



tut04

PROGRAMMIERUNG

ÜBUNG 4: TYPOLYMORPHIE & UNIFIKATION

Eric Kunze

eric.kunze@mailbox.tu-dresden.de

INHALT

1. Funktionale Programmierung
 - 1.1 Einführung in Haskell: Listen
 - 1.2 Algebraische Datentypen
 - 1.3 Funktionen höherer Ordnung
 - 1.4 **Typpolymorphie & Unifikation** ↗
 - 1.5 Beweis von Programmeigenschaften
 - 1.6 λ – Kalkül
2. Logikprogrammierung
3. Implementierung einer imperativen Programmiersprache
4. Verifikation von Programmeigenschaften
5. H_0 – ein einfacher Kern von Haskell

Typpolymorphie

TYPPOLYMORPHIE

- ▶ **bisher:** Funktionen mit konkreten Datentypen
z.B. $\text{length} :: [\underline{\text{Int}}] \rightarrow \text{Int}$
- ▶ **Problem:** Funktion würde auch auf anderen
Datentypen funktionieren
z.B. $\text{length} :: [\underline{\text{Bool}}] \rightarrow \text{Int}$
- ▶ **Lösung:** Typvariablen und polymorphe Funktionen
 $\text{length} :: [\underline{a}] \rightarrow \text{Int}$

Bei konkreter Instanziierung wird Typvariable an
entsprechenden Typbezeichner gebunden (z.B. $a = \text{Int}$
oder $a = \text{Bool}$).
$$\begin{array}{c} [\underline{a}] \rightarrow \underline{a = \text{Int}} \\ \text{length } [1,2,3] \end{array}$$

2

Übungsblatt 4

Aufgabe 1

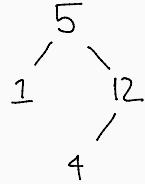
AUFGABE 1 — TEIL (A)

Beispielbaum mit mindestens 5 Blättern

```
data BinTree a = Branch a (BinTree a) (BinTree a) | Leaf a  
               Branch Int Bin Bin
```

```
testTree :: BinTree Int
```

```
testTree = Branch 5  
          (Leaf 1)  
          (Branch 12  
            Branch 4)
```



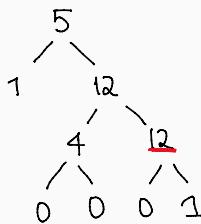
3

AUFGABE 1 — TEIL (A)

Beispielbaum mit mindestens 5 Blättern

```
data BinTree a = Branch a (BinTree a) (BinTree a) | Leaf a deriving Show
```

```
1 testTree :: BinTree Int  
2 testTree = Branch 5  
3           (Leaf 1)  
4           (Branch 12  
5             (Branch 4  
6               (Leaf 0)  
7               (Leaf 0))  
8             (Branch 12  
9               (Leaf 0)  
10              (Leaf 1)))
```



3

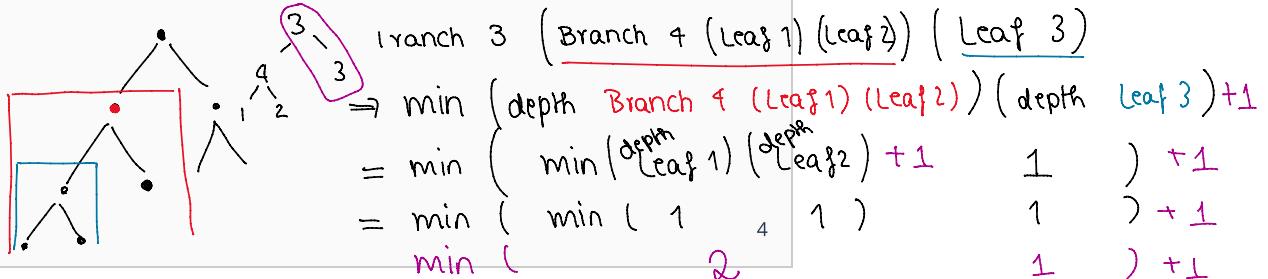
AUFGABE 1 — TEIL (B)

