Universität Bremen

Praktische Informatik 3 WS 16/17

Christoph Lüth
Tobias Brandt
Tristan Bruns
Johannes Ganser
Berthold Hoffmann
Alexander Kurth

8. Übungsblatt

Ausgabe: 13.12.16
Abgabe: 23.12.16, 12:00 Uhr

8.1 *Erste Schritte* *10 Punkte*

Während der Vorlesung haben Sie folgende Behauptungen mitgeschrieben. Leider waren Sie während der Vorlesung zu sehr durch lustige Katzenvideos auf Ihrem Handy abgelenkt, um sich zu notieren ob diese gelten oder falsch sind:

$$\texttt{length (map f xs)} = \texttt{length xs} \tag{1}$$
$$\texttt{sum (xs++ ys)} = \texttt{sum xs+sum ys} \tag{2}$$
$$\texttt{filter f (reverse xs)} \neq \texttt{filter f xs} \tag{3}$$
$$\texttt{take n (take m xs)} = \texttt{take (n+m) xs} \tag{4}$$
$$\texttt{filter f (takeWhile f xs)} = \texttt{takeWhile f xs} \tag{5}$$

Beweisen Sie diese Behauptungen, oder widerlegen Sie sie durch ein Gegenbeispiel. Konsultieren sie hierzu falls nötig die Implementierung der verwendeten Funktionen in der Haskell Standard Bücherei.

8.2 *Haskell Space Programm* *10 Punkte*

Raumfahrt ist eine teure Angelegenheit, und an so ziemlich jeder Stelle kann etwas schiefgehen. Da ist es schön, wenn man weiß, dass wenigstens die Software korrekt funktioniert.

Die Menge an Treibstoff, die eine Rakete benötigt, ist näherungsweise proportional zur Masse, die im Weltraum ankommt. Deshalb sollen Komponenten, die im Verlauf des Fluges nicht mehr benötigt werden (z.B. leere Treibstofftanks und die zugehörigen Triebwerke), von der Rakete entkoppelt werden, sobald sie ihren Zweck erfüllt haben.

Eine mehrstufige Rakete ist normalerweise so aufgebaut, dass sich ganz oben die Nutzlast befindet (z.B. eine Raumkapsel oder ein Satellit). Darunter befinden sich die Stufen, bestehend jeweils aus einem Entkoppler, einem Treibstofftank und einem Triebwerk.

Gezündet wird jeweils die unterste Stufe. Sobald ihr Treibstoff erschöpft ist, wird sie vom Rest der Rakete entkoppelt, und die Stufe darüber wird gezündet.

Wir modellieren die verschiedenen Bauteile einer Rakete als algebraischen Datentyp:

```
data Part = Payload | FuelTank | Engine | Decoupler
  deriving (Eq, Show)
```

Eine Rakete ist dementsprechend eine Liste aller Bauteile aus denen sie besteht:

```
type Rocket = [Part]
```

Ein Beispiel für eine zweistufige Rakete:

```
twoStageRocket = [Payload, Decoupler, FuelTank, Engine, Decoupler, FuelTank, Engine]
```

Die Funktion *makeRocket* konstruiert eine Rakete mit n Stufen:

```
makeRocket :: Int → Rocket
makeRocket n = extendRocket n [Payload]
```

```
stageParts = [Decoupler, FuelTank, Engine]

extendRocket :: Int → Rocket → Rocket
extendRocket n r
    | n ≤ 0 = r
    | otherwise = extendRocket (n − 1) (r ++ stageParts)
```

Es ist außerdem die Funktion *repList* gegeben, welche eine Liste n mal repliziert:

```
repList :: Int → [a] → [a]
repList n s
    | n ≤ 0 = []
    | otherwise = s ++ repList (n − 1) s
```

Beweisen sie nun folgende Behauptungen:

$$\texttt{r ++ repList n stageParts} == \texttt{extendRocket n r} \tag{6}$$

$$\texttt{makeRocket n} == \texttt{Payload : repList n stageParts} \tag{7}$$

? *Verständnisfragen*

1. Der Datentyp Stream α ist definiert als

   data Stream α = Cons α (Stream α)

   Gibt es für diesen Datentyp ein Induktionsprinzip? Ist es sinnvoll?

2. Welche nichtausführbaren Prädikate haben wir in der Vorlesung kennengelernt?

3. Wie kann man in einem Induktionsbeweis die Induktionsvoraussetzung stärken, und wann ist das nötig?

Änderungen:

- *Version 1.0 Ausgegebene Version*
- *Version 1.1 Formel für Behauptung (5) korrigiert.*