Universität Bonn
Institut für Informatik III
Jun.-Prof. Dr. Janis Voigtländer[1]
Dipl.-Math. Stefan Mehner

18.05.2015

Lösungseinreichung bis
02.06.2015, 07:00 Uhr

Übungen
Deskriptive Programmierung
SS 15

Blatt 5

Aufgabe 22 (zu lösen/einzureichen über Autotool, ∞ Fehlversuche erlaubt, [**5P**])**.**

Aufgabe 23 (einzureichen über eCampus, als Text-/Quelldatei, [**6P**])**.**
Betrachten Sie die folgenden Definitionen:

```
elems :: [Int] → [Int]
elems xs = foldr (λx ys → if elem x ys then ys else x : ys) [] xs
elem :: Int → [Int] → Bool
elem x xs = foldr (λy b → x == y || b) False xs
```

(a) Zu welchen Vergleichen, und in welcher Reihenfolge, mittels == führt der Aufruf *elems* $[1, 2, 3, 3, 4, 5, 5, 6]$?

(b) Finden Sie systematisch heraus, für welche Eingaben die Funktion *elems* gleiche Ausgabelisten wie die vordefinierte Funktion *nub* erzeugt (**import** Data.List), und für welche Eingaben (mit Typ [Int]) unterschiedliches Verhalten vorliegt.

Geben Sie Funktionen f und g an, so dass $elems = f \,.\, nub \,.\, g$ (auf Typ [Int]). Schreiben Sie einen QuickCheck-Test, der Ihre Behauptung bestätigt.

Aufgabe 24 (einzureichen über eCampus, als Quelldatei(en), [**5 Extrapunkte**])**.**
Erweitern Sie Ihr Programm aus Aufgabe 1 am Beginn der Lehrveranstaltung, so dass ein interaktives Abfragen von Aufgaben stattfindet, analog zu *„taskIO.hs“* aus der Vorlesung. Sorgen Sie außerdem dafür, dass keine Wiederholungen vorkommen (also unter den 20 abgefragten Aufgaben mit Sicherheit keine Aufgabe mehrmals).

[1] Bei Fragen wenden Sie sich bitte via E-Mail an Janis Voigtländer (jv@informatik.uni-bonn.de).

Hinweis: Bei den Autotool-Aufgaben zu Parserkombinatoren soll es wirklich darum gehen, die jeweiligen Parser mit den Mitteln/Operationen der Parser-Bibliothek auszudrücken. Insbesondere wird ein bloßes Einlesen der gesamten Eingabe als Ganzes, um sie dann jenseits der Parserabstraktion mit anderen Mitteln zu analysieren, nicht als korrekt bewertet. Nicht erlaubt ist also etwa eine triviale (ansonsten korrekte) „Lösung“ von Aufgabe 25 wie folgt:

$$
\begin{aligned}
&\mathit{exactly} :: \mathsf{String} \to \mathsf{Parser}\ () \\
&\mathit{exactly}\ s = \mathbf{do}\ s' \leftarrow \mathit{many}\ \mathit{item} \\
&\qquad\qquad\qquad\qquad \mathbf{if}\ s == s'\ \mathbf{then}\ \mathit{yield}\ ()\ \mathbf{else}\ \mathit{failure}
\end{aligned}
$$

Außerdem gilt, dass kein Zugriff auf die jeweilige interne Repräsentation von Parser-Typen besteht.

Aufgabe 25 (zu lösen/einzureichen über Autotool, 5 Fehlversuche erlaubt, [**3P**])**.**

Aufgabe 26 (zu lösen/einzureichen über Autotool, 5 Fehlversuche erlaubt, [**5P**])**.**

Aufgabe 27 (zu lösen/einzureichen über Autotool, ∞ Fehlversuche erlaubt, [**7P**])**.**

Hinweis: Für die folgende Autotool-Aufgabe gilt eine Ausnahme, dass beliebig viele Fehlversuche erlaubt sind. Es ist dennoch ratsam, die Aufgabe zunächst mit einem lokal installierten `GHC` zu lösen, erst dann hochzuladen. Dabei sollten Sie auch von der Möglichkeit Gebrauch machen, den Code zu kompilieren (statt nur interpretiert auszuführen), und so vor allem bei Verwendung der Compileroption `-O2` von einer schnelleren Ausführung der Testsuite zu profitieren. Als weitere Compileroption sollten Sie `-main-is Blueprint.main` übergeben.

Aufgabe 28 (zu lösen/einzureichen über Autotool, ∞ Fehlvers. erlaubt, [**6 Extrapunkte**])**.**