Institut für Informatik **Ludwig-Maximilians-Universität München**
Prof. Dr. François Bry

Programmierung und Modellierung, SoSe 16
Übungsblatt 9

Abgabe: bis Mo 13.06.2016 10:00 Uhr

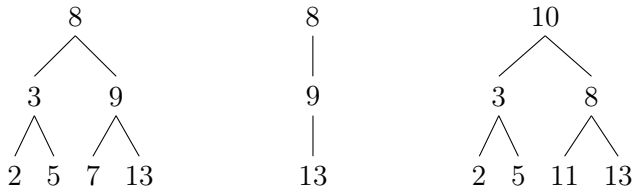
Besprechung: ab Di 14.06.2016

Aufgabe 9-1 Binäre Suchbäume

In der Vorlesung wurde die folgende Definition für Binärbäume eingeführt: Ein Binärbaum besteht aus Knoten und Blättern. Ein Blatt ist entweder leer oder hat einen Wert, während ein Knoten aus einem Wert und zwei Kindern besteht, die sowohl Blätter als auch wieder Knoten sein können. In Haskell lässt sich diese Definition wie folgt formulieren:

```
data BB a = L | K a (BB a) (BB a) deriving (Show)
```

Wir betrachten in dieser Aufgabe eine besondere Art von Binärbäumen: die binären Suchbäume. In einem binären Suchbaum muss gelten, dass die Werte im linken Teilbaum immer kleiner als der Wert des Vaterknotens sind, während die Werte im rechten Teilbaum immer größer als der Wert der Vaterknotens sind. Im folgenden sind drei Beispiele für Binärbäume gegeben:
Der linke Baum erfüllt alle Eigenschaften eines binären Suchbaums, wie auch der Listen ähnliche mittlere Baum. Im rechten Teilbaum verletzten sowohl die Wurzel, als auch der rechte Knoten mit dem Wert `8` in der ersten Ebene die Suchbaumeigenschaft.



a) Gegeben sei die Menge `[5, 7, 12, 3, 1, 9]`. Geben Sie einen möglichen binären Suchbaum für diese Menge als Ausdruck des Typs `BB` an.

b) Implementieren Sie eine Funktion `isBinarySearchTree :: Ord a => BB a -> Bool`, die überprüft, ob ein übergebener Binärbaum ein Suchbaum ist.

c) Implementieren Sie eine Funktion `depth :: (Num a, Ord a) => BB t -> a`, welche die Tiefe des Baums berechnet.

d) Implementieren Sie eine Funktion `insert :: Ord a => a -> BB a -> BB a`, die ein übergebenes Element an eine mögliche Position im übergebenen binären Suchbaum einfügt.

e) Implementieren Sie mit Hilfe Ihres Ergebnisses aus Teilaufgabe d) eine Funktion `buildTree :: Ord a => [a] -> BB a`, die aus einer Liste von Werten einen möglichen binären Suchbaum erstellt. Verwenden Sie hierfür die Funktion `foldr`.

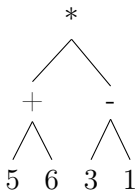
f) Das Aussehen der binären Suchbäume ist abhängig von der Reihenfolge der Elemente in der Liste, aus der sie konstruiert werden. Trotzdem sollen binäre Suchbäume vergleichbar sein. Implementieren Sie für den Datentyp `BB` die Typklasse `Eq`, so dass zwei binäre Suchbäume genau dann gleich sind, wenn beide binäre Suchbäume sind und die gleiche Menge an Werten repräsentieren.

Aufgabe 9-2 Mathematische Terme als Bäume

In der Vorlesung wurde bereits motiviert, dass man Terme mit Hilfe von Bäumen darstellen kann. Wenn man die Menge der Operatoren auf ein- und zweistellige Operatoren einschränkt, dann reichen bereits Binärbäume. In dieser Aufgabe soll ein Datentyp `Term` entwickelt werden, der dazu benutzt werden kann, beliebige Terme (mit den oben genannten Einschränkungen) darzustellen und auszuwerten.

a) Geben Sie eine mögliche Definition für diesen Datentyp an. Fürs erste ist es ausreichend, wenn die binären Operation `+`, `-`, $*$ und `/` ausgedrückt werden können.

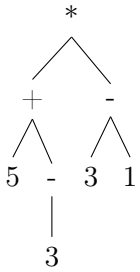
Das folgende Beispiel zeigt den Binärbaum für den Term $(5 + 6) * (3 - 1)$, sowie die Darstellung in dem zu entwickelnden Datentyp.




```
BinaryTerm Times
  (BinaryTerm Plus (C 5) (C 6))
  (BinaryTerm Minus (C 3) (C 1))
```

b) Implementieren Sie eine Funktion `eval :: Integral a => Term a => a`, welche einen übergebenen Term auswertet.

c) Erweitern Sie Ihre Lösung aus den Teilaufgaben a) und b) so, dass auch der einstellige Operator `-`, d.h. die Negation einer Zahl, unterstützt wird. Das Beispiel zeigt den Binärbaum für den Term $(5 + (-3) * (3 - 1))$ und die gewünschte Represenation als Haskell-Datentyp.




```
BinaryTerm Times
  (BinaryTerm Plus (C5)
    (UnaryTerm Minus (C 3))
  )
  (BinaryTerm Minus (C 3) (C 1))
```

d) Implementieren Sie für Ihren Datentyp einen (einfachen) Termvereinfacher. Die folgende Liste zeigt Beispiele für mögliche Vereinfachungen, die implementiert werden könnten. Selbstverständlich können Sie auch eigene Ideen implementieren.

- $-(-n) = n$
- $(x + (-y)) = x - y$
- $(x - (-y)) = x + y$