

# Einführung in die funktionale Programmierung

Wintersemester 2025/2026

## Aufgabenblatt Nr. 1

Abgabe: Dienstag 28.10.2025 bis 10:00, online in Moodle

### Aufgabe 1 (15 Punkte)

Sei  $s$  der folgende Ausdruck des Lambda-Kalküls:

$$s := (\lambda x, y, z. x\ y\ z) \ \lambda x, y. x\ \lambda y, z. z$$

- a) Schreiben Sie  $s$  ohne abkürzende Schreibweisen und mit voller Klammerung auf. (5 Punkte)
- b) Geben Sie  $s$  in Haskell-Notation an und geben Sie den entstandenen Ausdruck im Haskell-Interpreter `ghci` ein. (5 Punkte)
- c) Geben Sie einen  $\alpha$ -äquivalenten Ausdruck zu  $s$  an, der die *Distinct Variable Convention* erfüllt. (5 Punkte)

Hinweis: In Haskell ist die Schreibweise

`\x y z -> x y z` möglich anstelle von

`\x -> (\y -> (\z -> (x y) z))` d.h. man kann auch mehrstellige Funktionen direkter hinschreiben. Wir schreiben auch abkürzend (zum Beispiel)  $\lambda x, y, z. x\ y\ z$  für den Ausdruck  $\lambda x. \lambda y. \lambda z. x\ y\ z$  bzw.  $(\lambda x. (\lambda y. (\lambda z. ((x y) z))))$ , und entsprechend für andere Varianten.

Sie können für die Aufgabe auch das Haskell-Programm `LExp.hs` verwenden, das auf der Webseite der Vorlesung bereitgestellt wird. z.B. `run "(\lambda x. x y z) u"` ergibt  $((u\ y)\ z)$ .

Vorsicht: `LExp.hs` hat etwas andere (erweiterte) Syntax)

### Aufgabe 2 (20 Punkte)

Die Collatz-Funktion kann in Haskell programmiert werden durch:

```
collatz 1 = 1
collatz n = if even n
            then collatz (n `div` 2)
            else collatz (3*n+1)
```

Fügen Sie obigen Programmcode in eine Datei namens `Collatz.hs` ein und laden Sie das Programm anschließend im `ghci`<sup>1</sup>.

Testen Sie anschließend die Funktion für mindestens 10 verschiedene Werte.

Anmerkung: Ein offenes Problem ist ob die Collatz-Funktion gleich ist zur (konstanten) Funktion die bei jeder Eingabe von natürlichen Zahlen immer 1 ausgibt.

Zum Debuggen würde man gerne die Zwischenwerte von  $n$  bei jedem Rekursionsschritt sehen. Die Haskell-Bibliothek `Debug.Trace` stellt die Funktion `trace :: String -> a -> a` bereit, die zwei Argumente erwartet und das erste (als Debug-Output) ausdrückt und anschließend das zweite als Resultat liefert.

Verwenden Sie `trace` in geeigneter Weise, um die Zwischenwerte von  $n$  auf der Konsole auszudrucken. Testen Sie die modifizierte Funktion erneut.

### Hinweise:

- Das Modul `Debug.Trace` kann eingebunden werden, indem Sie am Anfang des Quelltextes die Zeile

```
import Debug.Trace
```

einfügen.

- Zum Konvertieren einer Zahl in einen `String` können Sie die vordefinierte Funktion `show` verwenden.

## Aufgabe 3 (15 Punkte)

Lambda-Abstraktionen sind *anonyme Funktionen*, also Funktionen die keinen Namen haben. Z.B. könnten wir für das Haskell-Programm

```
fun1 x y = x + 2*y
fun2 x y z = (fun1 x z) - (fun1 (2*x) y)
fun3 f g x = f (g x) (g x)
```

und den Aufruf `fun1 (3-1) 5` auch den gleichwertigen Ausdruck

```
(\x y -> x + 2*y) (3-1) 5
```

eingeben.

- Geben Sie für die folgenden Aufrufe gleichwertige Ausdrücke an, ohne die Funktionen `fun1`, `fun2` und `fun3` zu verwenden und geben Sie die entstandenen Ausdrücke im `ghci` ein:
  - `fun1 (2*6) (3*8)` (3 Punkte)
  - `fun3 fun1 (fun2 1 2) 3` (5 Punkte)
  - `fun1 (fun2 1 2 3) (fun1 2 7)` (5 Punkte)
- Welche Probleme entstehen, wenn Sie `collatz 5` (siehe Aufgabe 2) in einen solchen Ausdruck überführen möchten? (2 Punkte)

---

<sup>1</sup>Wenn Sie nicht wissen, wie das funktioniert, verwenden Sie die Hilfe des `ghci`, indem Sie `:?` im `ghci` eintippen