minimale Tiefe eines Baumes

```
data BinTree a = Branch a (BinTree a) (BinTree a) | Leaf a
depth :: BinTree a -> Int
```

$$\text{depth} (\text{Leaf } _) = 1$$

$$\text{depth} (\text{Branch } _ l r) = \min (\text{depth } l) (\text{depth } r) + 1$$



$$= 2$$

AUFGABE 1 — TEIL (B)

minimale Tiefe eines Baumes

```
data BinTree a = Branch a (BinTree a) (BinTree a) | Leaf a
depth :: BinTree a -> Int
```

```

1 -- Aufgabe 1 (b)
2 depth :: BinTree a -> Int
3 depth (Leaf \_) = 1
4 depth (Branch \_ l r) = min (depth l) (depth r) + 1

```

4

AUFGABE 1 — TEIL (C)

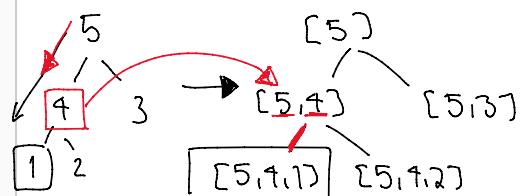
Baum mit Beschriftungsfolge neu beschriften

```
data BinTree a = Branch a (BinTree a) (BinTree a) | Leaf a
paths :: BinTree a -> BinTree [a]
```

```

paths t = go [] t
go :: [a] → BinTree a → BinTree [a]
go save (Leaf x) = Leaf (save ++ [x])
go save (Branch x l r)
= Branch (save ++ [x])
  (go (save ++ [x]) l)
  (go (save ++ [x]) r)

```



5

AUFGABE 1 — TEIL (C)

Baum mit Beschriftungsfolge neu beschriften

```
data BinTree a = Branch a (BinTree a) (BinTree a) | Leaf a
paths :: BinTree a -> BinTree [a]
```

```

1 -- Aufgabe 1 (c)
2 paths :: BinTree a -> BinTree [a]
3 paths = go []
4 where
5   go :: [a] -> BinTree a -> BinTree [a]
6   go prefix (Leaf x) = Leaf (prefix ++ [x])
7   go prefix (Branch x l r)
8     = Branch (prefix ++ [x])
9     (go (prefix ++ [x]) l)
10    (go (prefix ++ [x]) r)

```

↑ $g(t) = g(t)$
← $\delta = g$

5

AUFGABE 1 — TEIL (D)

Map für Bäume

Typvariable

data BinTree a = Branch a (BinTree a) (BinTree a) | Leaf a

tmap :: $(a \rightarrow b) \rightarrow (\text{BinTree } a \rightarrow \text{BinTree } b)$

$a = \text{String}$ $b = \text{Int}$

$\text{String} \rightarrow \text{Int}$ String Int

$\text{tmap } f (\text{Leaf } x) = \text{Leaf } (f x)$

$\text{tmap } f (\text{Branch } x l r)$

$= \text{Branch } (f x) (\text{tmap } f l) (\text{tmap } f r)$

vgl. Listen-map:

$\text{map } f(x: xs) = f x : \text{map } f xs$

6

AUFGABE 1 — TEIL (D)

Map für Bäume

```
data BinTree a = Branch a (BinTree a) (BinTree a) | Leaf a  
tmap :: (a -> b) -> BinTree a -> BinTree b
```

```
1 -- Aufgabe 1 (d)  
2 tmap :: (a -> b) -> BinTree a -> BinTree b  
3 tmap f (Leaf x) = Leaf (f x)  
4 tmap f (Branch x l r) = Branch (f x) (tmap f l) (tmap f r)
```

6

Übungsblatt 4

Aufgabe 2

AUFGABE 2 — TEIL (A)

Liste von Paaren → Paare von Listen

→ unzip

unpairs :: [(a,b)] -> ([a], [b])

[(1,2), (3,4), (5,6)]

