technische universität dortmund

FLDIT

Prof. Dr. Peter Padawitz

Lehrstuhl 1
Logik in der Informatik

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

Wintersemester 2018/2019

Übungen zu Funktionaler Programmierung
Übungsblatt 7

Ausgabe: 23.11.2018, **Abgabe:** 30.11.2018 – 16:00 Uhr, **Block:** 3

Aufgabe 7.1 (6 Punkte) *Arithmetische Ausdrücke*

a) Definieren Sie eine Konstante `expr :: Exp String` für den Ausdruck $5x + 3y^2 + 10$. Stellen Sie den Ausdruck als Elemente vom Typ `Exp String` da. (2 Punkte)

b) Schreiben Sie die Listenkomprehension `solutions :: [(Int,Int,Int)]` um. Machen Sie sinnvollen Gebrauch von dem Ausdruck `expr` und der Funktion `foldArith evalAlg`. (4 Punkte)

Aufgabe 7.2 (6 Punkte) *Boolesche Ausdrücke*
Implementieren Sie folgende boolesche Ausdrücke als Haskell-Konstanten vom Typ `BExp String`.

a) `bexpr1` = $b \vee \neg b$

b) `bexpr2` = $(x \vee \mathit{false}) \wedge y$

c) `bexpr3` = $b \wedge (x \leq x + 10)$

Aufgabe 7.3 (6 Punkte) *Abstrakte Datentypen, Algebren und Faltungen*
Die Funktion `toInt` soll einen Wert vom Typ `PosNat` den entsprechenden Wert vom Typ `Int` wandeln. Dabei soll die Funktion Gebrauch von einer Faltung machen.

```
data PosNat = One | Succ' PosNat deriving Show

toInt :: PosNat -> Int
toInt = foldPosNat intAlg
```

Implementieren Sie den abstrakten Datentypen, die Algebra und die Faltungen. Gehen Sie in folgender Reihenfolge vor.

a) Implementieren Sie einen abstrakte Datentypen (eine Signatur) `PosNatSig` für den Datentypen `PosNat`.

b) Implementieren Sie eine Faltung `foldPosNat`, welche mithilfe einer Algebra Werte vom Typ `PosNat` auswertet (faltet).

c) Implementieren Sie die Algebra `intAlg` vom Typ `PosNatSig`, die einen Wert vom Typ `PosNat` den entsprechenden Wert vom Typ `Int` wandeln.

Aufgabe 7.4 (6 Punkte) *Boolesche Algebra*
Es sind die Typen und Funktionen `BStore`, `BExpSig`, `foldBExp` und `evalB` gegeben. Die genaue Definition finden Sie in der Vorgabedatei `Blatt07.hs`.

Definieren Sie eine boolesche Algebra

```
evalBAlg :: BExpSig x (Store x -> Int) (Store x -> BStore x -> Bool).
```

Orientieren Sie sich an der Definition der arithmetischen Algebra `evalAlg` (Folie 103). Bei korrekter Definition verhält sich die Funktion `evalB` ähnlich wie `eval` (Folie 104), bzw. `foldArith evalAlg`. Anstelle arithmetischer Ausdrücke werden aber boolesche Ausdrücke vom Typ `BExp x` ausgewertet. Die Funktion `evalB` benötigt zwei Variablenbelegungen. Eine für boolesche Variablen und die andere für arithmetische Variablen.
Beispiel: `evalB (\"x" -> 5) (\"b" -> True) bexpr3`