LV „ Programmierung “
Prof. Dr. H. Vogler / Dipl.-Math. R. Vater

SS 09

Aufgabenblatt zur 5. Übung

Zeitraum: 11.05. bis 15.05.2009

Achtung: Vorlesungsraumverlegung: einmalig am Donnerstag, dem 14.05.2009, 5. DS
von AUDIMAX nach ZEU/LICH/H.

1. Aufgabe: AGS 11.35*

Folgende Definitionen seien gegeben:

```
rev :: [t] -> [t]
rev []     = []
rev (a:xs) = (rev xs) ++ [a]

map :: (t->t) -> [t] -> [t]
map f []     = []
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:

`rev (map f xs) = map f (rev xs)`

Beim Beweis dürfen Sie die Beziehungen `map f (l1++l2) = (map f l1) ++ (map f l2)` und `map f [a] = [f a]` für beliebige Listen `l1`,`l2` und Funktionen `f : t->t` sowie Elemente `a` mit `a :: t` nutzen. Geben Sie bei Umformungen die jeweils benutzten Gesetzmäßigkeiten/Definitionen an.

2. Aufgabe: (AGS 11.31)

Folgende Definitionen seien gegeben:

```
data Tree = Leaf Int | Branch Int Tree Tree
```

```
prod:: [Int] -> Int
prod [] = 1
prod (a:xs) = a * (prod xs)

prodTree:: Tree -> Int
prodTree (Leaf x) = x
prodTree (Branch x y z) = x * (prodTree y) * (prodTree z)

collapse:: Tree -> [Int]
collapse (Leaf x) = [x]
collapse (Branch x y z) = (collapse y) ++ [x] ++ (collapse z)
```

Zeigen Sie unter Verwendung der oben aufgeführten Definitionen durch strukturelle Induktion die Gültigkeit der folgenden Gleichung:

`prodTree t = prod (collapse t)`

Gehen Sie davon aus, dass die Beziehung `prod (x ++ y) = (prod x) * (prod y)` bereits bewiesen ist.

3. Aufgabe: (AGS 11.41*)

(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: (AGS 11.37*)

```
1   data Tree = Leaf Int | Branch Int Tree Tree

2   sumTree :: Tree -> Int
3   sumTree (Leaf i) = i
4   sumTree (Branch i t1 t2) = i + (sumTree t1) + (sumTree t2)

5   revTree :: Tree -> Tree
6   revTree (Leaf i) = Leaf i
7   revTree (Branch i t1 t2) = Branch i (revTree t2) (revTree t1)

8   overlay :: Tree -> Tree -> Tree
9   overlay (Leaf i1) (Leaf i2) = Leaf (i1+i2)
10  overlay (Branch i1 t11 t12) (Branch i2 t21 t22) =
        Branch (i1+i2) (overlay t11 t21) (overlay t12 t22)
```

Zeigen Sie unter Verwendung der oben aufgeführten Definitionen mit Hilfe der strukturellen Induktion die Gültigkeit der Gleichung

`sumTree (overlay t t) = 2 * (sumTree (revTree t))` für jedes `t :: Tree`.

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