↳ ([1,3,5], [2,4,6])

→ unpairs [] = ([], []) ↳ {b}

• unpairs ((a,b): xs) = let (as,bs) = unpairs xs

$\frac{(x): xs}{(a,b): xs} \quad \text{in } \left(\underbrace{a: as}_{[a]}, \underbrace{b: bs}_{[b]} \right)$

• unpairs ((a,b): xs) = (a: as, b: bs) = (a1: a2: a3: ... : [], b1: b2: ... : [])

where (as,bs) = unpairs xs

([], [])

7

AUFGABE 2 — TEIL (A)

Liste von Paaren → Paare von Listen

unpairs :: $[(a, b)] \rightarrow ([a], [b])$

```
1 -- Aufgabe 2 (a)
2 unpairs :: [(a, b)] -> ([a], [b])
3 unpairs []           = []
4 unpairs ((a, b):xs) = let (as, bs) = unpairs xs
                           in (a:as, b:bs)
```

7

AUFGABE 2 — TEIL (B)

```
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
```

```
map (uncurry (+)) [(1,2), (3,4)]
= map (uncurry (+)) (1,2) : [(3,4)]
= (uncurry (+) (1,2)) : map (uncurry (+)) [(3,4)]
= 1 + 2 : map (uncurry (+)) (3,4) : []
= 3 : (uncurry (+) (3,4)) : map (uncurry (+)) []
= 3 : (3 + 4) : []
= 3 : 7 : []
= [3, 7]
```

8

AUFGABE 2 — TEIL (B)

```
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
```

```
map (uncurry (+)) [(1,2), (3,4)]
= map (uncurry (+)) ((1,2) : [(3,4)])
= uncurry (+) (1,2) : map (uncurry (+)) [(3,4)]
= (1 + 2) : map (uncurry (+)) [(3,4)]
= 3 : map (uncurry (+)) [(3,4)]
= 3 : map (uncurry (+)) ((3,4); [])
= 3 : uncurry (+) (3,4) : map (uncurry (+)) []
= 3 : (3 + 4) : map (uncurry (+)) []
= 3 : 7 : map (uncurry (+)) []
= 3 : 7 : []
= [3, 7]
```

8

AUFGABE 2 — TEIL (B)

```
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
```

```
map (uncurry (+)) [(1,2), (3,4)]
= map (uncurry (+)) ((1,2):[(3,4)])
= uncurry (+) (1,2) : map (uncurry (+)) [(3,4)]
= (1 + 2) : map (uncurry (+)) [(3,4)]
= 3 : map (uncurry (+)) [(3,4)]
= 3 : map (uncurry (+)) ((3,4);[])
= 3 : uncurry (+) (3,4) : map (uncurry (+)) []
= 3 : (3 + 4) : map (uncurry (+)) []
= 3 : 7 : map (uncurry (+)) []
= 3 : 7 : []
= [3, 7]
```

8

Unifikation & Unifikationsalgorithmus

UNIFIKATION

Motivation: Typüberprüfung

```
1 f :: (t, Char) -> (t, [Char])
2 f (...) = ...
3
4 g :: (Int, [u]) -> Int
5 g (...) = ...
6
7 h = (g . f)
```

(t, [Char])
(Int, [u])
t ↪ Int u ↪ Char

~~()² ((t), [] (Char))~~
~~()² ((Int), [] (u))~~

9

UNIFIKATION

Typausdrücke

- ▶ Int, Bool, Float, Char, String ←
- ▶ Typvariablen ← t, u
- ▶ Listen, Tupel, Funktionen ← {t}, (Char, u), t → Int

Typterm

- ▶ Übersetzung trans: Typausdruck → Typterm
- ▶ z.B.
$$\text{trans}((t, [Char])) = ()^2(t, [](\text{Char}))$$
$$\text{trans}((\text{Int}, [u])) = ()^2(\text{Int}, [](u))$$

$$\begin{aligned}\text{trans } [t] &= [](t) \\ \text{trans } (\text{Char}, u) &= ()^2(\text{Char}, u) \\ \text{trans } (t \rightarrow \text{Int}) &= \rightarrow (t, \text{Int})\end{aligned}$$

