PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

14. Januar 2004

Übungen zur „Praktischen Informatik III“, WS 2003/04

Nr. 10, Besprechung bzw. Abgabe: 21. und 22. Januar in den Übungen

---

A. Mündliche Aufgaben

46. Typen

Bestimmen Sie (falls möglich) den allgemeinsten Typ der folgenden Ausdrücke:

(e) `map map` (f) `map foldr` (g) `foldr map` (h) `foldr foldr`

47. Unifikation

Lösen Sie die folgenden Typgleichungen:

(a) `([f], Int -> (e, Int), b) = (d, c -> g, d)`

(b) `(c -> Int, [a], b, c) = (f, d, e, f)`

(c) `[[([Bool -> h], Bool, h)]] = [[([j], e, d -> e)]]`

B. Hausaufgaben

48. Typinferenz — 4 Punkte

Führen Sie für die folgende Funktion eine Typinferenz mit dem in der Vorlesung vorgestellten Verfahren durch:

```
span p []     = ([], [])
span p (x:xs) = let (ys, zs) = span p xs
                in if p x then (x:ys, zs)
                          else ([], x:xs)
```

49. Typklassen — 4 Punkte

Gegeben seien die folgenden Deklarationen:

```
-- verschiedene Baumdefinitionen
data BinTree a   = Leaf a  | Node (BinTree a) (BinTree a)
data LabTree l a = LLeaf a | LNode l (LabTree l a) (LabTree l a)
data STree a     = Empty   | Split a (STree a) (STree a)
data RoseTree a  = RNode a [RoseTree a]
data AExp a      = Val a   | Plus (AExp a) (AExp a)
                           | Mult (AExp a) (AExp a)

-- Typklasse Tree
class Tree t where
  subtrees :: t -> [t] -- liefert direkte Teilbaeume
```

(a) Geben Sie für jede Baumart eine Instanzendeklaration für die Klasse `Tree` an. / 1

(b) Definieren Sie die folgenden Funktionen: / 3

- `depth :: Tree t => t -> Int` bestimmt die Tiefe eines Baumes.
- `size  :: Tree t => t -> Int` bestimmt die Knotenanzahl eines Baumes.
- `dfs   :: Tree t => t -> [t]` liefert die Liste *aller* Teilbäume in Präfixordnung.

50. Unterklassen **4 Punkte**

Gegeben sei folgende Deklaration einer Unterklasse der Klasse `Tree` zur graphischen Ausgabe von Bäumen:

```
class Tree t => DrawTree t where
  drawTree :: t -> String
  labTree  :: t -> String

  drawTree = unlines . drawTree' labTree
```

(a) Geben Sie für die Baumdefinitionen aus der vorigen Aufgabe Instanzendeklarationen für `DrawTree` an. Die Defaultdefinition von `drawTree` soll dabei nicht verändert werden. / 1

(b) Definieren Sie die Funktion / 3

`drawTree' :: Tree t => (t -> String) -> t -> [String]`

Das erste Argument dieser Funktion ist eine Funktion mit dem Typ `(t -> String)`, die die Zeichenkettendarstellung der Wurzelmarkierung des Argumentbaums liefert. Im Beispiel (siehe unten) wird ein Knoten ohne Markierung durch das Zeichen `@` dargestellt. Das zweite Argument von `drawTree'` ist der darzustellende Baum. Das Resultat der Funktion ist eine Liste von Zeilen, die in der Defaultdefinition von `drawTree` mittels `unlines` zu einer Zeichenkette mit `\n`-Zeichen nach jeder Zeile zusammengefügt werden.

Beispiel:

```
> drawTree (Node (Node (Leaf 1) (Leaf 2))  (Node (Leaf 3) (Leaf 4)))
--@--@--1
  |  |
  |  `--2
  |
  `--@--3
     |
     `--4
```