Hochschule Flensburg
Fachbereich 3
Angewandte Informatik

Jan Christiansen

Hochschule Flensburg
University of Applied Sciences

4. Übung zur Vorlesung „Fortgeschrittene funktionale Programmierung“

Funktionen und Faltung
Labor am Freitag, 21. Oktober 2016, 12:15 Uhr

Aufgabe 1 - Mengen als Funktionen

Definieren Sie das folgende Typsynonym.

$$\textbf{type}\ \mathit{IntSet} = \mathit{Int} \to \mathit{Bool}$$

Der Typ *IntSet* soll genutzt werden, um eine Menge von ganzen Zahlen zu repräsentieren. Implementieren Sie die folgenden Funktionen.

- Die Konstante $\mathit{empty} :: \mathit{IntSet}$ liefert eine leere Menge.
- Die Funktion $\mathit{isElem} :: \mathit{IntSet} \to \mathit{Int} \to \mathit{Bool}$ testet, ob ein Element in der Menge enthalten ist.
- Die Funktion $\mathit{singleton} :: \mathit{Int} \to \mathit{IntSet}$ liefert eine Menge mit einem Element.
- Die Funktion $\mathit{insert} :: \mathit{Int} \to \mathit{IntSet} \to \mathit{IntSet}$ fügt ein Element zu einer Menge hinzu.
- Die Funktion $\mathit{remove} :: \mathit{Int} \to \mathit{IntSet} \to \mathit{IntSet}$ entfernt ein Element aus einer Menge.
- Die Funktion $\mathit{union} :: \mathit{IntSet} \to \mathit{IntSet} \to \mathit{IntSet}$ vereinigt zwei Mengen.
- Die Funktion $\mathit{intersect} :: \mathit{IntSet} \to \mathit{IntSet} \to \mathit{IntSet}$ berechnet den Schnitt von zwei Mengen.
- Die Funktion $\mathit{listToSet} :: [\mathit{Int}] \to \mathit{IntSet}$ liefert zu einer Liste von ganzen Zahlen die Menge, die die gleichen Elemente enthält.

Können Sie *insert* mit Hilfe der anderen Funktionen definieren?

Aufgabe 2 - RNA-Sequenzen

Eine RNA-Sequenz besteht aus den Basen Uracil, Adenin, Cytosin und Guanin, die jeweils durch die Buchstaben U, A, C und G dargestellt werden. Schreiben Sie eine Funktion $\mathit{parseBase} :: \mathit{Char} \to \mathit{Maybe\ Base}$, die ein Zeichen in eine Base abbildet. Dabei soll der Datentyp *Maybe* signalisieren, ob es sich bei dem Zeichen um eine Base gehandelt hat. Schreiben Sie eine Funktion $\mathit{parseRNA} :: \mathit{String} \to \mathit{Maybe\ RNA}$, die eine Zeichenketten in eine RNA-Sequenz umwandelt.

Überlegen Sie sich, welche Funktion einer Faltung des Datentyp *Maybe* entspricht und implementieren Sie diese Funktion. Implementieren Sie auf Basis der Faltung einen Infixoperator $(>\!\!>\!\!-) :: \mathit{Maybe}\ a \to (a \to \mathit{Maybe}\ b) \to \mathit{Maybe}\ b$. Definieren Sie außerdem eine Funktion $\mathit{result} :: a \to \mathit{Maybe}\ a$ und verwenden $(>\!\!>\!\!-)$ und *result* zur Implementierung der Funktion *parseRNA*.

Aufgabe 3 - Parsen mit Fehlermeldungen

Wir wollen in dieser Aufgabe einen Parser für RNA-Sequenzen schreiben, die uns genauere Auskunft über den Fehler gibt, der beim Parsen aufgetreten ist. Definieren Sie dazu eine Funktion $parseBase :: Char \to Either\ String\ Base$. Dabei soll der Datentyp *String* für Fehlermeldungen genutzt werden.

Definieren Sie eine Faltung für den Datentyp *Either*. Impementieren Sie auf Basis der Faltung einen Infixoperator $(>> -) :: Either\ c\ a \to (a \to Either\ c\ b) \to Either\ c\ b$. Implementieren Sie außerdem eine Funktion $result :: a \to Either\ c\ a$. Was müssen Sie an der Funktion *parseRNA* ändern, um Fehlermeldungen beim Parsen weiterzuleiten.

Aufgabe 4 - Refactoring der Graphik

Definieren Sie für den Datentyp *XML* vom vorherigen Aufgabenzettel die folgende Funktion.

$$foldXML :: (String \to a) \to (String \to [Attr] \to [a] \to a) \to XML \to a$$

Diese Funktion stellt ebenfalls die Faltung des Datentyp *XML* dar. Implementieren Sie die Funktion *xmlToString* mit Hilfe der Faltung.

Hinweis: Bitte geben Sie die Lösungen der Aufgaben per Mail oder auf Papier bis zum 28.10. ab. Die Papierlösungen können im ersten Stock von Haus A in den Schrank in das entsprechende Fach geworfen werden.