

# Programmierung

## 04. Übungsblatt

Zeitraum: 02. – 06. Mai 2016

*Beachten Sie die Vorlesungs- bzw. Übungsverlegungen am 05. und 06. Mai!*

### Übung 1 (AGS 12.1.34)

Gegeben ist der Datentyp

```
data Tree = Node Int [Tree]
```

eines Baumes, bei dem jeder Knoten eine beliebige Anzahl an Kindbäumen haben kann (gegeben in einer Liste vom Typ [Tree]).

- Geben Sie die Definition der Funktion `noLeaves :: Tree -> Int` an, die ermittelt, wie viele Blattknoten ein gegebener Baum vom Typ `Tree` enthält. Ein Blattknoten ist ein Knoten mit einer leeren Liste an Kindbäumen.
- Geben Sie die Definition der Funktion `even :: Tree -> Bool` an, die zu einem gegebenen Baum vom Typ `Tree` ermittelt, ob jeder Knoten eine gerade Anzahl an Nachfolgern hat und in diesem Fall `True` zurückgibt (ansonsten `False`). Sie dürfen dabei auf die Funktion `length :: [Int] -> Int` zurückgreifen, die die Länge einer Liste ermittelt.

### Übung 2 (AGS 12.2.12)

- Gegeben seien folgende Terme über dem Rangalphabet  $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ :

$$t_1 = \sigma(\sigma(x_1, \alpha), \sigma(\gamma(x_3), x_3)),$$
$$t_2 = \sigma(\sigma(\gamma(x_2), \alpha), \sigma(x_2, x_3)).$$

Wenden Sie den Unifikationsalgorithmus auf die Terme  $t_1$  und  $t_2$  an. Geben Sie anschließend den von Ihnen bestimmten allgemeinsten Unifikator an.

- Geben Sie zwei weitere Unifikatoren an.
- Geben Sie zwei Terme  $t_1$  und  $t_2$  (über einem beliebigen Alphabet) an, so dass im Laufe der Anwendung des Unifikationsalgorithmus auf  $t_1$  und  $t_2$  der Occur-Check fehlschlägt!

### Übung 3 (AGS 12.3.20)

Folgende Definitionen seien gegeben:

```
1 foo :: [Int] -> [Int]
2 foo [] = []
3 foo (x:xs) = x : x : (-1) : foo xs
4
5 sum :: [Int] -> Int
6 sum [] = 0
7 sum (x:xs) = x + sum xs
8
9 length :: [Int] -> Int
10 length [] = 0
11 length (x:xs) = 1 + length xs
```

Die folgende Aussage soll mittels struktureller Induktion über Listen bewiesen werden:

(A): Für jede Liste  $xs :: [Int]$  gilt:

$$\text{sum (foo xs)} = 2 * \text{sum xs} - \text{length xs} .$$

Bearbeiten Sie die folgenden Teilaufgaben; geben Sie bei jeder Umformung die benutzte *Definition*, bzw. die *Induktionsvoraussetzung* an; quantifizieren Sie alle Variablen.

- Zeigen Sie den Induktionsanfang.
- Geben Sie die Induktionsvoraussetzung *vollständig* an.
- Zeigen Sie den Induktionsschritt.

#### Übung 4 (AGS 12.3.11)

Folgende Definitionen seien gegeben:

```
1 data Tree = Leaf Float | Branch Float Tree Tree
2
3 add :: Tree -> Float -> Tree
4 add (Leaf x) a = Leaf (x+a)
5 add (Branch x l r) a = Branch (x+a/3) (add l (a/3)) (add r (a/3))
6
7 rev :: Tree -> Tree
8 rev (Leaf x) = Leaf x
9 rev (Branch x l r) = Branch x (rev r) (rev l)
10
11 sum :: Tree -> Float
12 sum (Leaf x) = x
13 sum (Branch x l r) = x + (sum l) + (sum r)
```

Zeigen Sie für die oben aufgeführten Definitionen mit Hilfe der strukturellen Induktion, dass die folgende Gleichung für einen beliebigen Baum  $t :: Tree$  und eine beliebige Zahl  $a :: Float$  erfüllt ist:

$$\text{sum (add t a)} = (\text{sum (rev t)}) + a$$

Geben Sie bei Umformungen die jeweils benutzten Gesetzmäßigkeiten / Definitionen an.

#### Zusatzaufgabe 1 (AGS 12.3.22 ★)

Folgende Definitionen seien gegeben:

```
1 data Tree a = Branch a (Tree a) (Tree a) | Leaf a
2
3 -- | Knotenbeschriftungen des Baumes in preorder
4 p :: Tree a -> [a]
5 p (Leaf x) = [x]
6 p (Branch x s t) = [x] ++ (p s ++ p t)
7
8 d :: Tree a -> Tree a -> Tree a
9 d (Leaf x) u = Branch x u u
10 d (Branch x s t) u = Branch x s (d t u)
```

Sei  $a$  ein beliebiger Typ. Die folgende Aussage soll mittels Induktion über  $Tree\ a$  bewiesen werden:

(A) Für jedes  $t :: Tree\ a$  und für jedes  $u :: Tree\ a$  gilt:

$$p (d\ t\ u) = p\ t\ ++\ (p\ u\ ++\ p\ u).$$

Nutzen sie folgende Eigenschaft:

(B) Für alle Listen  $xs$ ,  $ys$ ,  $zs :: [a]$  gilt:

$$xs ++ (ys ++ zs) = (xs ++ ys) ++ zs.$$

Bearbeiten Sie die folgenden Teilaufgaben; geben Sie bei jeder Umformung die benutzte *Definition*, *Eigenschaft*, bzw. die *Induktionsvoraussetzung* an; quantifizieren Sie alle Variablen. *Hinweis:* Für den Beweis von (A) genügt Induktion über der Struktur des Baumes  $t$ .

(a) Zeigen Sie den Induktionsanfang.

(b) Zeigen Sie den Induktionsschritt inklusive der Induktionsvoraussetzung.

### Zusatzaufgabe 2 (AGS 12.3.6 \*)

Folgende Definitionen seien gegeben:

```
1 rev :: [t] -> [t]
2 rev [] = []
3 rev (a:xs) = rev xs ++ [a]
4
5 map :: (t->t) -> [t] -> [t]
6 map f [] = []
7 map f (a:xs) = f a : map f xs
```

Zeigen Sie unter Verwendung der oben aufgeführten Definitionen mit Hilfe der Induktion über Listen, dass folgende Gleichung für beliebige Listen  $xs$  gilt:

$$\text{rev} (\text{map } f \text{ } xs) = \text{map } f (\text{rev } xs)$$

Beim Beweis dürfen Sie die Beziehungen  $\text{map } f (l1 ++ l2) = \text{map } f l1 ++ \text{map } f l2$  und  $\text{map } f [a] = [f a]$  für beliebige Listen  $l1, l2$  und Funktionen  $f :: t \rightarrow t$  sowie Elemente  $a$  mit  $a :: t$  nutzen. Geben Sie bei Umformungen die jeweils benutzten Gesetzmäßigkeiten und Definitionen an.