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 8

**Ausgabe:** 8.12.2017, **Abgabe:** 15.12.2017 – 16:00 Uhr, **Block:** 4

**Aufgabe 8.1** (3 Punkte) *Baumfaltungen*
Definieren Sie folgende Funktionen als Baumfaltungen (`foldBtree` bzw. `foldTree`) und geben Sie den Typ an. Wählen Sie den Typ möglichst allgemein.

a) `or_ :: Tree Bool -> Bool` – Verhält sich wie `or`. Enthält ein Knoten `True`, ist das Ergebnis `True`, sonst `False`.

b) `preorderB :: Bintree a -> [a]` – Die Knoten des Binärbaumes (`Bintree`) werden als Liste in Hauptreihenfolge (pre-order) wiedergegeben.

**Lösungsvorschlag**

```
or_ :: Tree Bool -> Bool
or_ = foldTree id (||) False (||)

preorderB :: Bintree a -> [a]
preorderB = foldBtree [] (\a l r -> a : l ++ r)
```

**Aufgabe 8.2** (3 Punkte) *Faltungen*

a) Schreiben Sie die Faltung `foldPosNat` für den Typen `PosNat`.

b) Schreiben Sie eine Funktion `toInt` die Werte vom Typ `PosNat` in entsprechende Werte vom Typ `Int` wandelt. Benutzen Sie dazu die Faltung `foldPosNat`.

**Lösungsvorschlag**

```
foldPosNat :: val -> (val -> val) -> PosNat -> val
foldPosNat val _ One = val
foldPosNat val f (Succ' n) = f (foldPosNat val f n)

toInt :: PosNat -> Int
toInt = foldPosNat 1 (+1)
```

**Aufgabe 8.3** (3 Punkte) *Arithmetische Ausdrücke kompilieren*
Geben Sie die Kommandosequenz für die Auswertung `execute (exp2code expr) ([],vars)` an. Zu jedem Kommando soll auch der Stapelinhalt (Stack) nach der Ausführung angegeben werden. Dabei sei der Ausdruck `expr` und die Belegungsfunktion `vars` wie folgt definiert:

```
expr :: Exp String
expr = Prod [Con 2, Con 9 :- Var "x"]

vars :: Store String
vars "x" = 6
```

Mit `getResult (execute (exp2code expr) ([],vars))` kann das Ergebnis angezeigt werden. Diese Hilfsfunktion wird für die Bearbeitung der Aufgabe nicht benötigt.

```
getResult :: Estate x -> Int
getResult = head . fst
```

**Lösungsvorschlag**

| Kommando | Stapel |
|---|---|
| `Push 2` | `[2]` |
| `Push 9` | `[9,2]` |
| `Load "x"` | `[6,9,2]` |
| `Sub` | `[3,2]` |
| `Mul 2` | `[6]` |

**Aufgabe 8.4** (3 Punkte) *Fixpunkte*
Gegeben sei folgender Datentyp für den Restklassenring $\mathbb{Z}_{10}$:

```
data Mod10 = Z0 | Z1 | Z2 | Z3 | Z4
           | Z5 | Z6 | Z7 | Z8 | Z9
           deriving (Show, Eq, Ord, Enum)

succ10 Z9 = Z0
succ10 x  = succ x

pred10 Z0 = Z9
pred10 x  = pred x
```

Definieren Sie folgende Funktionen mithilfe der Fixpunktfunktion `fixpt`.

a) `lfp, gfp :: Mod10` – Der kleinste bzw. größte Fixpunkt der Funktion `f`.

```
f :: Mod10 -> Mod10
f x | x < Z4 = succ10 x
    | x > Z6 = pred10 x
    | otherwise      = x
```

Aufgrund der Typklasse `Ord` wird `Mod10` ein CPO mit

$$\texttt{Z}i \texttt{ <= Z}j \equiv i \leq j \text{ für } i, j \in \mathbb{Z}_{10}$$

und `maximum` als Supremumsbildung.

b) `evens :: [Mod10]` – Die kleinste Lösung der Gleichung

$$evens = \{0\} \cup \{x + 2 \mid x \in evens\}\,.$$

Dabei entspricht `[Mod10]` dem Potenzmengenverband $\mathcal{P}(\mathbb{Z}_{10})$ (Folie 143). Es wird eine Schrittfunktion $\Phi : \mathcal{P}(\mathbb{Z}_{10}) \to \mathcal{P}(\mathbb{Z}_{10})$ bzw. `phi :: [Mod10] -> [Mod10]` benötigt. Die Fixpunkte der Funktion müssen den Lösungen der Gleichung entsprechen.

**Lösungsvorschlag**

```
lfp, gfp :: Mod10
lfp = fixpt (<=) f Z0
gfp = fixpt (>=) f Z9

evens :: [Mod10]
evens = fixpt subset phi [] where
  phi :: [Mod10] -> [Mod10]
  phi m = [Z0] `union` [succ10 (succ10 x) | x <- m]
```