Prova Scritta di Linguaggi di programmazione
Corso di Linguaggi di programmazione I
06/07/2004

Prolog

Scrivere un predicato `almostBalanced/1` in SICSTUS PROLOG per determinare se un albero binario ha la seguente proprietà: per ogni nodo le altezze dei figli destro e sinistro differiscono al massimo di 1. L'altezza è la lunghezza del massimo cammino. Si ricordi che si possono usare i funzionali `abs` e `max`.
Ad esempio `almostBalanced(tree(4,void,tree(3,void,void)))` è vero.
Si scriva il programma con variabili anonime ove possibile.

Unificazione

Si dica se esiste un unificatore più generale di `f(g(X),X,h(X))` e `f(T,g(Z),h(T))` e in caso affermativo lo si calcoli.

Semantica

Si consideri il seguente frammento di programma espresso in un linguaggio imperativo C-like con scoping statico:

```
{   int x:=7;
    proc Q(value int y; ref int w)
    {
            x := 3*y;
            w := w+1;
            y := w+2*y;
    }
Q(x, x);
}
```

si fornisca la semantica denotazionale del frammento, mostrando l'ambiente e la memoria nel momento in cui termina la valutazione della chiamata di procedura Q.

NOME E COGNOME:	Numero di Matricola:

Prolog

```
almostBalanced(void, 0).
almostBalanced(tree(_,L,R), D) :-
  almostBalanced(L, DL),
  almostBalanced(R, DR),
  Diff is abs(DL-DR),
  Diff =< 1,
  D is 1+max(DL,DR).
```

Unificazione

Non esiste unificatore.

Prova Scritta di Linguaggi di programmazione
Corso di Linguaggi di programmazione II
06/07/2004

Curry

Data la seguente definizione del tipo di dato astratto (polimorfo) "Alberi Binari di Ricerca"

```
data BST a = Void | Node a (BST a) (BST a)
```

si scriva un predicato (deterministico e non necessariamente flexible) Curry `almostBalanced` (un predicato è una funzione booleana) per determinare se un albero binario ha la seguente proprietà: per ogni nodo le altezze dei figli destro e sinistro differiscono al massimo di 1. L'altezza è la lunghezza del massimo cammino. Si ricordi che si possono le funzioni di libreria `abs` e `max`.
Ad esempio `almostBalanced(Node 4 Void (Node 3 Void Void))` riduce a `True`.

Narrowing

Si applichi un passo di narrowing a partire dal termine f(g(x),h(x)) usando le regole
g(z) -> h(z)
h(1) -> 3

CLP(R)

Si scriva un programma CLP(R), dove R è il dominio dei Reali, che data una matrice di dimensioni mxn in input restituisce in output la corrispondente matrice trasposta (di dimensioni nxm).

NOME E COGNOME:	Numero di Matricola:

Curry

```
almostBalanced :: BST a -> Bool
almostBalanced T = isbalanced
  where
    (isbalanced, _) = almost T

    almost Void = (True, 0)
    almost (Node _ l r) =
      let (balL,depL) = almost l
          (balR,depR) = almost r
      in if balL && balR && abs (depL-depR) <= 1
            then (True, 1+max depL depR)
            else (False, 0)
```

Narrowing

f(g(x),h(x)) -> {} f(h(x),h(x)) + {x/1} f(g(1),3)