

# Algorithmen und Programmierung

## 11. Aufgabenblatt

### Aufgabe 11.1

```
module Natuerlich where

data Nat = Null | Nf Nat

-- Konvertiert eine Zahl in einen String,
-- der die Form eines Nat hat
convert :: Int -> String
convert 0 = "Null"
convert n = "Nf (" ++ (convert (n-1)) ++ ")"

-- Gibt den Nat als Zahl aus
instance Show Nat where
    show a = show (count a)
    where
        count Null    = 0
        count (Nf a) = 1 + count a

-- == und /= auf Nat
instance Eq Nat where
    (==) Null Null      = True
    (==) Null (Nf a)   = False
    (==) (Nf a) Null   = False
    (==) (Nf a) (Nf b) = a == b

-- <=, <, >=, >, ... auf Nat
instance Ord Nat where
    (<=) Null Null      = True
    (<=) Null (Nf a)   = True
    (<=) (Nf a) Null   = False
    (<=) (Nf a) (Nf b) = a <= b

-- + und * auf Nat
instance Num Nat where
    (+) a Null    = a
    (+) a (Nf b) = Nf a + b
    (*) a Null    = Null
    (*) Null b    = Null
    (*) a (Nf b) = a + (a * b)
    signum       = error ""
    abs          = error ""
    fromInteger  = error ""

-- Nichtlinear rekursiver Algorithmus
fibolin :: Nat -> Nat
```

```

fibolin Null = Null
fibolin (Nf Null) = Nf Null
fibolin (Nf (Nf a)) = fibolin (Nf a) + fibolin a

-- Endrekursiver Algorithmus
fiboen :: Nat -> Nat
fiboen a = fibo' a (Nf Null) Null
  where
    fibo' Null n_m1 n_m2      = n_m2
    fibo' (Nf Null) n_m1 n_m2 = n_m1
    fibo' (Nf a) n_m1 n_m2
      = fibo' a (n_m1+n_m2) n_m1

```

## Aufgabe 11.2

- a) Wie viele terminierende Reduktionen von  $\text{quadrat}(4+5)$  gibt es?  
(mit  $\text{quadrat } x = x*x$ )

$\text{quadrat } (4+5)$ $\rightarrow (4+5) * (4+5)$ $\rightarrow 9 * (4+5)$ $\rightarrow 9 * 9$ $\rightarrow 81$	$\text{quadrat } (4+5)$ $\rightarrow (4+5) * (4+5)$ $\rightarrow (4+5) * 9$ $\rightarrow 9 * 9$ $\rightarrow 81$	$\text{quadrat } (4+5)$ $\rightarrow \text{quadrat } 9$ $\rightarrow 9 * 9$ $\rightarrow 81$
--	--	---

- b) Gesucht sind die Folgen von Reduktionen bei applikativer bzw. normaler Reduktion für folgenden Ausdruck:

$\text{map } f [1..2]$  where  $f n = 10*10+n$

Applikativ:

```

map f [1..2]
→ map f [1,2]
→ f 1 : map f [2]
→ 10*10+1 : map f [2]
→ 101 : map f [2]
→ 101 : 10*10+2 : map f []
→ 101 : 102 : map f []
→ 101 : 102 : []
→ [101,102]

```

Normal:

```

map f [1..2]
→ f 1 : map f [2]
→ 10*10+1 : map f [2]
→ 101 : (f 2 : map f [])
→ 101 : (10*10+2 : map f [])

```

→ 101 : (102 : [])  
→ [101,102]

c)

$$\begin{aligned}f(0) &= 0 \\f(1) &= 1 \\f(n+2) &= 2f(n+1) - f(n)\end{aligned}$$

$$\begin{aligned}f(3) &= 2f(2) - f(1) \\&= 2(2f(1) - f(0)) - f(1) \\&= 2(2 \cdot 1 - 0) - f(1) \\&= 2 \cdot 2 - f(1) \\&= 4 - 1 \\&= \underline{3}\end{aligned}$$

d) Beweis durch vollständige Induktion

**Induktionsbehauptung:**

$$f(n) = 2f(n-1) - f(n-2) = n \quad \forall n \in \mathbb{N}$$

**Induktionsanfang:**

$$f(0) = 0 \Leftrightarrow 0 = 0$$

$$f(1) = 1 \Leftrightarrow 1 = 1$$

Hier muss ein zweiteiliger Induktionsanfang gemacht werden, da die Funktion sich auf die 2 jeweils vorher berechneten Argumente  $f(n-1)$  und  $f(n-2)$  abstützt.

