TECHNISCHE UNIVERSITÄT DRESDEN

PROGRAMMIERUNG

Übung 1: Einführung in Haskell

Eric Kunze
eric.kunze@mailbox.tu-dresden.de

TU Dresden, 12. April 2019

Allgemeine Hinweise

Vorlesung:

- Freitag, 2. DS (9:20 - 10:50), HSZ/0003
- Skript und Aufgabensammlung im Copyshop „Die Kopie“

Github:
`https://github.com/oakoneric/programmierung-ss19`

Übungen:

- meine Übungsgruppen:
  Donnerstag, 1. DS und Freitag, 4. DS

Verlegungen:

Uns betreffen zwei Feiertage: **Karfreitag** (Freitag, 19.04.) und **Himmelfahrt** (Donnerstag, 30.05.)

Mögliche Alternativtermine:

- Montag, 3. DS
- Montag, 5. DS
- Mittwoch, 2. DS
- Donnerstag, 2. DS (gerade Woche, insbesondere am 18.04.)

Haskell installieren und compilieren

- **Glasgow Haskell Compiler** (ghc(i)) :
  `https://www.haskell.org/ghc/`
- Ubuntu:
  `sudo apt install ghc`
- **Terminal:**
  `ghci <modulname>`
- **Module laden:**
  `:load <modulname>`

Haskell & Listen

- Wenn a ein Typ ist, dann bezeichnet `[a]` den Typ “Liste mit Elementen vom Typ a”, insbesondere haben alle Elemente einer Liste den gleichen Typ

- **cons-Operator “ : ”** Trennung von head und tail einer Liste
  `[ x1 , x2 , x3 , x4 , x5] = x1 : [x2 , x3 , x4 , x5]`

- **Verkettungsoperator “ ++ ”** Verkettung zweier Listen gleichen Typs
  `[x1 , x2 , x3] ++ [x4 , x5 , x6 , x7] = [x1 , x2 , x3 , x4 , x5 , x6 , x7]`

Das Prinzip der Rekursion

Satz. Sei B eine Menge, $b \in B$ und $F\colon B \times \mathbb{N} \to B$ eine Abbildung. Dann liefert die Vorschrift

$$f(0) := b \tag{1a}$$

$$f(n+1) := F(f(n), n) \quad \forall n \in \mathbb{N} \tag{1b}$$

genau eine Abbildung $f : \mathbb{N} \to B$.

Beweis. vollständige Induktion:

(IA) Für $n = 0$ ist $f(0) = b$ eindeutig definiert.

(IS) Angenommen $f(n)$ sei eindeutig definiert. Wegen (1b) ist dann auch $f(n+1)$ eindeutig definiert.

Man kann dieses Prinzip der Rekursion auf weitere Mengen (unabhängig von den natürlichen Zahlen) erweitern (↗ Formale Systeme).

Aufgabe 1 – Fakultätsfunktion

Fakultät

$$n! = \prod_{i=1}^{n} i \tag{2}$$

Daraus bilden wir nun eine Rekursionsvorschrift:

$$n! = \boldsymbol{n} \cdot \prod_{i=1}^{\boldsymbol{n-1}} i = n \cdot (n-1)! \tag{3}$$

Um die Rekursion vollständig zu definieren, benötigen wir einen (oder i.a. mehrere) **Basisfall**. Wann können wir also die Rekursion der Fakultät abbrechen?

$$0! = 1 \qquad 1! = 1 \qquad 2! = 2 \qquad \cdots \tag{4}$$

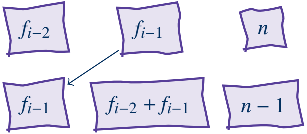
⇒ Welcher Basisfall ist sinnvoll? $0! = 1.$

Aufgabe 2 – Fibonacci-Zahlen

$$f_n := \begin{cases} 1 & \text{falls } n = 0 \\ 1 & \text{falls } n = 1 \\ f_{n-1} + f_{n-2} & \text{sonst} \end{cases} \tag{5}$$

$\Rightarrow$ Rekursionsvorschrift schon gegeben.

Verfahren ohne Rekursion.



Anmerkung.

Explizite Formel:

$$f_n = \frac{\Phi^n - \left(-\frac{1}{\Phi}\right)^n}{\sqrt{5}} \qquad \text{mit} \qquad \Phi = \frac{1 + \sqrt{5}}{2} \tag{6}$$