6. Arbeitsblatt

Woche 26.01.1998 und Woche 02.02.1998

Leitfragen

1. Was sind Funktionen höherer Ordnung? Gibt es entsprechende Konzepte in anderen Programmiersprachen, die Sie kennen?

2. Was ist der Unterschied zwischen den folgenden Funktionstypen:

   ```
   (Int-> String)-> Int     Int-> (String-> Int)     Int-> String-> Int
   (Int, String)-> Int      a-> String-> a           (a, a-> String)-> String
   ```

   Geben Sie für jeden Funktionstypen eine Funktion an, die diesen Typ besitzt.

3. Welche der drei "Basisfunktionale" `map`, `filter` und `fold` läßt sich mit Hilfe der jeweils zwei anderen definieren? Gibt es eine einzige Funktion, daß sich zu allen drei spezialisieren läßt?

4. Was ist der Unterschied zwischen `foldr` und `foldl`, und wann sind sie gleich (d.h. für welche `f`, `e` ist `foldr f e` = `foldl f e`)? Wann sind überhaupt zwei Funktionen gleich?

5. Eine Funktion, die Listen sortiert, läßt sich auf zwei Weisen darstellen:

   ```
   sort :: (a-> a-> Bool)-> [a]-> [a]
   sort :: Ord a => [a]-> [a]
   ```

   Was ist der Unterschied, und was sind die jeweiligen Vor- und Nachteile?

6. Versuchen Sie, einen Algorithmus zu skizzieren, der den Typ eines Haskell-Ausdrucks bestimmt.

Anwendungsaufgaben (die in der Gruppe entwickelt und analysiert werden sollen)

1. Geben Sie Implementationen für die Funktionen `map`, `filter`, `foldr`, `foldl` an.

2. Geben Sie Implementationen für folgende Funktionen an:

   ```
   zipWith   :: (a-> b-> c)-> [a]-> [b]-> [c]
   applyAll  :: [a-> a]-> a-> a
   ```

3. Ein Rückblick auf das 2. Arbeitslatt: Geben Sie alternative Implementation von `totalSales`, `maxSales`, `zeroSales` unter Benutzung von Funktionen höherer Ordnung.

4. Ein Rückblick auf das 3. Arbeitsblatt: Geben Sie eine Implementation des ADT `Store` als `Var-> Int`.

5. Ein *Palindrom* ist ein Wort oder Satz, der von vorne und hinten gleich gelesen werden kann; ein *Anagramm* ist ein Wort, welches durch Vertauschen der Buchstaben eines anderen Wortes entsteht. Implementieren Sie die Funktionen

   ```
   is_palindrome :: String-> Bool
   is_anagramme  :: String-> String-> Bool
   ```