Gedächtnisprotokoll der 1. ProMo Klausur SS12 - Bry, Hausmann, Zauner

Max

September 6, 2012

1) a) Definieren sie eine monomorphe Funktion
   b) Definieren sie eine polymorphe Funktion
   c) Definieren sie eine gecurryte Funktion, deren ungecurryte Version ein Tupel enthält.
   d) Definieren sie eine Funktion höherer Ordnung

2) a) Schreiben sie eine Funktion `(order b b')`, die dieselbe Signatur/Wirkung hat wie `b orelse b'`
   b) Definieren sie ein A und ein A', so dass A und A' syntaktisch korrekt sind, aber verschiedene Ergebnisse liefern in Bezug auf die beiden Funktionen.

3) Definieren sie

```
reverse z = let
              val y = ref [];
              val x = z
            in
              while not(hd(!x) != []) do (
                y := hd(!x) :: !y;
                x := tl(!x);
              )
              !y;
            end
```

   in rein funktionaler Schreibweise, wobei die entstehende Funktion nicht selbst rekursiv sein darf.

4) Sei folgender Code gegeben:

```
let
  val y = 5;
  val x = 2;
  fun g(y) = x * y;
  fun f(x,y) = x + g(y)
in
  (g(y), f(5,7))
end;
```

   a) Was ist der Wert dieses Ausdrucks bei statischer Bindung?
   b) Was ist der Wert dieses Ausdrucks bei dynamischer Bindung?
   c) Wo ist der Nachteil der dynamischen Bindung?

5) Seien

```
fun f x = 1+x
fun g x = f(x) * f(x)
fun h x y = x+1 > 0 orelse f(y) < 0
```

Geben sie zu den folgenden fünf Auswertungs ausdrücken an, ob es sich hierbei um ein korrektes Fragment einer Evaluation handelt, die bei normaler Ausführung, bei applikativer Ausführung und gleicher Semantik von Sonderausdrücken wie in SML, bei keiner von beiden oder bei beiden, handelt.

a)

6) Definieren sie das Standardskalarprodukt über $\mathbb{Z}$,

$$\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z},\ x, y \mapsto \sum_i x_i y_i$$

funktional mit Hilfe von `List.Map` und `List.foldl/foldr` sowie

```
fun zip (x::xs)(y::ys) = (x,y) :: zip xs ys
  | zip [] [] = []
```

7) Seien

```
val ks = [3, 5, ~7]
val pred = fn x => x < 0
fun f p x (y :: ys) = if p(y) then f(not o p) x ys
                              else f p x ys
    f p x _         = p(x)
fun g x = let
            fun f x = x
          in
            g f x
          end
```

Geben sie die Werte und Typen von folgenden Ausdrücken an

```
f
f pred 0 []
f pred 0 ks
g
g ks
```

8) Sei `datatype 'a formula = Con of bool | Var of 'a | Not of a' formula | Or of a' formula * a' formula| And of a' formula * a' formula;`
In der Angabe war noch ein Beispielbaum und eine Repräsentation einer Formel gegeben gegeben. Deklarieren sie `eval (f)` als `int formula -> bool`, so dass Zahlen ≥ 0 eine wahre Variablenbelegung ergeben.

9) a) Was unterscheidet exceptions/Ausnahmen von anderen Programmbestandteilen in Hinsicht auf die Auswertung?

   b) Warum benutzt man sie trotzdem?

Korrekturen bitte an Max ät fs.lmu.de.