

Übungen zu Funktionaler Programmierung

Übungsblatt 2

Ausgabe: 19.10.2018, **Abgabe:** 26.10.2018 – 16:00 Uhr, **Block:** 1

Das Übungsblatt behandelt Themen bis einschließlich Folie 35.

Hinweis: Um dieses Übungsblatt zu lösen, sind folgende Äquivalenzen hilfreich:

$x \otimes y \Leftrightarrow (\otimes) \ x \ y$	(Operator als Funktion)
$f \ x \ y \Leftrightarrow x \ `f` \ y$	(Funktion als Operator)
$\backslash x \rightarrow \dots \backslash z \rightarrow e \Leftrightarrow \backslash x \dots z \rightarrow e$	(λ -Ausdrücke zusammenfassen)
$f = \backslash x \dots z \rightarrow e \Leftrightarrow f \ x \dots z = e$	(Applikative Definition)
$f \ x_1 \dots z_1 \mid g_1 = e_1 \Leftrightarrow f \ x \dots z =$	(Patternmatching-Umformung)
\vdots	
$\text{case } (x, \dots, z) \text{ of}$	
$f \ x_N \dots z_N \mid g_N = e_N$	$(x_1, \dots, z_1) \mid g_1 \rightarrow e_1$
\vdots	
	$(x_N, \dots, z_N) \mid g_N \rightarrow e_N$

Aufgabe 2.1 (6 Punkte) λ -Ausdrücke auswerten

Werten Sie folgende Ausdrücke *schrittweise* und *lazy* (*leftmost-outermost*) aus.

- $(\backslash x \ y \rightarrow y \ x) \ a \ b$
- $(\backslash y \ z \rightarrow z) \ ((\backslash x \rightarrow x \ x) \ (\backslash x \rightarrow x \ x)) \ a$
- $(\backslash f \ g \ x \rightarrow f \ (g \ x)) \ (\backslash y \rightarrow y \ y) \ (\backslash z \rightarrow a)$

Aufgabe 2.2 (4 Punkte) Fallunterscheidungen

Implementieren Sie folgende Funktionen in Haskell mit der `case_of`-Syntax und geben Sie die Typen der Funktionen an.

- Benutzen Sie für folgende Funktion die *Fallunterscheidungen nach Muster*.

$$f(e) = \begin{cases} x + y, & \text{falls } e = \text{Left}(x, y) \text{ und } x, y \in \text{Float} \\ x(pt) + y(pt), & \text{falls } e = \text{Right}(pt) \text{ und } pt \in \text{Point} \end{cases}$$

- Benutzen Sie für folgende Funktion die *Fallunterscheidungen nach Bedingung*.

$$g(x, y) = \begin{cases} x * y, & \text{falls } x \text{ gerade} \\ x - y, & \text{falls } x > 50, y > 100 \text{ und } x \text{ ungerade} \\ x/y, & \text{falls } y \neq 0 \text{ und } x \text{ ungerade} \\ x + y, & \text{sonst} \end{cases}$$

Sie können die Haskell-Funktion `div` und `even` benutzen.

Aufgabe 2.3 (6 Punkte) *Präfixdarstellung*

Fügen Sie die impliziten Klammern in folgende Haskell-Ausdrücke ein. Wandeln Sie danach den Ausdruck in seine Präfixdarstellung.

- a) `x + y + 5 * z`
- b) `f . g $ h $ f x`
- c) `f 5 True 3`

Aufgabe 2.4 (8 Punkte) *Haskell-Funktion auswerten*

Werten Sie den folgende Ausdrücke aus, indem Sie erst den Operator *schrittweise* in einen λ -Ausdruck umformen und dann den Ausdruck *schrittweise* auswerten.

- a) `False || True` Die Disjunktion ist wie folgt definiert.

```
(||) :: Bool -> Bool -> Bool
True  || _ = True
False || x = x
```

- b) `h . h'` Der Kompositionsoperator ist auf Folie 32 definiert.