Tüübiklassid

Motivatsioon

- Aritmeetilised operaatorid:

$$(+) :: Int \to Int \to Int$$
$$(+) :: Float \to Float \to Float$$

  - Tüüp $(+) :: a \to a \to a$ on liiga üldine!
- Polümorfne võrdus:

$$(\equiv) :: a \to a \to Bool$$

  - Kuidas kontrollida näiteks funktsioonide võrdust?
- Polümorfne väljastus:

$$show :: a \to String$$

  - Funktsioon *show* on igal tüübil defineeritud erinevalt!

Tüübiklassid

Klasside defineerimine

$$\textbf{class}\ [\mathit{context} \Rightarrow]\ \mathit{classname}\ \mathit{tvar}\ \textbf{where}\ \{\mathit{cbody}\}$$

- Ülemääratud operaatoreid defineeritakse *tüübiklassidega*
- Klassidefinitsiooni keha koosneb meetodide tüübisignatuuridest ja *default*-definitsioonidest

Klass Eq (definitsioon prelüüdist)

$$\begin{aligned}&\textbf{class}\ Eq\ a\ \textbf{where}\\ &\quad (\equiv), (\not\equiv)\ ::\ a \to a \to Bool\\ &\quad x \not\equiv y \quad = not\ (x \equiv y)\end{aligned}$$

NB!

Meetodide tüüpides esinevad tüübimuutujad on kitsendatud vastava klassikontekstiga

$$(\equiv)\ ::\ Eq\ a \Rightarrow a \to a \to Bool$$

Tüübiklassid

Klassikuuluvuse deklareerimine

$$\textbf{instance}\ [\mathit{context} \Rightarrow]\ \mathit{classname}\ \mathit{type}\ \textbf{where}\ \{\mathit{ibody}\}$$

- Deklareeritav tüüp peab olema kujul $tcon\ tvar_1 \ldots tvar_n$, kusjuures kõik tüübimuutujad peavad olema erinevad
- Deklaratsiooni keha koosneb meetodide definitsioonidest
- Default-definitsiooniga meetodi pole vaja (aga võib) uuesti defineerida

Tüübiklassid

Kahendlehtpuude võrdus

$$\textbf{data}\ \mathit{IntBtree} = \mathit{Lf}\ \mathit{Int} \mid \mathit{Br}\ \mathit{IntBtree}\ \mathit{IntBtree}$$

$$\begin{aligned}
&\textbf{instance}\ \mathit{Eq}\ \mathit{IntBtree}\ \textbf{where}\\
&\quad \mathit{Lf}\ i1 \equiv \mathit{Lf}\ i2 = i1 \equiv i2\\
&\quad \mathit{Br}\ t11\ t12 \equiv \mathit{Br}\ t21\ t22 = t11 \equiv t21 \wedge t12 \equiv t22\\
&\quad _ \equiv _ = \mathit{False}
\end{aligned}$$

Listide võrdus (defineeritud prelüüdis)

$$\begin{aligned}
&\textbf{instance}\ \mathit{Eq}\ a \Rightarrow \mathit{Eq}\ [a]\ \textbf{where}\\
&\quad [\,] \equiv [\,] = \mathit{True}\\
&\quad (x : xs) \equiv (y : ys) = x \equiv y \wedge xs \equiv ys\\
&\quad _ \equiv _ = \mathit{False}
\end{aligned}$$

Tüübiklassid

Põõsaste võrdus

$$\begin{aligned}
&\textbf{data}\ Bush\ a = One\ a \mid Two\ (Bush\ a)\ (Bush\ a) \\
&\qquad\qquad\qquad\qquad \mid Many\ [Bush\ a] \\
&\textbf{instance}\ Eq\ a \Rightarrow Eq\ (Bush\ a)\ \textbf{where} \\
&\quad One\ x \equiv One\ y = x \equiv y \\
&\quad Two\ x1\ x2 \equiv Two\ y1\ y2 = x1 \equiv y1 \wedge x2 \equiv y2 \\
&\quad Many\ xs \equiv Many\ ys = xs \equiv ys \\
&\quad _ \equiv _ = False
\end{aligned}$$

NB!

