Real World Haskell

Blatt 2

Julian Fleischer, Alexander Steen

Montag, den 23.07.2013

Aufgabe 1 (**do-Syntax**) Formen Sie die folgenden Ausdrücke korrekt von do-Notation in `>>=` -Schreibweise und umgekehrt um:

(0)

```
evalM :: Monad m => Expr -> m Float
evalM = evalExpr fC fA fD
  where fC = return
    fA m1 m2 = m1 >>= (\x -> m2 >>= (\y -> ... {- kommt auf die Monade an -})
    fD m1 m2 = m1 >>= (\x -> m2 >>= (\y -> ... {- kommt auf die Monade an -})
```

(1)

```
main = do
    putStrLn "Hallo Welt"
    putStrLn "da draussen"
```

(2)

```
headSafe [] >>= return . (+1)
```

wobei `headSafe :: [a] -> Maybe a`.

(3)

```
f = do
    x
    c <- z
    let g z = z in
        g c
```

Aufgabe 2 (**Either-Monade**)
Definieren Sie den Datentyp Either als Monade:
`instance Monad (Either String) where ...`
Erfüllt ihre Monade die Monadengesetze?

Aufgabe 3 (**Taschenrechner mit Fehlerbehandlung**) Benutzen Sie die Either-Monade aus Aufgabe 2 um Ihre Funktion `calculate` (Blatt 1, Aufgabe 2) um Fehlerbehandlung (z.B. beim Teilen durch Null) zu erweitern. Welche Veränderungen (z.B. Typsignatur, ...) sind hierfür zunächst notwendig?

Aufgabe 4 (**Taschenrechner mit Trace**) Betrachten Sie den Datentyp `data Trace a = T a String`, mit dem man eine Protokollierung von Berechnungen (Trace) umsetzen kann. Dabei ist das Ergebnis einer Berechnung `T p s`, wobei `p` der eigentliche Wert der Berechnung und `s` der erzeugte Trace ist. Zeigen Sie, dass `Trace` eine Monade ist, indem Sie eine `instance`-Delaration angeben.
Schreiben Sie nun zunächst eine Funktion `trace :: String -> Trace ()`, die einen String in einen Trace aufnimmt.
Benutzen Sie nun die Trace-Monade und `trace` um Ihre Funktion `calculate` alle Zwischenergebnisse der Berechnung ausgeben zu lassen.

Aufgabe 5 (**Cabal**) Verpacken Sie unter Nutzung von Cabal Ihre Either- und Trace-Monade als Bibliothek. Dokumentieren Sie diese Bibliothek samt etwaigen utility-Funktionen angemessen.

Aufgabe 6 (**Stateful transformation**)
Gegeben ist folgender Datentyp für einen polymorphen Binärbaum:
`data Tree a = Leaf | Node (Tree a) a (Tree a)`
Schreiben Sie mit Hilfe der Statemonade aus der Vorlesung eine Funktion
`label :: Eq a => Tree a -> Tree Int`, die die Knotenwerte durch natürliche Zahlen ersetzt. Dabei sollen gleiche Knotenwerte durch die gleiche Zahl ersetzt werden.

Definieren sie zunächst einen geeigneten Zustandstypen sowie einen Startzustand für die Traversierung. Den Code der Statemonade finden Sie auf der Veranstaltungsseite.