Universität Bremen

Praktische Informatik 3 WS 14/15

Christoph Lüth
Sandor Herms
Daniel Müller
Jan Radtke
Henrik Reichmann
Sören Schulze
Felix Thielke

9. Übungsblatt

Ausgabe: 16.12.14
Abgabe: 09.01.15

9.1 *Zum Warmwerden...* *5 Punkte*

Zeigen Sie folgende drei Behauptungen:

```
length (reverse xs) =length xs
map f xs ++ map f ys =map f (xs++ys)
map f (reverse xs) =reverse (map f xs)
```

Haskell Space Program

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.

In dieser Übung geben wir Ihnen zwei Haskell-Funktionen, und Sie sollen bei jeder dieser Funktionen die Korrektheit zeigen, indem Sie eine vorgegebene Eigenschaft beweisen.

9.2 *Mehrstufige Raketen* *8 Punkte*

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 stellen eine Rakete in Haskell als Liste von Bauteilen dar, wobei wir von oben nach unten verfahren.

```
data Part = Payload | FuelTank | Engine | Decoupler
          deriving (Eq, Show)
type Rocket = [Part]
```

Eine zweistufige Rakete hätte z. B. die Darstellung:[1]

```
exampleRocket :: Rocket
exampleRocket = [Payload, Decoupler, FuelTank, Engine, Decoupler, FuelTank, Engine]
```

Die Funktion makeRocket soll eine Rakete mit einer vorgegebenen Anzahl n an Stufen konstruieren:

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

extendRocket :: Int → Rocket → Rocket
extendRocket n r | n < 0     = error "You␣will␣not␣go␣to␣space␣today"
                 | n == 0    = r
                 | otherwise = extendRocket (n−1) (r ++ stageParts)
```

[1] Warum ist ein Entkoppler direkt unter der Nutzlast? Es könnte sich um eine Raumkapsel handeln, von der vor dem Wiedereintritt in die Atmosphäre alles abgekoppelt werden muss, um den Hitzeschild freizulegen.

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

1. Zeigen Sie zunächst per Induktion über n, dass für alle $n \geq 0$ und alle $r \in$ [Part] gilt:

$$r \mathbin{+\!\!+} \text{repList } n \text{ stageParts} = \text{extendRocket } n \text{ r}.$$

2. Zeigen Sie nun, dass für alle gilt $n \geq 0$ gilt:

$$\text{makeRocket } n = \text{Payload} : \text{repList } n \text{ stageParts}, \tag{1}$$

wobei:

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

9.3 *Entkoppeln* — *7 Punkte*

Jetzt wird's ernst. Die Rakete ist abgehoben, und der Treibstoff der ersten Stufe geht zur Neige. In letzter Sekunde kommen Zweifel auf, ob die für das Entkoppeln der untersten Stufe zuständige Funktion richtig funktioniert:

```
decouple :: Rocket → (Rocket, [Part])
decouple r = decouple' r [] []

decouple' :: [Part] → [Part] → [Part] → (Rocket, [Part])
decouple' []              acc1 acc2 = (acc1, acc2)
decouple' (Decoupler:r) acc1 acc2 = decouple' r (acc1 ++ acc2) [Decoupler]
decouple' (p:r)           acc1 acc2 = decouple' r acc1 (acc2 ++ [p])
```

Die Funktion decouple soll die Rakete am untersten Entkoppler aufteilen. Es wird ein Tupel zurückgegeben, dessen erste Komponente die verbleibende Rakete und dessen zweite Komponente den entkoppelten Teil darstellt (inklusive des Entkopplers).

1. Zeigen Sie zunächst für alle $a, b, r, t, u \in$ [Part] folgende Eigenschaft per Induktion über r:

$$(a,b) = \text{decouple' } r \ t \ u \quad \Rightarrow \quad t \mathbin{+\!\!+} u \mathbin{+\!\!+} r = a \mathbin{+\!\!+} b.$$

2. Zeigen Sie nun für alle $a, b, r \in$ Rocket folgende Eigenschaft:

$$(a,b) = \text{decouple } r \quad \Rightarrow \quad a \mathbin{+\!\!+} b = r. \tag{2}$$

Beware of bugs in the above code, I have only proved it correct, not tried it.
– Donald E. Knuth

Needs more struts.
– Jebediah Kerman

? *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?
4. Gibt es einen Weihnachtsmann?