On kasutatud *kolme* erinevat võrdust!!!

$$(\equiv) :: a \to a \to Bool$$
$$(\equiv) :: Bush\ a \to Bush\ a \to Bool$$
$$(\equiv) :: [Bush\ a] \to [Bush\ a] \to Bool$$

Tüübiklassid

Aksioomid

Võrdusoperaator ($\equiv$) peaks olema refleksiive, sümmeetriline ja transitiivne; so. peaks täitma järgmisi aksioome:

$$
\begin{aligned}
x \equiv x &= \mathit{True} \\
x \equiv y &\Leftrightarrow y \equiv x \\
x \equiv y \wedge y \equiv z &\Rightarrow x \equiv z
\end{aligned}
$$

NB!

Haskelli kompilaator ei kontrolli, kas defineeritav ”isend” rahuldab antud aksioome või mitte, ning on puhtalt programmeerija ülesanne garanteerida, et ta seda tõepoolest teeb.

Tüübiklassid

Intuitsioon

- Klassidest võib mõelda, kui predikaatidest üle tüüpide
- Tüübid, mis rahuldavad neid predikaate, omavad "lisa funktsionaalsust"
- Klasside deklaratsioonid defineerivad mis liiki "lisa funktsionaalsusega" on tegemist
- "Isendite" deklaratsioonid defineerivad selle "lisa funktsionaalsuse" konreetse tüübi jaoks

Tüübiklassid

Klassikuuluvuse tuletamine

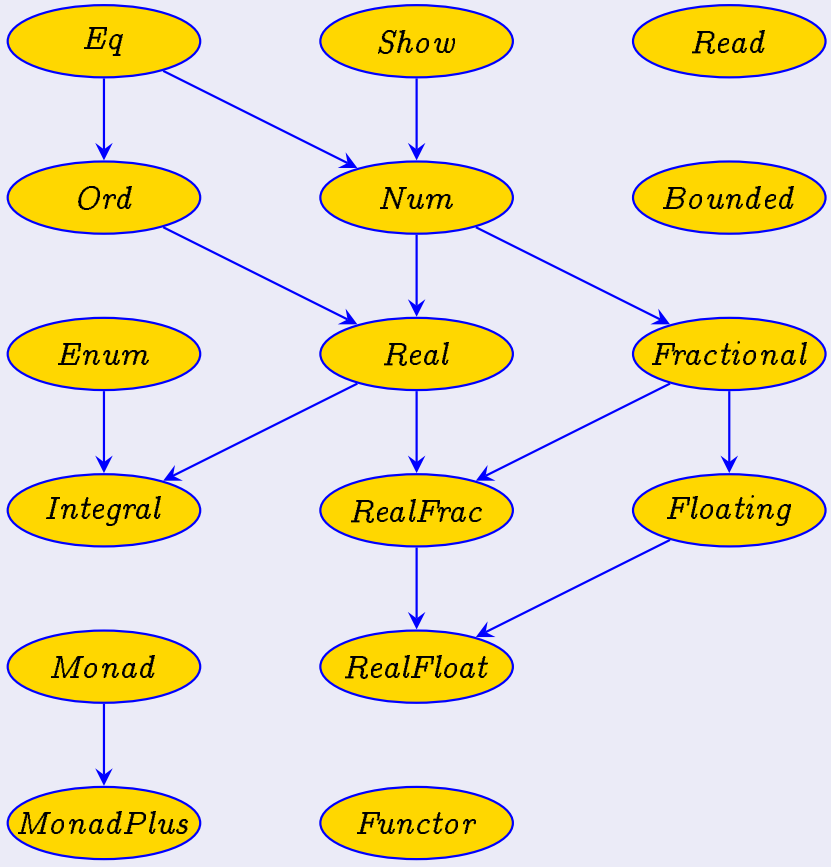
$$\begin{aligned}
&\textbf{data } \mathit{Bool} = \mathit{False} \mid \mathit{True} \\
&\quad \textbf{deriving } (\mathit{Eq}, \mathit{Ord}, \mathit{Ix}, \mathit{Enum}, \mathit{Read}, \mathit{Show}, \mathit{Bounded}) \\
&\textbf{data } \mathit{Maybe}\ a = \mathit{Nothing} \mid \mathit{Just}\ a \\
&\quad \textbf{deriving } (\mathit{Eq}, \mathit{Ord}, \mathit{Read}, \mathit{Show}) \\
&\textbf{data } \mathit{Either}\ a\ b = \mathit{Left}\ a \mid \mathit{Right}\ b \\
&\quad \textbf{deriving } (\mathit{Eq}, \mathit{Ord}, \mathit{Read}, \mathit{Show}) \\
&\textbf{data } \mathit{Ordering} = \mathit{LT} \mid \mathit{EQ} \mid \mathit{GT} \\
&\quad \textbf{deriving } (\mathit{Eq}, \mathit{Ord}, \mathit{Ix}, \mathit{Enum}, \mathit{Read}, \mathit{Show}, \mathit{Bounded})
\end{aligned}$$

NB!

deriving abil saab tuletada ainult eeldefineeritud klasse *Eq*, *Ord*, *Enum*, *Bounded*, *Show* ja *Read*.

