Aufgabenblatt zur 3. Übung

Zeitraum: 26.04. bis 29.04.2011

1. Aufgabe: (Klausuraufgabe 02.11)

Schreiben Sie die in den folgenden Teilaufgaben geforderten Funktionen in HASKELL, einschließlich ggf. benötigter Hilfsfunktionen. Geben Sie zu *jeder* Funktion eine Typdeklaration an!

(a) Eine Liste von ganzen Zahlen soll auf folgende Eigenschaft hin überprüft werden: Die Differenz zwischen zwei benachbarten Elementen soll abwechselnd positiv und negativ sein, nicht jedoch null. Dabei darf die Liste mit einer positiven oder negativen Differenz beginnen, und sie darf einelementig oder leer sein.

Geben Sie eine Funktion `test` an, die eine Liste entgegennimmt und genau dann `True` liefert, wenn die Liste die besagte Eigenschaft hat.

(b) Geben Sie eine Funktion `delcon :: Int -> [[Int]] -> [Int]` an, die aus einer Liste von Listen alle diejenigen Listen entfernt, deren Elemente in Summe die im ersten Argument gegebene ganze Zahl ergeben, und die verbleibenden Listen der Reihe nach zu einer Liste konkateniert.

2. Aufgabe: (AGS 11.13)

Gegeben sei der folgende polymorphe algebraische Datentyp:

```
data Tree a = Branch (Tree a) (Tree a) | Leaf a
```

mit dessen Hilfe sich binäre Bäume konstruieren lassen, die an den Blättern Werte des Typs `a` speichern.

(a) Programmieren Sie eine Funktion `check :: Tree Bool -> Bool`, die genau dann `True` liefert, wenn der Eingabebaum mindestens ein Blatt mit dem gespeicherten Wert `False` besitzt.

(b) Programmieren Sie eine Funktion `toList :: Tree Int -> [Int]`, die aus einem Baum des Typs `Tree Int` eine Liste der gespeicherten Werte der Blätter generiert, und zwar in der Reihenfolge von **rechts nach links** gesehen.

(c) Programmieren Sie eine Funktion `toTree :: [Int] -> Tree Int`, die eine Liste von Zahlen als Argument nimmt und einen Baum erzeugt, welcher - zusätzlich zu einem Blatt mit dem gespeicherten Wert 42 - für jedes Element der Liste genau ein Blatt mit dessen Wert besitzt. (Hinweis: Die Form des Baumes spielt keine Rolle.)

(d) Programmieren Sie eine Funktion `transform :: [Bool] -> [Bool]`, die aus einer Liste von Wahrheitswerten alle Werte `False` herausstreicht und die Anzahl der Werte `True` verdoppelt.

3. Aufgabe: (AGS 11.18)

Gegeben sei der polymorphe Typ:

```
data Tree t = Leaf t | Branch t  (Tree t)  (Tree t)
```

(a) Schreiben Sie eine polymorphe Funktion `liste` (einschließlich Typdefinition von `liste`), die aus jedem Baum B o. g. Typs jeweils eine Liste des Typs `[t]` aller Knotenbewertungen des Baumes B in der Reihenfolge $<$ linker Teilbaum , Wurzelbewertung , rechter Teilbaum $>$ erzeugt („inorder“–Durchlauf).

(b) Seien `f::t -> t` eine Funktion, `x` eine Variable vom Typ `Tree t` und `map` die Ihnen bekannte Haskell–Funktion.

Schreiben Sie eine Funktion `g` (einschließlich Typdefinition von `g`), so dass folgende Gleichung gilt:

```
map f (liste x) = liste (g f x)
```

4. Aufgabe:(AGS 11.28*)

Es sei δ ein dreistelliges, γ ein einstelliges und α ein nullstelliges Funktionssymbol. Des Weiteren sei $V = \{x_1, \ldots, x_5\}$ eine Menge von Variablen.

Wenden Sie den Unifikationsalgorithmus jeweils auf die Terme t_1 und t_2 an und ermitteln Sie jeweils den allgemeinsten Unifikator:

- $t_1 = \delta(\gamma(x_1), \delta(\gamma(\alpha), \gamma(x_2), \gamma(\gamma(\alpha))), x_3)$
  $t_2 = \delta(\gamma(\gamma(x_5)), \delta(x_4, \gamma(x_2), \gamma(x_1)), \alpha)$
- $t_1 = \delta(\gamma(\alpha), x_3, \gamma(\gamma(x_3)))$
  $t_2 = \delta(x_1, \delta(\alpha, x_2, \alpha), x_2)$

Zusatzaufgabe1: (AGS 11.19*)

(a) Definieren Sie eine Funktion `addL`, welche die Elemente einer Liste von ganzen Zahlen aufsummiert. Geben Sie den Typ von `addL` an.

(b) Gegeben sei der Datentyp `data UTree = UI [UTree] | UL Int` zur Repräsentation von Bäumen bei denen i) die inneren Knoten (beschriftet mit `UI`) beliebig viele Nachfolger haben und ii) die Blattknoten (beschriftet mit `UL`) Integer-Werte tragen. Definieren Sie eine Funktion `addU :: UTree -> Int`, welche alle `Int`-Werte in einem solchen Baum aufsummiert. Nutzen Sie dafür die Funktion `addL` und die Funktion `map :: (a -> b) -> [a] -> [b]` aus der Vorlesung, die eine Liste elementweise transformiert.

(c) Definieren Sie einen polymorphen Datentyp `Tree a` zur Repräsentation von Bäumen, bei denen i) die inneren Knoten genau drei Nachfolger haben und ii) die Blattknoten Werte eines beliebigen, aber festen Typs `a` tragen. Geben Sie außerdem für diesen Datentyp (nur) den Typ einer polymorphen Funktion `mapTree` an, welche analog zur bekannten Funktion `map` für Listen, einen Baum blattweise transformiert. (D.h. `mapTree` besitzt zwei Argumente - eins für die Transformation der Blattwerte und eins für den Eingabebaum.)

Zusatzaufgabe2: (AGS 11.30*)

(a) Es seien σ ein zweistelliges, γ ein einstelliges und α ein nullstelliges Funktionssymbol.
$V = \{x_1, x_2, x_3\}$ sei eine Menge von Variablen.
Wenden Sie den Unifikationsalgorithmus auf die Terme t_1 und t_2 an und ermitteln Sie deren allgemeinsten Unifikator:

$$t_1 = \sigma(\sigma(\gamma(x_1), x_2), \gamma(\gamma(\alpha)))$$
$$t_2 = \sigma(\sigma(\gamma(\alpha), \gamma(\gamma(x_1))), \gamma(x_3))$$

Geben Sie dabei jeweils die benutzte Regel an.

(b) Identifizieren Sie die Typsymbole σ, γ und α mit den Typkonstruktoren $()^2$, $[\,]$ und `Int`.
Geben Sie nun zu t_1, t_2 aus (a) die äquivalenten Typterme und Typausdrücke an.