4. Übung (29. April–03. Mai 2019)

Programmierung

Beachten Sie die Übungsverlegungen zum 1. Mai.

Aufgabe 1 (AGS 12.1.16a, 12.1.48b,c, 12.1.56)
Gegeben sei der polymorphe algebraische Datentyp

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

(a) Schreiben Sie einen Baum des Typs `Tree Int` mit mindestens 5 Blättern auf.

(b) Geben Sie eine Funktion `depth :: Tree a -> Int` an, welche für einen Baum `t` die Länge des kürzesten Pfads von der Wurzel zu einem Blattknoten von `t` berechnet. Die Länge eines Pfads ist dabei die Anzahl der auf ihm vorkommenden Knoten, d.h. der Pfad von der Wurzel zur Wurzel selbst hat die Länge `1`.

(c) Geben Sie eine Funktion `paths :: Tree a -> Tree [a]` an, die in einem Baum die Beschriftung jedes Knotens u durch die Liste der Knotenbeschriftungen auf dem Pfad vom Wurzelknoten zu u ersetzt. Ist der Wurzelknoten z. B. in `t` mit `5` beschriftet, so ist er in `paths t` mit `[5]` beschriftet, und ist sein erster Kindknoten in `t` mit `3` beschriftet, dann ist dieser in `paths t` mit `[5,3]` beschriftet.

(d) Schreiben Sie eine Funktion `tmap :: (a -> b) -> Tree a -> Tree b`, so dass für alle `f :: a -> b` und `t :: Tree a` gilt, dass `tmap f t` der Baum ist, der entsteht, indem jeder Knoten `v` von `t` durch `f v` ersetzt wird.

Aufgabe 2 (ABS 12.1.51a,b)

(a) Schreiben Sie die Funktion `unzip :: [(a, b)] -> ([a], [b])`, welche eine Liste von Paaren in zwei Listen aufspaltet. Dabei soll die erste Liste die ersten Komponenten und die zweite Liste die zweiten Komponenten der Paare enthalten, z.B.:
`unzip [('a', 1), ('b', 2), ('c', 3)] = (['a', 'b', 'c'], [1, 2, 3])`. Die Reihenfolge der Elemente soll erhalten bleiben.

(b) Gegeben seien folgende Funktionen.

```
1 map :: (a -> b) -> [a] -> [b]
2 map _ []       = []
3 map f (x : xs) = f x : map f xs
4
5 uncurry :: (a -> b -> c) -> (a, b) -> c
6 uncurry f (x, y) = f x y
```

Werten Sie den Term `map (uncurry (+)) [(1, 2), (3, 4)]` *schrittweise* aus. Sie dürfen dabei sinnvolle Abkürzungen einführen.

Aufgabe 3 (AGS 12.2.12)
Gegeben seien folgende Terme über dem Rangalphabet $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$:

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

(a) Wenden Sie den Unifikationsalgorithmus auf die Terme t_1 und t_2 an. Geben Sie anschließend den von Ihnen bestimmten allgemeinsten Unifikator an.

(b) Geben Sie zwei weitere Unifikatoren an.

(c) Geben Sie zwei Terme t_1 und t_2 (über einem beliebigen Alphabet) an, so dass im Laufe der Anwendung des Unifikationsalgorithmus auf t_1 und t_2 der Occur-Check fehlschlägt.

Zusatzaufgabe 1 (ABS 12.1.21a–c)
Gegeben sei der folgende polymorphe algebraische Datentyp:

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

mit dessen Hilfe sich binäre Bäume konstruieren lassen, die an den Blättern Werte des Typs `a` speichern.

(a) Programmieren Sie eine Funktion `check :: Tree Bool -> Bool`, die genau dann `True` liefert, wenn der Eingabebaum mindestens ein Blatt mit dem gespeicherten Wert `False` besitzt.

(b) Programmieren Sie eine Funktion `toList :: Tree a -> [a]`, die aus einem Baum des Typs `Tree a` eine Liste der gespeicherten Werte der Blätter generiert, und zwar in der Reihenfolge von rechts nach links gesehen.

(c) Programmieren Sie eine Funktion `toTree :: [Int] -> Tree Int`, die eine Liste von Zahlen als Argument nimmt und einen Baum erzeugt, welcher - zusätzlich zu einem Blatt mit dem gespeicherten Wert 42 - für jedes Element der Liste genau ein Blatt mit dessen Wert besitzt. Die Form des Baumes spielt keine Rolle.

Zusatzaufgabe 2 (AGS 12.2.15)
Gegeben seien die Terme

$$t_1 = \sigma(\alpha,\ \sigma(\gamma(\alpha), \sigma(x_2, x_3 \qquad))) \,,$$
$$t_2 = \sigma(\alpha,\ \sigma(x_1 \quad, \sigma(x_2, \sigma(x_2, x_1))))$$

über dem Rangalphabet $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$. 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.