10

UNIFIKATION

Typausdrücke

- ▶ Int, Bool, Float, Char, String
- ▶ Typvariablen
- ▶ Listen, Tupel, Funktionen

Typterm

- ▶ Übersetzung trans: Typausdruck → Typterm
- ▶ z.B.
$$\text{trans}((t, [Char])) = ()^2(t, [](\text{Char}))$$
$$\text{trans}((\text{Int}, [u])) = ()^2(\text{Int}, [](u))$$

Beide Typausdrücke können in Übereinstimmung gebracht werden, wenn die Typterm $\text{trans}((t, [Char]))$ und $\text{trans}((\text{Int}, [u]))$ unifizierbar sind

10

UNIFIKATION

Tytausdrücke

- Int, Bool, Floar, Char, String
- Typvariablen
- Listen, Tupel, Funktionen

Typterm

- Übersetzung $trans$: Tytausdruck \rightarrow Typterm
- z.B.

$$trans((t, [Char])) = ()^2(t, [](Char))$$

$$trans((Int, [u])) = ()^2(Int, [](u))$$

Beide Tytausdrücke können in Übereinstimmung gebracht werden, wenn die Typterm $trans((t, [Char]))$ und $trans((Int, [u]))$ unifizierbar sind $\rightarrow t = \text{Int}$ und $u = \text{Char}$

10

UNIFIKATIONSALGORITHMUS

- **gegeben:** zwei Typterm t_1, t_2
- **Ziel:** entscheide, ob t_1 und t_2 unifizierbar sind

Wir notieren die beiden Typterm als Spalten:

$$\begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \quad \text{bzw.} \quad \begin{pmatrix} ()^2(t, [](Char)) \\ ()^2(Int, [](u)) \end{pmatrix}$$

Unifikationsalgorithmus erstellt eine Folge von Mengen M_i , wobei die M_{i+1} aus M_i hervorgeht, indem eine der vier Regeln angewendet wird.

$$M_1 := \left\{ \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \right\} \quad \text{bzw.} \quad M_1 := \left\{ \begin{pmatrix} ()^2(t, [](Char)) \\ ()^2(Int, [](u)) \end{pmatrix} \right\}$$

$$M_2 = \{(), ., ()\}$$

11

UNIFIKATIONSALGORITHMUS – REGELN

- **Dekomposition.** Sei $\delta \in \Sigma$ ein k -stelliger Konstruktor, $s_1, \dots, s_k, t_1, \dots, t_k$ Terme über Konstruktoren und Variablen.

$$\begin{pmatrix} \delta(\dots) \\ \delta(\dots) \end{pmatrix} \rightsquigarrow \begin{pmatrix} s_1, \dots, s_k \\ t_1, \dots, t_k \end{pmatrix}$$

- **Elimination.** Sei x eine Variable!

$$\begin{pmatrix} x \\ x \end{pmatrix} \rightsquigarrow \emptyset$$

- **Vertauschung.** Sei t keine Variable.

$$\begin{pmatrix} t \\ x \end{pmatrix} \rightsquigarrow \begin{pmatrix} x \\ t \end{pmatrix} \quad \begin{matrix} u \\ ((\text{Char}, t)) \end{matrix} \quad \begin{matrix} \leftarrow \text{Var.} \\ \leftarrow \text{Term} \end{matrix}$$

- **Substitution.** Sei x eine Variable, t keine Variable und x kommt nicht in t vor (occur check). Dann ersetze in jedem anderen Term die Variable x durch t .

$$u \mapsto (\text{Char}, t)$$

12

UNIFIKATIONSALGORITHMUS

Ende: keine Regel mehr anwendbar – Entscheidung:

