Programmierung

05. Übungsblatt

Zeitraum: 18. – 22. Mai 2015

Übung 1 (fakultativ, da nicht in VL behandelt; AGS 12.1.33)
Schreiben Sie die Typklasse `Zippable`, welche die Funktion `genericZip` und `genericZipWith` enthält. Die Funktionen sollen Verallgemeinerungen der Funktionen `zip` und `zipWith` für Listen auf weitere Datenstrukturen sein. Definieren Sie eine Instanz von `Zippable` für Listen und definieren Sie eine sinnvolle Instanz für die Datenstruktur `Tree`.

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

Fügen Sie der Typklasse eine sinnvolle Standardimplementierung hinzu.

Zusatzaufgabe: Definieren Sie auch eine Instanz für den Datentyp

```
data RoseTree a = RNode a [RoseTree a] deriving Show
```

Hilfestellung: Funktionen aus dem Modul `Prelude`.

```
zip :: [a] -> [b] -> [(a, b)]
zipWith :: (a -> b -> c) -> [a] -> [b] -> [c]
```

Übung 2
Gegeben sei die folgende Funktionsdefinition.

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

Werten Sie schrittweise den Aufruf `foldl (\x y -> 2*x+y) 0 [1,0,1]` aus! Welche Funktionalität hat er?

Übung 3 (AGS 12.2.4)
Es seien δ ein dreistelliges, σ ein zweistelliges, γ ein einstelliges und α ein nullstelliges 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:
$t_1 = \delta(\gamma(x_3), \gamma(\gamma(\alpha)), \sigma(\gamma(x_2), x_1))$
$t_2 = \delta(\gamma(x_3), \gamma(x_3), \sigma(\gamma(\alpha), \gamma(\gamma(x_2))))$
Wenden Sie bei jedem Umformungsschritt nur eine Regelsorte an und geben Sie sie jeweils an.

Übung 4 (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:

```
sum (foo xs) = 2 * sum xs - length xs .
```

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

(a) Zeigen Sie den Induktionsanfang.
(b) Geben Sie die Induktionsvoraussetzung *vollständig* an.
(c) Zeigen Sie den Induktionsschritt.

Zusatzaufgabe 1 (AGS 12.2.1 ⋆)
Es sei δ ein dreistelliges, γ ein einstelliges und α ein nullstelliges Basisfunktionssymbol. Des Weiteren sei $V = \{x_1, \ldots, x_5\}$ eine Menge von Variablen.

Wenden Sie den Unifikationsalgorithmus jeweils auf die Terme t_1 und t_2 an und ermitteln Sie jeweils den allgemeinsten Unifikator:

- $t_1 = \delta(\gamma(x_1), \delta(\gamma(\alpha), \gamma(x_2), \gamma(\gamma(\alpha))), x_3)$
  $t_2 = \delta(\gamma(\gamma(x_5)), \delta(x_4, \gamma(x_2), \gamma(x_1)), \alpha)$
- $t_1 = \delta(\gamma(\alpha), x_3, \gamma(\gamma(x_3)))$
  $t_2 = \delta(x_1, \delta(\alpha, x_2, \alpha), x_2)$

Zusatzaufgabe 2 (AGS 12.3.5 ⋆)
Zeigen Sie mit Hilfe des Prinzips der Induktion über Listen die Gültigkeit der folgenden (intuitiv einsichtigen) Gleichung:

```
map g (y ++ z) = map g y ++ map g z
```

Nutzen Sie dazu die folgende Definition von `map`:

```
1  map :: (t -> u) -> [t] -> [u]
2  map f [] = []
3  map f (x:xs) = f x : map f xs
```