TECHNISCHE UNIVERSITÄT DRESDEN
Fakultät Informatik • Institut für Theoretische Informatik • Lehrstuhl Grundlagen der Programmierung
Prof. Dr. H. Vogler / Dipl.-Inf. J. Osterholzer http://www.orchid.inf.tu-dresden.de

# Programmierung

## 05. Übungsblatt

Zeitraum: 08. – 12. Mai 2017

**Übung 1 (AGS 12.3.19)**

Folgende Definitionen seien gegeben:

```
data Tree a = Node a (Tree a) (Tree a) | Leaf a

mirror :: Tree a -> Tree a
mirror (Node x t1 t2) = Node x (mirror t2) (mirror t1)
mirror (Leaf x) = Leaf x

yield :: Tree a -> [a]
yield (Node _ t1 t2) = yield t1 ++ yield t2
yield (Leaf x) = [x]
```

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

(A): Für jeden Typ `a` und jeden Baum `t :: Tree 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.`

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

(a) Zeigen Sie den Induktionsanfang.

(b) Geben Sie die Induktionsvoraussetzung und zeigen Sie den Induktionsschritt.

**Übung 2 (AGS 12.2.13)**

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

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

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. Die erste Regelanwendung wurde bereits eingeleitet. Geben Sie anschließend den von Ihnen bestimmten allgemeinsten Unifikator an.

(b) Geben Sie zwei weitere Unifikatoren an.

**Übung 3 (AGS 12.4.1 ⋆)**

(a) Bestimmen Sie für jeden der folgenden $\lambda$-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 $\lambda$-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.
- $(\lambda x.(\lambda y.x\ z\ (y\ z)))\ (\lambda x.y\ (\lambda y.y))$
- $(\lambda x.(\lambda y.(\lambda z.z)))\ x\ (+\ y\ 1)$
- $(\lambda x.(\lambda y.x\ (\lambda z.y\ z)))\ (((\lambda x.(\lambda y.y))\ 8)\ (\lambda x.(\lambda y.y)\ x))$
- $(\lambda h.(\lambda x.h\ (x\ x))\ (\lambda x.h\ (x\ x)))\ ((\lambda x.x)\ (+\ 1\ 5))$
- $(\lambda f.(\lambda a.(\lambda b.f\ a\ b)))\ (\lambda x.(\lambda y.x))$

**Übung 4**

(a) Gegeben sei der $\lambda$-Term

$$\langle \text{pow} \rangle = \big(\lambda n f z.\ n\ (\lambda g x.\ g\ (g\ x))\ f\ z\big).$$

Berechnen Sie die Normalform von $\langle \text{pow} \rangle\ \langle 2 \rangle$. Sie müssen nicht alle Zwischenschritte angeben!

(b) Welche mathematische Funktion berechnet $\langle \text{pow} \rangle$?

(c) *Zusatzaufgabe:* Verallgemeinern Sie $\langle \text{pow} \rangle$.

**Zusatzaufgabe 1 (AGS 12.3.22 ⋆)**

Folgende Definitionnen seien gegeben:

```haskell
data Tree a = Branch a (Tree a) (Tree a) | Leaf a

-- | Knotenbeschriftungen des Baumes in preorder
p :: Tree a -> [a]
p (Leaf x)       = [x]
p (Branch x s t) = [x] ++ (p s ++ p t)

d :: Tree a -> Tree a -> Tree a
d (Leaf x) u       = Branch x u u
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 (12.2.14 ⋆)**

(a) Gegeben seien die Terme

$$t_1 = \delta(\alpha, \sigma(x_1, \alpha), \sigma(x_2, x_3)) \text{ und}$$
$$t_2 = \delta(\alpha, \sigma(x_1, x_2), \sigma(x_2, \gamma(x_2)))$$

über dem Rangalphabet $\Sigma = \{\delta^{(3)}, \sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$. 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) Gegeben seien die Haskell-Typterme

$t_1 =$ `(a, [a])`, $t_2 =$ `(Int, [Double])` und $t_3 =$ `(b, c)`.

Geben Sie für alle $1 \leq i < j \leq 3$ an, ob $t_i$ und $t_j$ unifizierbar sind, und ggf. ihren allgemeinsten Unifikator.