**Induktionsvoraussetzung.:**

Behauptung gilt für alle  $f(n)$  mit  $n \geq 0$

**Induktionsschluss:**

zu zeigen ist  $f(n+1) = n+1$

$$\begin{aligned} f(n+1) &= 2 \underbrace{f(n)}_{\text{nach IV: } f(n) = n} - \underbrace{f(n-1)}_{\text{nach IV: } f(n-1) = n-1} \\ &= 2n - (n-1) \\ &= 2n - n + 1 \\ &= n + 1 \end{aligned}$$

Nach Definition

Nach Induktionsvoraussetzung

q.e.d.

Dies ist eine wahre Aussage. Unsere Behauptung gilt.

### Aufgabe 11.3

#### a) Unifizierung von $(a,[b])$ mit $(a,c)$ zu $(\text{Bool}, [\text{Bool}])$

$$\begin{aligned} (a,[b]) &= \underbrace{[\text{Bool}/a]}_{\text{ersetze a durch Bool}} (a,[b]) \\ &= [\text{Bool}/b] (a,[\text{Bool}]) \end{aligned}$$

$$\begin{aligned} (a,c) &= [\text{Bool}/a] (a,c) \\ &= [[\text{Bool}]/c] (a,[\text{Bool}]) \end{aligned}$$

#### b) Signaturen

```
curry :: ((a,b) -> c) -> a -> b -> c
uncurry :: (a -> b -> c) -> (a,b) -> c
id :: t -> t
```

**curry id**

$$\text{curry} :: \underbrace{\underbrace{(a,b) \rightarrow c}_t}_{t \rightarrow t} \rightarrow a \rightarrow b \rightarrow c \qquad \text{id} :: \underbrace{t \rightarrow t}_{((a,b) \rightarrow c)}$$

Es gilt somit :

- i)  $(a,b) \rightarrow c \equiv t \rightarrow t$
- ii)  $t \equiv (a,b)$
- iii)  $t \equiv c \Rightarrow c \equiv t \equiv (a,b)$

Signatur von curry id :

$$\text{curry id} :: a \rightarrow b \rightarrow \underbrace{c}_{\substack{t \\ (a,b)}}$$

$$\Rightarrow \text{curry id} :: a \rightarrow b \rightarrow (a,b)$$

### uncurry id

$$\text{uncurry id} :: \underbrace{\underbrace{a \rightarrow (b \rightarrow c)}_t}_{t \rightarrow t} \rightarrow (a,b) \rightarrow c \qquad \text{id} :: \underbrace{t \rightarrow t}_{(a \rightarrow b \rightarrow c)}$$

Es gilt somit :

- i)  $\underbrace{a \rightarrow b \rightarrow c \equiv a \rightarrow (b \rightarrow c)}_{\text{da } \rightarrow \text{ rechtsassoziativ}} \equiv t \rightarrow t$
- ii)  $t \equiv (b \rightarrow c)$
- iii)  $t \equiv a \Rightarrow a \equiv t \equiv (b \rightarrow c)$

Signatur von uncurry id :

$$\text{uncurry id} :: (a, \underbrace{b \rightarrow c}_t)$$

$$\Rightarrow \text{uncurry id} :: (b \rightarrow c, b) \rightarrow c$$

### curry (curry id)

$$\text{curry} :: \underbrace{\underbrace{(a,b) \rightarrow c}_{d} \rightarrow e \rightarrow (d \rightarrow e)}_{d \rightarrow (e \rightarrow (d \rightarrow e))} \rightarrow a \rightarrow b \rightarrow c \quad \text{curry id} :: \underbrace{d \rightarrow (e \rightarrow (d \rightarrow e))}_{(a,b) \quad c}$$

