Programmierung

06. Übungsblatt

Zeitraum: 01. – 05. Juni 2015

Übung 1 (AGS 12.2.11)

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

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

Wenden Sie den Unifikationsalgorithmus auf die Terme t_1 und t_2 an. Wenden Sie bei jedem Umformungsschritt nur eine Regelsorte an und geben Sie diese jeweils an. Geben Sie anschließend den von Ihnen bestimmten allgemeinsten Unifikator an.

(b) Geben Sie zwei weitere Unifikatoren an.

Übung 2 (AGS 12.3.11)

Folgende Definitionen seien gegeben:

```
data Tree = Leaf Float | Branch Float Tree Tree

add :: Tree -> Float -> Tree
add (Leaf x)       a = Leaf (x + a)
add (Branch x l r) a = Branch (x + a/3) (add l (a/3)) (add r (a/3))

rev :: Tree -> Tree
rev (Leaf x)         = Leaf x
rev (Branch x l r)   = Branch x (rev r) (rev l)

sum :: Tree -> Float
sum (Leaf x)         = x
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 jeden Baum `t :: Tree` und jede Zahl `a :: Float` erfüllt ist:

```
sum (add t a) = (sum (rev t)) + a
```

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

Übung 3

Gegeben sei die folgende Funktionsdefinition.

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f z []     = z
foldr f z (x:xs) = f x (foldr f z xs)
```

Werten Sie schrittweise den Ausruck `foldr (+) 0 [3,7]` aus!

Übung 4 (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 12.2.26)

(a) Es seien δ ein dreistelliges, σ ein zweistelliges und γ ein einstelliges Funktionssymbol. $V = \{x_1, x_2, x_3\}$ sei eine Menge von Variablen. Wenden Sie den Unifikationsalgorithmus auf die Terme t_1 und t_2 an und ermitteln Sie deren allgemeinsten Unifikator:
$t_1 = \sigma\big(\gamma(x_2), \sigma(\gamma(x_3), x_3)\big)$
$t_2 = \sigma\big(x_1, \sigma(x_1, x_2)\big)$
Wenden Sie bei jedem Umformungsschritt nur eine Regelsorte an und geben Sie diese jeweils an.

(b) Geben Sie den von Ihnen bestimmten allgemeinsten Unifikator an.

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:

`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 oder Definitionen an.