Haskell Live

[05] Aufgabenblatt 3 [doch nicht :-]

Bong Min Kim
e0327177@student.tuwien.ac.at

Christoph Spörk
christoph.spoerk@inode.at

Florian Hassanen
florian.hassanen@googlemail.com

Bernhard Urban
lewurm@gmail.com

5. November 2010

Algebraische Datentypen

Algebraische Datentypen Definitionen werden mit dem Schlüsselwort *data* eingeleitet. Die simplester Form eines algebraischen Datentyps ist die Enumeration der Elemente

```
import Data.Char
data Baum = Knoten2 Baum Baum |
   Knoten1 Baum |
   Blatt
   deriving (Eq, Show)
data Groesse = Gross | Klein | Winzig
```

```
  deriving Show
type BaumKlassifizierer = Baum → String

winzigBaum = Blatt
kleinBaum  = Knoten1
               Blatt
kleinBaum2 = Knoten2
               Blatt
               Blatt
grossBaum = Knoten2
               (Knoten1
                  Blatt)
               Blatt
grossBaum2 = Knoten1
               (Knoten2 Blatt
                  (Knoten1 Blatt))
grossBaum3 = Knoten2
               (Knoten2
                  Blatt
                  Blatt)
               (Knoten2 Blatt Blatt)

showP :: Baum → String
showP baum = "(" ++ show baum ++ ")"

var1 :: BaumKlassifizierer
var1 Blatt         = showP Blatt ++ " ist " ++ show Winzig
var1 (Knoten1 x) = showP (Knoten1 x) ++ " ist " ++ if (x ≡ Blatt)
                      then show Klein
                      else show Gross
```

```
var1 (Knoten2 x y) = showP (Knoten2 x y) ⧺ " ist " ⧺ if (x ≡ Blatt ∧ y ≡ Blatt)
  then show Klein
  else show Gross

var2 :: BaumKlassifizierer
var2 Blatt                 = showP Blatt ⧺ " ist " ⧺ show Winzig
var2 (Knoten1 Blatt) = showP (Knoten1 Blatt) ⧺ " ist " ⧺ show Klein
var2 (Knoten2 Blatt Blatt) = showP (Knoten2 Blatt Blatt) ⧺ " ist " ⧺ show Klein
  -- var2 (Knoten2 (Knoten2 a b) (Knoten2 c d)) = showP (Knoten2 (Knoten2 a b) (Knoten2 c d)) ++ ïst exorbitant "++ show Gross
var2 baum                  = showP baum ⧺ " ist " ⧺ show Gross

reminder ganzes@(erstes : rest) = "ganzes = " ⧺ show ganzes ⧺ " erstes = " ⧺ show erstes ⧺ " rest = " ⧺ show rest

var3 :: BaumKlassifizierer
var3 baum@Blatt                 = showP baum ⧺ " ist " ⧺ show Winzig
var3 baum@(Knoten1 Blatt) = showP baum ⧺ " ist " ⧺ show Klein
var3 baum@(Knoten2 Blatt Blatt) = showP baum ⧺ " ist " ⧺ show Klein
  -- var3 baum@(Knoten2 (Knoten2 _ _) (Knoten2 _ _)) = showP baum ++ ïst exorbitant "++ show Gross
var3 baum                       = showP baum ⧺ " ist " ⧺ show Gross

var4 :: BaumKlassifizierer
var4 baum =
  case baum of
    Blatt                 → showP baum ⧺ " ist " ⧺ show Winzig
    (Knoten1 Blatt) → showP baum ⧺ " ist " ⧺ show Klein
    (Knoten2 Blatt Blatt) → showP baum ⧺ " ist " ⧺ show Klein
  -- (Knoten2 (Knoten2 _ _) (Knoten2 _ _) -> showP baum ++ ïst exorbitant "++ show Gross
    _                     → showP baum ⧺ " ist " ⧺ show Gross
```

```
mix :: String → String → String
mix links rechts =
  case (links, rechts) of
    ((l : ls), (r : rs)) → l : r : (mix ls rs)
    ("", "") → ""   -- not needed actually
    ("", _)  → rechts
    (_, "")  → links
test_mix = mix "aaaaaaaaaaaaa" "bbbbbbb"


  -- use responsiibly
wasZum__O_o x y =
  case x of
    (x : y : xs) → y
    [y]          → y
    []           → y


baumCreator :: String → Baum
baumCreator string = fst (baumCreatorHelper string)


baumCreatorHelper :: String → (Baum, String)
baumCreatorHelper ('b' : rest) = (Blatt, rest)
baumCreatorHelper ('1' : rest) = (Knoten1 baum, restrest)
                                 where (baum, restrest) = baumCreatorHelper rest
baumCreatorHelper ('2' : rest) = (Knoten2 links rechts, restrestrest)
                                 where (links, restrest) = baumCreatorHelper rest
                                   (rechts, restrestrest) = baumCreatorHelper restrest
baumCreatorHelper (_ : rest) = (baum, restrest)
                                 where (baum, restrest) = baumCreatorHelper rest
```