Tüübiklassid

Eeldefineeritud klasside hierarhia



Tüübiklassid

Klass *Ord*

$$
\begin{aligned}
&\textbf{class } (Eq\ a) \Rightarrow Ord\ a \textbf{ where} \\
&\quad compare \qquad\qquad :: a \to a \to Ordering \\
&\quad (<), (\leqslant), (\geqslant), (>) :: a \to a \to Bool \\
&\quad max, min \qquad\qquad :: a \to a \to a \\
&\quad compare\ x\ y \mid x \equiv y \qquad = EQ \\
&\qquad\qquad\qquad\quad \mid x \leqslant y \qquad = LT \\
&\qquad\qquad\qquad\quad \mid otherwise = GT \\
&\quad x \leqslant y \quad = compare\ x\ y \not\equiv GT \\
&\quad x < y \quad = compare\ x\ y \equiv LT \\
&\quad x \geqslant y \quad = compare\ x\ y \not\equiv LT \\
&\quad x > y \quad = compare\ x\ y \equiv GT \\
&\quad max\ x\ y = \textbf{if } x \geqslant y \textbf{ then } x \textbf{ else } y \\
&\quad min\ x\ y = \textbf{if } x < y \textbf{ then } x \textbf{ else } y
\end{aligned}
$$

Tüübiklassid

Klass *Enum*

$$
\begin{array}{l}
\textbf{class } \mathit{Enum}\ a \textbf{ where} \\
\quad \begin{array}{lll}
\mathit{toEnum} & :: \mathit{Int} \to a & \\
\mathit{fromEnum} & :: a \to \mathit{Int} & \\
\mathit{enumFrom} & :: a \to [a] & \texttt{-- } [n\,..] \\
\mathit{enumFromThen} & :: a \to a \to [a] & \texttt{-- } [n, n'\,..] \\
\mathit{enumFromTo} & :: a \to a \to [a] & \texttt{-- } [n\,..\,m] \\
\mathit{enumFromThenTo} & :: a \to a \to a \to [a] & \texttt{-- } [n, n'\,..\,m]
\end{array} \\
\mathit{succ}, \mathit{pred} :: \mathit{Enum}\ a \Rightarrow a \to a \\
\mathit{succ} = \mathit{toEnum} \circ (+1) \circ \mathit{fromEnum} \\
\mathit{pred} = \mathit{toEnum} \circ (\mathit{subtract}\ 1) \circ \mathit{fromEnum}
\end{array}
$$

Arvudega seotud klassid

Klass *Num*

$$
\begin{aligned}
&\textbf{class}\ (Eq\ a, Show\ a) \Rightarrow Num\ a\ \textbf{where}\\
&\quad (+), (-), (*) :: a \to a \to a\\
&\quad negate\ \ \ \ \ \ \ :: a \to a\\
&\quad abs, signum\ :: a \to a\\
&\quad fromInteger :: Integer \to a\\
&\quad x - y = x + negate\ y
\end{aligned}
$$

Klass *Real*

$$
\begin{aligned}
&\textbf{class}\ (Num\ a, Ord\ a) \Rightarrow Real\ a\ \textbf{where}\\
&\quad toRational :: a \to Rational
\end{aligned}
$$

Arvudega seotud klassid

Klass *Integral*

$$\begin{aligned}&\textbf{class}\ (Real\ a, Enum\ a) \Rightarrow Integral\ a\ \textbf{where}\\ &\quad quot, rem \qquad\quad :: a \to a \to a\\ &\quad div, mod \qquad\quad :: a \to a \to a\\ &\quad quotRem, divMod :: a \to a \to (a, a)\\ &\quad toInteger \qquad\quad :: a \to Integer\end{aligned}$$

Klass *Fractional*

$$\begin{aligned}&\textbf{class}\ (Num\ a) \Rightarrow Fractional\ a\ \textbf{where}\\ &\quad (/) \qquad\quad :: a \to a \to a\\ &\quad recip \qquad\quad :: a \to a\\ &\quad fromRational :: Rational \to a\end{aligned}$$

NB!

