Praktische Informatik 3 WS 06/07

6. Übungsblatt

Ausgabe: 23.01.07
Abgabe: 06.02.07

Christoph Lüth <cxl>
Matthias Berger <tokio>
Klaus Hartke <hartke>
Cui Jian <ken>
Friederike Jolk <rikej>
Christian Maeder <maeder>
Diedrich Wolter <dwolter>

16 *Bäume umdrehen* *5 Punkte*

Genau wie eine Liste kann man einen Baum auch umdrehen (allerdings nur seitlich, ansonsten ragen die Blätter in die Luft). Die Funktion `flip` soll bei einem Baum die Reihenfolge der Knoten umdrehen. Sie hat folgende Signatur

```
flip :: Tree a-> Tree a
```

und soll folgende Spezifikation erfüllen:

```
inorder (flip t) = rev (inorder t)
```

Geben Sie eine Implementation von `flip` an, und zeigen Sie, dass `flip` die Spezifikation erfüllt.

17 *Wahrheit oder Lüge?* *10 Punkte*

Zwischen den Studenten Sebastian Schlaudrauf und Bastian Blitzmerker ist ein erbitterter Streit entbrannt: beide haben während der Vorlesung verschiedene Behauptungen mitgeschrieben, aber vergessen, welche davon wahr, und welche falsch sind!

Helfen Sie den beiden, und zeigen oder widerlegen Sie folgende Behauptungen:

1. `rev (rev x)= x`
2. `map f (map g x)= map (f. g) x`
3. `postorder (flip t) = preorder t`
4. `preorder t = rev (postorder t)`

`preorder` und `postorder` ist die Traversion der Knoten des Baumes in Prä- und Postordnung (siehe Vorlesung vom 05.12.06).

Hinweis: Das Lemma `rev (xs++ ys)= rev ys ++ rev xs` aus der Vorlesung könnte sich als hilfreich erweisen.

18 *Korrekt gefiltert* *5 Punkte*

Im Jahre 2013 soll die erste deutsche bemannte Marsmission gestartet werden. Die Steuersoftware für die Marsmission ist natürlich in Haskell geschrieben, weil Korrektheit in der Raumfahrt oberstes Gebot ist und man Haskell-Programme so einfach verifizieren kann.

Ein Teil der Steuersoftware benutzt binäre Bäume (wofür ist nicht klar, schließlich gibt es keine Bäume auf dem Mars, aber das ist hier nicht weiter relevant). Dazu wird unter anderem eine Filter-Funktion für binäre Bäume mit folgender Signatur benötigt:

```
tfilter :: (a-> Bool)-> Tree a-> Tree a
```

`tfilter p t` soll einen Baum zurückliefern, der genau alle Knoten von `t` enthält, welche das Prädikat `p` erfüllen, oder mit anderen Worten, `tfilter` soll analog der Funktion `filter` für Listen arbeiten. Sie sollen diese Funktion implementieren, und darüberhinaus Ihre Implementation als korrekt beweisen.

1. Geben Sie eine *Spezifikation* der Funktion `tfilter` an. (*2 Punkte*)
2. *Implementieren* Sie die Funktion `tfilter`. (*1 Punkte*)
3. Beweisen Sie die *Korrektheit Ihrer Implementation*, d.h. dass die Funktion die Spezifikation erfüllt. *(2 Punkte)*

Hinweis: Die Implementation von `tfilter` ist ähnlich der Implementation des Löschens von Knoten in einem (balancierten) Baum (siehe Vorlesung vom 12.12.06).

Dies ist Version 1.0 vom 2007/01/23 13:20:57.