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

SS 11

Aufgabenblatt zur 4. Übung

Zeitraum: 02.05. bis 06.05.2011

1. Aufgabe: (AGS 11.31)

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 diese jeweils an.

2. Aufgabe: (AGS 11.34*)

Gegeben seinen folgende Definitionen von Funktionen:

```
sumlist1 :: [Int] -> Int
sumlist1 [] = 0
sumlist1 (b:x) = b + (sumlist1 x)

sumlist2 :: [(Int,Int)] -> Int
sumlist2 [] = 0
sumlist2 ((a,b):y) = (a + b) + (sumlist2 y)

zip :: [t] -> [u] -> [(t,u)]
zip [] [] = []
zip (a:x) [] = []
zip [] (b:y) = []
zip (a:x) (b:y) = ((a,b):(zip x y))
```

Zeigen Sie mit Hilfe des Prinzips der Induktion über Listen, dass folgende Gleichung unter der Bedingung $length(\texttt{x1}) = length(\texttt{x2})$ gilt:

`sumlist2 (zip x1 x2) = (sumlist1 x1) + (sumlist1 x2)`

3. Aufgabe: (AGS 11.44)

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 beliebigen Baum `t :: Tree` und eine beliebige Zahl `a :: Float` erfüllt ist:

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

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

Zusatzaufgabe: (AGS 11.41*)

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

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

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

overlay :: Tree -> Tree -> Tree
overlay (Leaf i1) (Leaf i2) = Leaf (i1+i2)
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.