Lösung für Übungsblatt 19 zu "Programmiersprachen"

Berthold Hoffmann, Studiengang Informatik (hof@informatik.uni-bremen.de)

Besprechung war am 29. Juni 2010

Schritte in die Funktionale Glückseligkeit

In der Vorlesung haben wir gemeinsam folgende Fragen untersucht:

1. Welche Konzepte sind für das funktionale Programmieren wichtig?
2. Welche Konzepte heben es von anderen Programmierstilen ab?
3. In welche Rangfolge würden Sie diese Konzepte setzen?

Folgende Liste haben wir zusammengestellt, wobei wir über die Reihenfolge nicht ganz einig waren. Diese Liste ist nach dem Motto "Vom Grundsätzlichen zum Speziellen" geordnet:

1. *Algebraische Datentypen*: zusammengesetzte Daten werden als (rekursive) Summen und Produkte konstruiert.
   Zeiger werden vor dem Benutzer verborgen.
2. *Mustervergleich* (*Pattern Matching*): Funktionen werden mit Mengen von (rekursiven) Gleichungen über Konstrukterausdrücken (*pattern*, Muster) definiert.
3. *Höhere Funktionen*: Funktionen können Parameter und Resultate von Funktionen sein.
4. *Parametrische Polymorphie*: Typen können *Typvariablen* enthalten, und Funktionen können polymorph typisiert sein.
5. *Referentielle Transparenz*: Funktionen haben keine Seiteneffekte: gleiche Ausdrücke haben immer den gleichen Wert.
   Die ist das *Reinheitsgebot* für funktionale Sprachen.
6. *Currying* und *partielle Parametrisierung*: Eine an sich mehrstellige Operation wie (+):: **Num** a =>(a,a) -> a wird als einstellige Funktion (+):: **Num** a =>a -> (a -> a) aufgefasst, und ein Ausdruck (+) 13 iste eine partielle Funktionsauwendung, die eine einstellige Funktion liefert (die eine gegeben Zahl um 13 erhöht).
7. *Verzögerte Auswertung* (*lazy evalzuation*): Die aktuellen Parameter einer Funktion werden erst ausgewertet, wenn sie das erste Mal gebraucht werden, und auch nur so weit, wie ihr Wert für die Auswertung benötigt wird.

Das Reinheitsgebot ist zwar ein wichtiges Unterscheidungsmerkmal, an dem sich Sprachen wie ML und Scala von Sprachen wie Miranda, Haskell und Clean abgrenzen lassen; es ist aber fragwürdig, ob der *Overhead* für das Einbauen kontrollierter Seiteneffekte durch Monaden (Haskell) oder *Uniqueness typing* (Clean) das Ergebnis rechtfertigt.

Folgende Konzepte z.B. von Haskell wurden als nicht so wichtig angesehen (und in der Diskussion gar nicht erwähnt):

1. *Listenumschreibungen*: Listen können "mengentheoretisch" definiert und berechnet werden
2. *Typklassen*
3. *Monaden*