6. Übung (13.–17. Mai 2019)

Programmierung

Aufgabe 1 (AGS 12.3.19)

Folgende Definitionen seien gegeben:

```
1 data BinTree a = Branch a (BinTree a) (BinTree a) | Leaf a
2
3 mirror :: BinTree a -> BinTree a
4 mirror (Branch x t1 t2) = Branch x (mirror t2) (mirror t1)
5 mirror (Leaf x) = Leaf x
6
7 yield :: BinTree a -> [a]
8 yield (Branch _ t1 t2) = yield t1 ++ yield t2
9 yield (Leaf x) = [x]
```

Zeigen Sie unter Verwendung der obigen Definitionen durch strukturelle Induktion die Gültigkeit der Gleichung

(A): Für jeden Typ `a` und jeden Baum `t :: BinTree a` gilt

`reverse (yield t) = yield (mirror t)` .

Nutzen Sie folgende Eigenschaften: Für alle Typen `a`, Werte `x :: a` und Listen `xs, ys :: [a]` gilt

(E1): `reverse [x] = [x]`, und

(E2): `reverse (xs ++ ys) = reverse ys ++ reverse xs`.

Geben Sie bei jeder Umformung die benutzte *Definition*, *Eigenschaft* bzw. *Induktionsvoraussetzung* an; quantifizieren Sie alle Variablen.

Aufgabe 2 (AGS 12.3.13)

Folgende Definitionen seien gegeben:

```
1  data IntTree = Leaf Int | Branch IntTree IntTree
2
3  add :: IntTree -> Int
4  add (Leaf a)         = a
5  add (Branch t1 t2)   = add t1 + add t2
6
7  sub :: IntTree -> Int
8  sub (Leaf a)         = a
9  sub (Branch t1 t2)   = sub t1 - sub t2
10
11 neg :: Int -> IntTree -> IntTree
12 neg i (Leaf a)       = Leaf (a * i)
13 neg i (Branch t1 t2) = Branch (neg i t1) (neg (-i) t2)
```

Zeigen Sie für die oben aufgeführten Definitionen mit Hilfe der Induktion über Bäume, dass die folgende Gleichung für jeden Baum `t :: IntTree` und jede ganze Zahl `i :: Int` erfüllt ist:

```
add (neg i t) = i * sub t .
```

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

Aufgabe 3 (AGS 12.4.1 ⋆)

(a) Bestimmen Sie für jeden der folgenden λ-Terme t die Mengen $FV(t)$ und $GV(t)$:

- $(\lambda x.x\ y)\ (\lambda y.y)$
- $(\lambda x.(\lambda y.z\ (\lambda z.z\ (\lambda x.y))))$
- $(\lambda x.(\lambda y.x\ z\ (y\ z)))\ (\lambda x.y\ (\lambda y.y))$

(b) Reduzieren Sie die folgenden λ-Terme zu Normalformen. Schreiben Sie – bevor Sie einen Ableitungsschritt ausführen – für die relevanten (Teil-)Ausdrücke die Mengen der freien bzw. der gebundenen Vorkommen von Variablen auf.

- $(\lambda x.(\lambda y.x\ z\ (y\ z)))\ (\lambda x.y\ (\lambda y.y))$
- $(\lambda x.(\lambda y.(\lambda z.z)))\ x\ (+\ y\ 1)$
- $(\lambda x.(\lambda y.x\ (\lambda z.y\ z)))\ (((\lambda x.(\lambda y.y))\ 8)\ (\lambda x.(\lambda y.y)\ x))$
- $(\lambda h.(\lambda x.h\ (x\ x))\ (\lambda x.h\ (x\ x)))\ ((\lambda x.x)\ (+\ 1\ 5))$
- $(\lambda f.(\lambda a.(\lambda b.f\ a\ b)))\ (\lambda x.(\lambda y.x))$

Zusatzaufgabe 1 (AGS 12.3.22)

```
1 data BinTree a = Branch a (BinTreee a) (BinTree a) | Leaf a
2
3 p :: BinTree a -> [a]
4 p (Leaf x)       = [x]
5 p (Branch x s t) = [x] ++ (p s ++ p t)
6
7 d :: BinTree a -> BinTree a -> BinTree a
8 d (Leaf x) u       = Branch x u u
9 d (Branch x s t) u = Branch x s (d t u)
```

Zeigen Sie unter Verwendung der obigen Definitionen durch strukturelle Induktion die Gültigkeit der Gleichung

$$\texttt{p (d t u)} = \texttt{p t ++ (p u ++ p u)} \tag{A}$$

für jeden Typ `a` und alle Bäume `t, u :: BinTree a`. Sie dürfen dabei nutzen, dass für alle Typen `a` und alle Listen `xs, ys, zs :: [a]` gilt:

$$\texttt{xs ++ (ys ++ zs)} = \texttt{(xs ++ ys) ++ zs} \tag{B}$$

Hinweis: Für den Beweis von (A) genügt Induktion über der Struktur des Baumes `t`.