Hochschule Flensburg
Fachbereich 3
Angewandte Informatik

Jan Christiansen

Hochschule Flensburg
University of Applied Sciences

3. Übung zur Vorlesung „Fortgeschrittene funktionale Programmierung“

Listen und Bäume

Labor am Montag, 14. Oktober 2016, 12:15 Uhr

Aufgabe 1 - Refactoring der Graphik

Der Datentyp *Graphic*, den Sie in der zweiten Übung implementiert haben, entspricht im Endeffekt einer Liste vom Typ $[Object]$. Diese Eigenschaft wollen wir nun nutzen, um vordefinierte Funktionen nutzen zu können.

Ersetzen Sie den Typ *Graphik* durch die folgende Definition.

$$\textbf{type}\ Graphic = [Object]$$

Ein solches Typsynonym führt dabei nur einen neuen Namen für einen Typen ein. Das heißt, an jeder Stelle, an der ein Wert vom Typ *Graphic* verwendet wird, können Sie stattdessen einen Wert vom Typ $[Object]$ verwenden.

Zum Beispiel ist der Typ *String* ein Typsynonym für den Typ $[Char]$. Die Funktion, $(+\!\!+)$, mit der Sie zwei Zeichenketten konkatenieren können, können Sie daher auch verwenden, um andere Listen zu konkatenieren. Überprüfen Sie, ob Sie die Funktion $(+\!\!+)$ auf Listen in Ihrer Implementierung nutzen können. Schauen Sie außerdem, ob Sie die Funktion *map* verwenden können.

Aufgabe 2 - XML-Datenstruktur

Definieren Sie den folgenden Datentyp zur Darstellung eines XML-Dokumentes.

$$\begin{aligned}&\textbf{data}\ XML = XText\ String\\ &\qquad\qquad \mid\ XNode\ String\ [Attr]\ [XML]\\ &\textbf{data}\ Attr = String := String\end{aligned}$$

Dabei verwendet der Typ *Attr* einen Infix-Konstruktor. Das heißt, der Konstruktor des Typs *Attr* heißt := und wird zwischen seine beiden Argumente geschrieben. Der Name eines Infix-Konstruktors muss immer mit einem Doppelpunkt starten.

Definieren Sie eine Funktion $xmlToString :: XML \to String$, um ein XML-Dokument in die entsprechende String-Darstellung umzuwandeln. Schauen Sie, ob Sie die vordefinierten Funktionen *map*, *concatMap* und *unwords* verwenden können. Schreiben Sie die Funktionen *objToSVG* und *toSVG* aus der zweiten Übung so um, dass Sie den Typ $Object \to XML$ bzw. $Graphic \to XML$ erhalten.

Aufgabe 3 - Binärbäume

Definieren Sie einen polymorphen Datentyp *BinTree* für Binärbäume, der an den inneren Knoten und den Blättern beschriftet ist. Definieren Sie die folgenden Funktionen auf dem Datentyp *BinTree*.

- Definieren Sie eine Funktion $sumTree :: BinTree\ Int \to Int$, welche die Summe aller Werte in einem mit Zahlen beschrifteten Baum berechnet.

- Definieren Sie eine Funktion $values :: BinTree\ a \rightarrow [a]$, die alle Werte eines Baumes in einer Liste zurückliefert.
- Definieren Sie eine Funktion $mapTree :: (a \rightarrow b) \rightarrow BinTree\ a \rightarrow BinTree\ b$, die eine gegebene Funktion auf alle Werte im Baum anwendet.

Hinweis: Bitte geben Sie die Lösungen der Aufgaben per Mail oder auf Papier bis zum 21.10. ab. Die Papierlösungen können im ersten Stock von Haus A in den Schrank in das entsprechende Fach geworfen werden.