WiSe 2015/2016

Funktionale Programmierung

11. Übungsblatt (Abgabe: Mi., den 20.Januar, um 10:10 Uhr)

Prof. Dr. Margarita Esponda

Ziel: SKI-Kombinatoren und Beweisen von Programmeigenschaften.

1. Aufgabe (4 Punkte)

Die Collatz-Folge, die im Jahr 1937 von Lothar Collatz entdeckt worden ist, wird wie folgt definiert.

Die Collatz-Zahl C_{i+1} mit **i>0** wird wie folgt berechnet:

$$C_{i+1} = \begin{cases} \frac{C_i}{2} & \text{wenn } C_i \text{ gerade} \\ C_i \cdot 3 + 1 & \text{wenn } C_i \text{ ungerade} \end{cases}$$

D.h. die Folge startet mit einer beliebigen Zahl **n**. Wenn **n** gerade ist, ist die Folgezahl gleich $\frac{n}{2}$ und wenn **n** ungerade ist, wird die Folgezahl gleich $(n \cdot 3 + 1)$.

Die Vermutung ist, dass unabhängig davon mit welcher natürlichen Zahl gestartet wird, die Folge immer mit dem Zahlenzyklus 4, 2, 1 endet. Eine Vermutung die bis jetzt noch nicht bewiesen worden ist.

Definieren Sie unter Verwendung der **fix** Funktion aus der Vorlesungsfolien und Anonyme Funktionen in Haskell eine rekursive Funktion für die Berechnung der Collatz-Folge.

Anwendungsbeispiel:

(**fix collatz**) 10 => [10, 5, 16, 8, 4, 2, 1]

2. Aufgabe (6 Punkte)

Zeigen Sie, dass folgende Lambda- und Kombinatoren-Ausdrücke äquivalent sind.

$$\lambda x.\lambda y.(yy) \equiv K(SII)$$
$$\lambda x.y(xy) \equiv S(Ky)(SI(Ky))$$

Verwenden Sie dafür zuerst die Transformations- bzw. Eliminierungsregeln aus der Vorlesung.

Begründen Sie die einzelnen Schritte indem sie die verwendete Transformations-/ Eliminierungsregeln angeben.

3. Aufgabe (4 Punkte)

Zeigen Sie, dass folgende Lambda- und Kombinatoren-Ausdrücke äquivalent sind.

$$\lambda s.\lambda x.\, s(s(x)) \equiv S(S(KS)(S(KK)I))(S(S(KS)(S(KK)I))(KI))$$

Verwenden Sie die Funktionsapplikation auf zwei Argumente, um die Äquivalenz zu zeigen.

Begründen Sie die einzelnen Schritte indem sie die verwendete Reduktionsegeln angeben.

4. Aufgabe (4 Punkte)

Betrachten Sie folgende Funktionsdefinitionen:

```
map f   []     = []
map f (x:xs) = (f x): map f xs

(++)  []  ys  = ys
(++) (x:xs) ys = x:(xs ++ ys)
```

Beweisen Sie mittels struktureller Induktion über **xs** folgende Eigenschaft:

map f xs ++ map f ys == map f (xs ++ ys)

5. Aufgabe (12 Punkte)

Betrachten Sie folgende Funktionsdefinitionen:

```
(++)   []      ys        = ys
(++)  (x:xs)  ys        = x : (xs ++ ys)

reverse   []            =  []
reverse  (x:xs)         =  reverse xs  ++  [x]

elem  x  []                      = False
elem  x  (y:ys) | x==y           =  True
                | otherwise      =  elem  ys

dropWhile p []  =  []
dropWhile p (x:xs)  =  if  p x
                       then dropWhile p xs
                       else (x:xs)

takeWhile p []  =  []
takeWhile p (x:xs)   = if p x
                       then x:(takeWhile p xs)
                       else []
```

Beweisen Sie mittels struktureller Induktion über die Liste **xs**, dass für jede endliche Listen **xs** und **ys** folgende Gleichungen gelten:

a) (takeWhile p xs) ++ (dropWhile p xs) = xs

b) reverse (xs ++ ys) = reverse ys ++ reverse xs

c) elem a (xs ++ ys) = elem a xs || elem a ys