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 10

Ausgabe: 22.12.2017, **Abgabe:** 12.1.2017 – 16:00 Uhr, **Block:** 5

Aufgabe 10.1 (2 Punkte) *Funktor*
Schreiben Sie eine Instanz der Klasse `Functor` für den folgenden Datentyp für nichtleere binäre Bäume:

```
data STree a = BinS (STree a) a (STree a) | LeftS (STree a) a
             | RightS a (STree a) | LeafS a deriving Show
```

Beispiel: `fmap succ $ BinS (LeftS (LeafS 9) 2) 4 (RightS 7 (LeafS 3))`
`  ⇝ BinS (LeftS (LeafS 10) 3) 5 (RightS 8 (LeafS 4))`

Lösungsvorschlag

```
instance Functor STree where
  fmap f t = case t of
    BinS l a r -> BinS (fmap f l) (f a) (fmap f r)
    LeftS l a -> LeftS (fmap f l) (f a)
    RightS a r -> RightS (f a) (fmap f r)
    LeafS a -> LeafS (f a)
```

Aufgabe 10.2 (3 Punkte) *do-Notation*
Gegeben sei folgende Listenkomprehension:

```
solutions :: [(Int , Int , Int )]
solutions = [ (x,y,z) | z <- [0..] , y <- [0..z^2] , x <- [0..z^2]
            , 2*x^3 + 5*y + 2 == z^2]
```

a) Überführen Sie die Listenkomprehension in die do-Notation.

b) Überführen Sie die do-Notation in monadische Operatoren und Funktionen (>>=,>>,`return`).

Lösungsvorschlag

a)
```
solutions' :: [(Int , Int , Int )]
solutions' = do
   z <- [0..]
   y <- [0..z^2]
   x <- [0..z^2]
   guard $ 2*x^3 + 5*y + 2 == z^2
   return (x,y,z)
```

b)
```
solutions'' :: [(Int , Int , Int )]
solutions'' =
  [0..] >>= \z ->
  [0..z^2] >>= \y ->
  [0..z^2] >>= \x ->
  (guard $ 2*x^3 + 5*y + 2 == z^2) >>
  return (x,y,z)
```

Aufgabe 10.3 (3 Punkte) *Plusmonade*
Implementieren Sie die Funktion `preorderM` als Verallgemeinerung der Funktion `preorderB`:

`preorderM :: MonadPlus m => Bintree a -> m a`.

Die Funktion `preorderM` soll sich bei einer Festlegung des Rückgabetyps auf eine Liste von Werten wie `preorderB` verhalten:

`preorderM $ Fork 3 (leaf 4) (leaf 5) :: [Int]` $\rightsquigarrow$ `[3,4,5]`

Darüber hinaus soll `preorderM` aber beispielsweise auch für einen in `Maybe` eingebetteten Wert funktionieren und dann den Wert des Wurzelknotens zurückgeben:

`preorderM $ Fork 3 (leaf 4) (leaf 5) :: Maybe Int` $\rightsquigarrow$ `Just 3`

Ist der Baum leer, gibt die Funktion für den Rückgabetyp `Maybe Int` den Fehlerwert `Nothing` zurück.

Lösungsvorschlag

```
preorderM :: MonadPlus m => Bintree a -> m a
preorderM Empty = mzero
preorderM (Fork a l r) = msum [return a, preorderM l, preorderM r]
```

> Für die folgenden Aufgaben lesen Sie bitte selbstständig den Abschnitt *Maybe- und Listenmonade* auf den Folien 189 und 190.

Aufgabe 10.4 (2 Punkte) *`Maybe`-Monade*
Gegeben sei die sichere Division `sdiv`.

```
sdiv :: Int -> Int -> Maybe Int
_ `sdiv` 0 = Nothing
x `sdiv` y = Just $ x `div` y
```

Definieren Sie die Funktion `f` vom Typ `Int -> Int -> Int -> Maybe Int`, welche die Gleichung

$$f(x,y,z) = \frac{\frac{18}{x}}{y} + \frac{6}{z}$$

erfüllt. Für die Divisionen muss die Funktion `sdiv` benutzt werden. Nutzen Sie nur die Monadeneigenschaft von `Maybe` und lösen Sie die Aufgabe in der `do`-Notation.
Beispiel: `f 2 3 2` $\rightsquigarrow$ `Just 6`

Lösungsvorschlag

```
f :: Int -> Int -> Int -> Maybe Int
f x y z = do
  r1 <- 18 `sdiv` x
  r2 <- r1 `sdiv` y
  r3 <- 6 `sdiv` z
  return $ r2 + r3
```

Aufgabe 10.5 (2 Punkte) *Listenmonade*
Gegeben sei die nichtdeterministische Transitionsfunktion `delta`, welche auf `graph1` basiert.

```
G _ delta = graph1
```

Definieren Sie die Funktion `g` vom Typ `Int -> Int -> [Int]`, welche die Gleichung

$$g(x, y) = \delta(x) * \delta(\delta(y))$$

erfüllt. Nutzen Sie nur die Monadeneigenschaft der Liste und lösen Sie die Aufgabe in der `do`-Notation.
Beispiel: `g 1 3` $\rightsquigarrow$ `[4,6,2,4,8,10,6,9,3,6,12,15]`

Lösungsvorschlag

```
g :: Int -> Int -> [Int]
g x y = do
  r1 <- delta x
  r2 <- delta y
  r3 <- delta r2
  return $ r1 * r3
```

Frohes Fest und einen guten Rutsch ins neue Jahr