

   è vero.

6. Si scriva una funzione Curry (non-deterministica) che fornisca una (ogni) rappresentazione di una (ogni) soluzione del gioco del Contadino con la Capra, i Cavoli e il Lupo. Il contadino deve riuscire a portare Capra, Cavoli e Lupo sulla riva opposta di un fiume con una barca che può contenere (oltre a lui) al massimo uno dei tre soggetti. Però se lascia soli Capra e Cavoli la Capra si mangerà i Cavoli e se lascia soli Lupo e Capra il Lupo si mangerà la Capra.

7. Si calcoli il minimo punto fisso dell'operatore delle conseguenze immediate per le risposte calcolate ($T_P^{ca}$) del seguente programma:

```
p(X,Y) :- r(X,Y), r(Y,X).
r(X,Y):-q(X,Y).
r(f(X),X).
q(X,f(X)).
q(a,a).
```

$\{q(A, f(A)), q(a, a), r(f(A), A), r(A, f(A)), r(a, a), p(A, f(A)), p(a, a), p(f(A), A)\}$