technische universität dortmund

FLDIT

Lehrstuhl 1
Logik in der Informatik

Prof. Dr. Peter Padawitz

Jos Kusiek (*jos.kusiek@tu-dortmund.de*) Wintersemester 2017/2018

Übungen zu Funktionaler Programmierung
Übungsblatt 1

Ausgabe: 13.09.2017, **Abgabe:** 20.09.2017 – 16:00 Uhr

Das `case`-Konstrukt wird im Abschnitt *Gleichungen, die Funktionen definieren* auf den Folien 19–22 vorgestellt. Bitte lesen Sie diesen Abschnitt selbstständig.

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 1.1 (3 Punkte) *Typeinferenz*
Berechnen Sie die Typen der folgenden Ausdrücke mithilfe der Typinferenzregeln.

a) `(\(x,y) -> Just 3) (3,3)` mit `3 :: Int`

b) `\f g x -> f (g x)`

Aufgabe 1.2 (3 Punkte) *λ-Ausdrücke Auswerten*
Werten Sie folgende Ausdrücke schrittweise aus.

a) `(\x y -> x * y) 3 2`

b) `(\f g x -> f (g x)) (\y -> y * 2) (\z -> z + 1)`

Aufgabe 1.3 (3 Punkte) *Haskell-Funktion Auswerten*
Gegeben sei folgende Haskell-Funktion:

```
and' :: Bool -> Bool -> Bool
and' False _ = False
and' True  b = b
```

Werten Sie den Ausdruck `and' True True` aus, indem Sie erst `and'` in einen λ-Ausdruck umformen und dann schrittweise auswerten.

Aufgabe 1.4 (3 Punkte) *Haskell-Funktionen definieren*
Schreiben Sie eine Haskell-Funktionen `ite` vom Typ `Bool -> Int -> Int -> Int`, welche die erste Ganzzahl (`Int`) zurückgibt, falls der erste Parameter vom Wert `True` ist und die zweite Ganzzahl sonst.
Beispiele:

```
ite True 3 9 ~> 3
ite False 3 9 ~> 9
```

a) Definieren Sie die Funktion als λ-Ausdruck mit `case`.

b) Definieren Sie die Funktion applikativ.

Definieren Sie die Funktionen ohne `if_then_else_`-Ausdruck.