Es gilt somit :

$$\begin{aligned} \text{i) } & \underbrace{d \rightarrow e \rightarrow (d,e) \equiv (d \rightarrow (e \rightarrow (d \rightarrow e)))}_{\text{da } \rightarrow \text{ rechtsassoziativ}} \equiv (a,b) \rightarrow c \\ \text{ii) } & d \equiv (a,b) \\ \text{iii) } & (e \rightarrow (\underbrace{d}_{\text{ii) (a,b)}} \rightarrow e)) \equiv c \quad \Rightarrow (e \rightarrow ((a,b) \rightarrow e)) \equiv c \end{aligned}$$

Signatur von curry (curry id):

$$\begin{aligned} \text{curry (curry id)} & :: a \rightarrow b \rightarrow \underbrace{c}_{(e \rightarrow (\underbrace{d}_{(a,b)} \rightarrow e))} \\ \Rightarrow \text{curry (curry id)} & :: a \rightarrow b \rightarrow e \rightarrow ((a,b) \rightarrow e) \end{aligned}$$

### uncurry (uncurry id)

$$\text{uncurry} :: \underbrace{\underbrace{a \rightarrow (b \rightarrow c)}_{(d \rightarrow e), d} \rightarrow e}_{((d \rightarrow e), d) \rightarrow e} \rightarrow (a,b) \rightarrow c \quad \text{uncurry id} :: \underbrace{((d \rightarrow e), d) \rightarrow e}_{a \rightarrow (b \rightarrow c)}$$

Es gilt somit :

$$\begin{aligned} \text{i) } & \underbrace{a \rightarrow b \rightarrow c \equiv a \rightarrow (b \rightarrow c)}_{\text{da } \rightarrow \text{ rechtsassoziativ}} \equiv ((d \rightarrow e), d) \rightarrow e \\ \text{ii) } & e \equiv (b \rightarrow c) \\ \text{iii) } & ((d \rightarrow \underbrace{e}_{\text{ii) (b \rightarrow c)}}), d) \equiv a \quad \Rightarrow ((d \rightarrow (b \rightarrow c)), d) \equiv a \end{aligned}$$

Signatur von uncurry (uncurry id):

$$\begin{aligned} \text{uncurry (uncurry id)} & :: (\underbrace{a}_{((d \rightarrow \underbrace{e}_{(b \rightarrow c)}), d)}) \rightarrow b \rightarrow c \\ \Rightarrow \text{uncurry (uncurry id)} & :: ((d \rightarrow (b \rightarrow c)), d, b) \rightarrow c \end{aligned}$$

### uncurry curry

$$\text{uncurry} :: \left( \underbrace{a}_{((d,e) \rightarrow f)} \rightarrow \underbrace{b}_d \rightarrow \underbrace{c}_{e \rightarrow f} \right) \rightarrow (a,b) \rightarrow c \qquad \text{curry} :: \underbrace{((d,e) \rightarrow f) \rightarrow (d \rightarrow (e \rightarrow f))}_{a \rightarrow b \rightarrow c}$$

Es gilt somit :

$$\text{i) } a \rightarrow b \rightarrow c \equiv \underbrace{((d,e) \rightarrow f) \rightarrow d \rightarrow e \rightarrow f}_{\text{da } \rightarrow \text{ rechtsassoziativ}} \equiv ((d,e) \rightarrow f) \rightarrow (d \rightarrow (e \rightarrow f))$$

$$\text{ii) } b \equiv d$$

$$\text{iii) } a \equiv \left( \underbrace{d}_{\text{ii) } b}, e \right) \rightarrow f$$

$$\text{iv) } e \rightarrow f \equiv c$$

Signatur von uncurry curry :

$$\text{uncurry curry} :: \left( \underbrace{a}_b, b \right) \rightarrow \underbrace{c}_{e \rightarrow f}$$

$$\Rightarrow \text{uncurry curry} :: ((b,e) \rightarrow f, b) \rightarrow e \rightarrow f$$

