technische universität dortmund

FLDIT

Lehrstuhl 1
Logik in der Informatik

Prof. Dr. Peter Padawitz

Jos Kusiek (*jos.kusiek@tu-dortmund.de*)
Lukasz Czajka (*lukasz.czajka@tu-dortmund.de*)

Wintersemester 2018/2019

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 ⊗ y ⇔ (⊗) x y                                   (Operator als Funktion)
f x y ⇔ x `f` y                                   (Funktion als Operator)
\x -> ... \z -> e ⇔ \x ... z -> e                 (λ-Ausdrücke zusammenfassen)
f = \x ... z -> e ⇔ f x ... z = e                 (Applikative Definition)

f x1 ... z1 | g1 = e1  ⇔ f x ... z =              (Patternmatching-Umformung)
          ⋮                  case (x, ..., z) of
f xN ... zN | gN = eN          (x1,...,z1) | g1 -> e1
                                      ⋮
                               (xN,...,zN) | gN -> eN
```

Aufgabe 2.1 (6 Punkte) *λ-Ausdrücke auswerten*
Werten Sie folgende Ausdrücke *schrittweise* und *lazy (leftmost-outermost)* aus.

a) `(\x y -> y x) a b`

b) `(\y z -> z) ((\x -> x x) (\x -> x x)) a`

c) `(\f g x -> f (g x)) (\y -> y y) (\z -> 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.

a) Benutzen Sie für folgende Funktion die *Fallunterscheidungen nach Muster*.

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

b) Benutzen Sie für folgende Funktion die *Fallunterscheidungen nach Bedingung*.

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

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.