- t_1, t_2 **unifizierbar:** M ist von der Form

$$\left\{ \left(\begin{array}{c} u_1 \\ t_1 \end{array} \right), \left(\begin{array}{c} u_2 \\ t_2 \end{array} \right), \dots, \left(\begin{array}{c} u_k \\ t_k \end{array} \right) \right\} \xleftarrow{\text{verso „Variablen“}} \text{„Terme ohne Variablen“}$$

wobei u_1, u_2, \dots, u_k paarweise verschiedene Variablen sind und nicht in t_1, t_2, \dots, t_k vorkommen.

allgemeinster Unifikator φ :

$$\varphi(u_i) = t_i \quad (i = 1, \dots, k)$$

$$\varphi(x) = x \quad \text{für alle nicht vorkommenden Variablen}$$

- t_1, t_2 **sind nicht unifizierbar:** M hat nicht diese Form und keine Regel ist anwendbar

13

Übungsblatt 4

Aufgabe 3

AUFGABE 3

$$\begin{aligned} & \left\{ \begin{pmatrix} \delta(\alpha, \sigma(x_1, \alpha), \sigma(x_2, x_3)) \\ \delta(\alpha, \sigma(x_1, x_2), \sigma(x_2, \gamma(x_2))) \end{pmatrix} \right\} \\ \xrightarrow{\text{Dek.}} & \left\{ \begin{pmatrix} \alpha \\ \alpha \end{pmatrix}, \begin{pmatrix} \sigma(x_1, \alpha) \\ \sigma(x_1, x_2) \end{pmatrix}, \begin{pmatrix} \sigma(x_2, x_3) \\ \sigma(x_2, \gamma(x_2)) \end{pmatrix} \right\} \\ \xrightarrow{\text{Dek.}_3} & \left\{ \begin{pmatrix} x_1 \\ x_1 \end{pmatrix}, \begin{pmatrix} \alpha \\ x_2 \end{pmatrix}, \begin{pmatrix} x_2 \\ x_2 \end{pmatrix}, \begin{pmatrix} x_3 \\ \gamma(x_2) \end{pmatrix} \right\} \\ \xrightarrow{\text{El.}_2} & \left\{ \begin{pmatrix} \alpha \\ x_2 \end{pmatrix}, \begin{pmatrix} x_3 \\ \gamma(x_2) \end{pmatrix} \right\} \\ \xrightarrow{\text{Vert.}} & \left\{ \begin{pmatrix} x_2 \\ \alpha \end{pmatrix}, \begin{pmatrix} x_3 \\ \gamma(x_2) \end{pmatrix} \right\} \\ \xrightarrow{\text{Subst.}} & \left\{ \begin{pmatrix} x_2 \\ \alpha \end{pmatrix}, \begin{pmatrix} x_3 \\ \gamma(\alpha) \end{pmatrix} \right\} \end{aligned}$$

14

AUFGABE 3

allgemeinster Unifikator:

$$x_1 \mapsto x_1 \quad x_2 \mapsto \alpha \quad x_3 \mapsto \gamma(\alpha)$$

15

AUFGABE 3

allgemeinster Unifikator:

$$x_1 \mapsto x_1 \quad x_2 \mapsto \alpha \quad x_3 \mapsto \gamma(\alpha)$$

Teilaufgabe (b)

$$\begin{aligned} t_1 &= (\text{a} , [\text{a}]) \\ t_2 &= (\text{Int} , [\text{Double}]) \\ t_3 &= (\text{b} , \text{c}) \end{aligned}$$

15

AUFGABE 3

allgemeinster Unifikator:

$$x_1 \mapsto x_1 \quad x_2 \mapsto \alpha \quad x_3 \mapsto \gamma(\alpha)$$

Teilaufgabe (b)

$$t_1 = (\text{a} , [a])$$

$$t_2 = (\text{Int} , [\text{Double}])$$

$$t_3 = (\text{b} , c)$$

- t_1 und t_2 sind *nicht* unifizierbar
- t_1 und t_3 sind unifizierbar mit
 $a \mapsto a \quad b \mapsto a \quad c \mapsto [a]$
- t_2 und t_3 sind unifizierbar mit $b \mapsto \text{Int} \quad c \mapsto [\text{Double}]$