$\mathit{grossBaum3'} = \mathit{baumCreator}$ `"22bb2bb"`
$\mathit{grossBaum3''} = \mathit{baumCreator}$ `"2 2bb 2bb"`
$\mathit{nichtGrossBaum3} = \mathit{baumCreator}$ `"2 2bb 2b111b"`

$\mathit{testsuite} :: \mathit{BaumKlassifizierer} \rightarrow [\mathit{String}]$
$\mathit{testsuite}\ \mathit{baumKlassifizierer} = [\mathit{baumKlassifizierer}\ \mathit{testBaum} \mid \mathit{testBaum} \leftarrow [\mathit{winzigBaum}, \mathit{kleinBaum}, \mathit{kleinBaum2}, \mathit{grossBaum}, \mathit{grossBaum2}$
$\mathit{testVar1} = \mathit{testsuite}\ \mathit{var1}$
$\mathit{testVar2} = \mathit{testsuite}\ \mathit{var2}$
$\mathit{testVar3} = \mathit{testsuite}\ \mathit{var3}$
$\mathit{testVar4} = \mathit{testsuite}\ \mathit{var4}$

Beispiel

1. $result \leftarrow baumCreator(''21bb'')$

2. $result \leftarrow fst(helperResult)$
   $helperResult \leftarrow baumCreatorHelper(''21bb'')$
   $helperResult \leftarrow baumCreatorHelper('2' : ''1bb'')$

3. $helperResult \leftarrow ((Knoten2 \quad links_1 \quad rechts_1), restrestrest_1)$
   $(links_1, restrest_1) \leftarrow baumCreatorHelper(''1bb'')$
   $(links_1, restrest_1) \leftarrow baumCreatorHelper('1' : ''bb'')$
   $(rechts_1, restrestrest_1) \leftarrow baumCreatorHelper(restrest_1)$

4. $(links_1, restrest_1) \leftarrow ((Knoten1 \quad baum_2), restrest_2)$
   $(baum_2, restrest_2) \leftarrow baumCreatorHelper(''bb'')$
   $(baum_2, restrest_2) \leftarrow baumCreatorHelper('b' : ''b'')$

5. $(baum_2, restrest_2) \leftarrow (Blatt, ''b'')$

6. $\Rightarrow$
   $(links_1, restrest_1) \leftarrow ((Knoten1 \quad Blatt), "b")$

7. $\Rightarrow$
   $(rechts_1, restrestrest_1) \leftarrow baumCreatorHelper("b")$
   $(rechts_1, restrestrest_1) \leftarrow baumCreatorHelper('b' : "")$

8. $(rechts_1, restrestrest_1) \leftarrow (Blatt, "")$

9. $\Rightarrow$
   $helperResult \leftarrow ((Knoten2 \quad (Knoten1 \quad Blatt) \quad Blatt), "")$

10. $\Rightarrow$
    $result \leftarrow fst(((Knoten2 \quad (Knoten1 \quad Blatt) \quad Blatt), ""))$
    $result \leftarrow (Knoten2 \quad (Knoten1 \quad Blatt) \quad Blatt)$