### c) Folgende Funktionen sind nicht unifizierbar

#### curry uncurry

$$\text{curry} :: \left( \left( \underbrace{a,b}_{d \rightarrow e \rightarrow f} \right) \rightarrow \underbrace{c}_{(d,e) \rightarrow f} \right) \rightarrow a \rightarrow b \rightarrow c \qquad \text{uncurry} :: \underbrace{(d \rightarrow e \rightarrow f) \rightarrow ((d,e) \rightarrow f)}_{(a,b) \rightarrow c}$$

Es gilt somit :

$$\text{i) } (a,b) \rightarrow c \equiv \underbrace{(d \rightarrow e \rightarrow f) \rightarrow (d,e) \rightarrow f}_{\text{da } \rightarrow \text{ rechtsassoziativ}} \equiv (d \rightarrow e \rightarrow f) \rightarrow ((d,e) \rightarrow f)$$

$$\text{ii) } ((d,e) \rightarrow f) \equiv c \qquad \text{o.k.}$$

$$\text{iii) } (d \rightarrow e \rightarrow f) \equiv (a,b) \quad \text{nicht unifizierbar!}$$

$\Rightarrow$  curry uncurry hat keinen eindeutigen Typ

#### curry curry

$$\text{curry} :: \underbrace{\left( \underbrace{(a,b) \rightarrow f}_{(d,e) \rightarrow f} \right) \rightarrow c}_{(d,e) \rightarrow f \rightarrow (d \rightarrow (e \rightarrow f))} \rightarrow a \rightarrow b \rightarrow c \quad \text{curry} :: \underbrace{\left( (d,e) \rightarrow f \right) \rightarrow (d \rightarrow e \rightarrow f)}_{(a,b) \rightarrow c}$$

Es gilt somit :

i)  $(a,b) \rightarrow c \equiv \underbrace{\left( (d,e) \rightarrow f \right) \rightarrow d \rightarrow e \rightarrow f}_{\text{da } \rightarrow \text{ rechtsassoziativ}} \equiv \left( (d,e) \rightarrow f \right) \rightarrow (d \rightarrow e \rightarrow f)$

ii)  $(d \rightarrow e \rightarrow f) \equiv c$  o.k.

iii)  $((d,e) \rightarrow f) \equiv (a,b)$  nicht unifizierbar!

$\Rightarrow$  curry curry hat keinen eindeutigen Typ

Unifizierung mit  $(d,e) \rightarrow f \rightarrow d \rightarrow e \rightarrow f$  wäre möglich.

#### Aufgabe 11.4

a.  $((\lambda z.zxx)z x)$

Ergebnis:

$$((\lambda z_1.z_1 x_1 x_1) z_2 x_2)$$

$z_1 z_2 x_2$  sind frei und nur  $z_2$  ist gebunden (siehe Indizes).

b.  $((\lambda x.(\lambda x.(\lambda x.x)x)x)x)$

Ergebnis:

$$((\lambda x_1.(\lambda x_2.(\lambda x_3.x_3)x_2)x_1)x_0)$$

$x_0$  ist frei, weil es in keinem Funktionsrumpf steht.

Die anderen Variablen sind gebunden.

c.  $(\lambda x.(xy(\lambda y.((xy)(\lambda x.wxyz))(\lambda z.wyz))))$

Ergebnis:

$$(\lambda x_1.(x_1 y_1 (\lambda y_2.((x_1 y_2) (\lambda x_2.w_1 x_2 y_2 z_1)) (\lambda z_2.w_2 y_2 z_2))))$$

$y_1, w_1, z_1, w_2$  sind frei. Alle anderen sind gebunden

d.  $(\lambda x.y(\lambda y.x(\lambda x.xz(\lambda y.yx))))$

Ergebnis:

$$(\lambda x_1.y_1 (\lambda y_2.x_1 (\lambda x_2.x_2 z_1 (\lambda y_3.y_3 x_2))))$$

$y_1, z_1$  sind frei. Die anderen sind gebunden.

Yark Schroeder, Ralf Anske, Abid Hussain, Patrick Schäfer et al.