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

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

3. Dezember 2003

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

Nr. 7, Besprechung bzw. Abgabe: 10. und 11. Dezember in den Übungen

---

A. Mündliche Aufgaben

30. Anwendung von Funktionen höherer Ordnung

Definieren Sie die folgenden Funktionen unter Verwendung vordefinierter Funktionen höherer Ordnung:

(a) `pairAndOne :: Num a => [a] -> [(a,a)]` paart jedes Element einer Liste von Zahlen (Typklasse `Num a`) mit dem um eins inkrementierten Element.
Beispiel: `pairAndOne [1,2,3]` $\Rightarrow^*$ `[(1,2), (2,3), (3,4)]`

(b) `addEachPair :: Num a => [(a,a)] -> [a]` addiert jedes Zahlenpaar einer Liste.
Beispiel: `addEachPair [(1,2), (2,3), (3,4)]` $\Rightarrow^*$ `[3,5,7]`

(c) `addPointwise :: Num a => [(a,a)] -> (a,a)` addiert komponentenweise die Elemente einer Liste von Zahlenpaaren.
Beispiel: `addPointwise [(1,2), (2,3), (3,4)]` $\Rightarrow^*$ `(6,9)`

31. Testfunktion höherer Ordnung

Die Funktion `exists :: (a -> Bool) -> [a] -> Bool` soll bei einem Aufruf `(exists p xs)` überprüfen, ob mindestens ein Element der Liste `xs` das Prädikat `p` erfüllt. Zum Beispiel soll der Ausdruck `exists even [3,5,7,6]` den Wert `True` ergeben.

(a) Erstellen Sie eine herkömmliche rekursive Definition der Funktion.

(b) Implementieren Sie die Funktion mittels `foldl` oder `foldr`.

32. Test auf geordnete Liste

Definieren Sie die Funktion `ordered :: [Int] -> Bool`, die testet, ob eine Liste ganzer Zahlen aufsteigend geordnet ist, unter Verwendung von geeigneten Funktionen höherer Ordnung.

B. Hausaufgaben

33. Zeilennummern — 2 Punkte

Schreiben Sie eine Funktion `addLineNumbers :: String -> String`, die einen Text um Zeilennummern erweitert.

Hinweis: Verwenden Sie die vordefinierten Funktionen `lines`, `unlines`, `zip` und `map`.

34. Listenspiegelung als Listenfaltung — 3 Punkte

Die Listenspiegelung `reverse :: [a] -> [a]` kann als Listenfaltung definiert werden:

```
reverse xs = foldl revOp [] xs
```

(a) Geben Sie die Definition des hierzu benötigten Faltungsoperators `revOp` an. / 1

(b) Zeigen Sie die Korrektheit der so definierten Listenspiegelung. / 1

(c) Analysieren Sie die Zeitkomplexität dieser Definition. / 1

35. Wertetabelle zu aussagenlogischer Formel — 7 Punkte

Ausagenlogische Formeln können in Haskell durch die folgende Datenstruktur repräsentiert werden:

```
data Formula = Var String | Not Formula
             | And Formula Formula | Or Formula Formula
```

Die Formel $A \wedge (A \vee B)$ wird durch den folgenden Term dargestellt:

```
And (Var "A") (Or (Var "A") (Var "B"))
```

(a) Schreiben Sie eine Funktion `vars :: Formula -> [String]`, die zu einer Formel eine Liste aller in der Formel vorkommenden Variablen bestimmt. / 1

(b) Eine Belegung von Variablen mit Wahrheitswerten kann als Liste von Paaren dargestellt werden:

```
type Assignment = [(String,Bool)]
```

Definieren Sie eine Funktion `assignments :: [String] -> [Assignment]`, die zu einer Liste von Variablen alle möglichen Belegungen der Variablen mit Wahrheitswerten berechnet. / 2

Zum Beispiel soll der Aufruf `assignments ["A","B"]` die folgende Liste liefern:

```
[ [("A",True), ("B",True)],
  [("A",True), ("B",False)],
  [("A",False), ("B",True)],
  [("A",False), ("B",False)] ]
```

(c) Definieren Sie eine Funktion `truthValue :: Assignment -> Formula -> Bool`, die den Wahrheitswert einer Formel in Abhängigkeit einer Belegung der in der Formel vorkommenden Variablen mit Wahrheitswerten bestimmt. / 2

(d) Schreiben Sie eine Funktion `truthTable :: Formula -> [(Assignment,Bool)]`, die zu einer Formel die Wahrheitstabelle in der durch den Typ festgelegten Form bestimmt. / 1

(e) Schreiben Sie eine Faltungsfunktion `foldFormula` und drücken Sie die Funktionen `vars` und `truthValue` als Instanzen dieser Funktion aus. / 1