15

Übungsblatt 4

Aufgabe 4

AUFGABE 4

$$\begin{aligned} & \left\{ \left(\sigma(\sigma(x_1, \alpha), \sigma(\gamma(x_3), x_3)) \right) \right\} \\ \xrightarrow{\text{Dek.}} & \left\{ \left(\sigma(x_1, \alpha) \right), \left(\sigma(\gamma(x_3), x_3) \right) \right\} \\ \xrightarrow{\text{Dek.}^2} & \left\{ \left(\begin{array}{c} x_1 \\ \gamma(x_2) \end{array} \right), \left(\begin{array}{c} \alpha \\ \alpha \end{array} \right), \left(\begin{array}{c} \gamma(x_3) \\ x_2 \end{array} \right), \left(\begin{array}{c} x_3 \\ x_3 \end{array} \right) \right\} \\ \xrightarrow{\text{El.}} & \left\{ \left(\begin{array}{c} x_1 \\ \gamma(x_2) \end{array} \right), \left(\begin{array}{c} \alpha \\ \alpha \end{array} \right), \left(\begin{array}{c} \gamma(x_3) \\ x_2 \end{array} \right) \right\} \\ \xrightarrow{\text{Dek.}} & \left\{ \left(\begin{array}{c} x_1 \\ \gamma(x_2) \end{array} \right), \left(\begin{array}{c} \gamma(x_3) \\ x_2 \end{array} \right) \right\} \\ \xrightarrow{\text{Vert.}} & \left\{ \left(\begin{array}{c} x_1 \\ \gamma(x_2) \end{array} \right), \left(\begin{array}{c} x_2 \\ \gamma(x_3) \end{array} \right) \right\} \quad x_2 \text{ kommt nicht in } \gamma(x_3) \text{ vor} \\ \xrightarrow{\text{Subst.}} & \left\{ \left(\begin{array}{c} x_1 \\ \gamma(\gamma(x_3)) \end{array} \right), \left(\begin{array}{c} x_2 \\ \gamma(x_3) \end{array} \right) \right\} \end{aligned}$$

16

AUFGABE 4

allgemeinster Unifikator:

$$x_1 \mapsto \gamma(\gamma(x_3)) \quad x_2 \mapsto \gamma(x_3) \quad x_3 \mapsto x_3$$

17

AUFGABE 4

allgemeinster Unifikator:

$$x_1 \mapsto \gamma(\gamma(x_3)) \quad x_2 \mapsto \gamma(x_3) \quad x_3 \mapsto x_3$$

weitere Unifikatoren:

$$\begin{array}{lll} x_1 \mapsto \gamma(\gamma(\alpha)) & x_2 \mapsto \gamma(\alpha) & x_3 \mapsto \alpha \\ x_1 \mapsto \gamma(\gamma(\gamma(\alpha))) & x_2 \mapsto \gamma(\gamma(\alpha)) & x_3 \mapsto \gamma(\alpha) \end{array}$$

17

AUFGABE 4

allgemeinster Unifikator:

$$x_1 \mapsto \gamma(\gamma(x_3)) \quad x_2 \mapsto \gamma(x_3) \quad x_3 \mapsto x_3$$

weitere Unifikatoren:

$$\begin{array}{lll} x_1 \mapsto \gamma(\gamma(\alpha)) & x_2 \mapsto \gamma(\alpha) & x_3 \mapsto \alpha \\ x_1 \mapsto \gamma(\gamma(\gamma(\alpha))) & x_2 \mapsto \gamma(\gamma(\alpha)) & x_3 \mapsto \gamma(\alpha) \end{array}$$

Fehlschlag beim occur-check:

Alphabet: $\Sigma = \{\gamma^{(1)}\}$

$$\begin{aligned} t_1 &= x_1 \\ t_2 &= \gamma(x_1) \end{aligned}$$

17

ENDE

Fragen?

18