Lisaks on eeldefineeritud *RealFrac*, *Floating*, *RealFloat*

Tüübiklassid

Klass *Show*

$$\textbf{type } ShowS = String \to String$$

$$\begin{aligned}
&\textbf{class } Show\ a \textbf{ where}\\
&\quad showsPrec :: Int \to a \to ShowS\\
&\quad showList \ \ :: [a] \to ShowS\\
&\quad show \qquad :: a \to String\\
&\quad show\ x \qquad\qquad = showsPrec\ 0\ x\ ""\\
&\quad showsPrec\ _\ x\ s = show\ x +\!\!+ s
\end{aligned}$$

Klass *Read*

$$\textbf{class } Read\ a \textbf{ where} \ldots$$

$$read :: (Read\ a) \Rightarrow String \to a$$

Klass *Show*

Arimteetiliste avaldiste näitamine

$$
\begin{aligned}
&showExp :: Expr \to String \\
&showExp\ (Num\ n) &&= show\ n \\
&showExp\ (Var\ v) &&= v \\
&showExp\ (e1 :+: e2) &&= showExp\ e1 \mathbin{+\!\!+} "+" \mathbin{+\!\!+} \\
&&&\quad showExp\ e2 \\
&showExp\ (e1 :*: e2) &&= showArg\ e1 \mathbin{+\!\!+} "*" \mathbin{+\!\!+} \\
&&&\quad showArg\ e2 \\
&showArg\ (e1 :+: e2) &&= "(" \mathbin{+\!\!+} \\
&&&\quad showExp\ (e1 :+: e2) \mathbin{+\!\!+} \\
&&&\quad ")" \\
&showArg\ e &&= showExp\ e
\end{aligned}
$$

NB!

Toodud definitsioon on ruutkeerukusega!

Klass *Show*

Eeldefineeritud abifunktsioone

$$
\begin{aligned}
&\mathit{shows} && :: \mathit{Show}\ a \Rightarrow a \to \mathit{ShowS}\\
&\mathit{shows} && = \mathit{showsPrec}\ 0\\
&\mathit{showChar} && :: \mathit{Char} \to \mathit{ShowS}\\
&\mathit{showChar} && = (:)\\
&\mathit{showString} && :: \mathit{String} \to \mathit{ShowS}\\
&\mathit{showString} && = (+\!\!+)\\
&\mathit{showParen} && :: \mathit{Bool} \to \mathit{ShowS} \to \mathit{ShowS}\\
&\mathit{showParen}\ b\ p && \\
&\quad = \mathbf{if}\ b\ \mathbf{then}\ \mathit{showChar}\ \text{'('} \circ p \circ \mathit{showChar}\ \text{')'}\ \mathbf{else}\ p
\end{aligned}
$$

Klass *Show*

Arimteetiliste avaldiste näitamine

```
instance Show Expr where
  showsPrec _ (Num n)      = shows n
  showsPrec _ (Var v)      = showString v
  showsPrec d (e1 :+: e2) = showParen (d > 0)
                              $ showsPrec 0 e1
                              . showString "+"
                              . showsPrec 0 e2
  showsPrec _ (e1 :*: e2) = showsPrec 1 e1
                              . showString "*"
                              . showsPrec 1 e2
```

Tüübiklassid

Klass *Functor*

$$\textbf{class } \mathit{Functor}\ f \textbf{ where}$$
$$\quad \mathit{fmap} :: (a \to b) \to (f\ a \to f\ b)$$

NB!

Muutuja f tähistab unaarset tüübikonstruktorit (mitte tüüpi)!

Kahendpuude funktoriaalus

$$\textbf{data } \mathit{Tree}\ a = \mathit{Empty}$$
$$\qquad\qquad\qquad\qquad \mid \mathit{Branch}\ a\ (\mathit{Tree}\ a)\ (\mathit{Tree}\ a)$$
$$\textbf{instance } \mathit{Functor}\ \mathit{Tree} \textbf{ where}$$
$$\quad \mathit{fmap}\ f\ \mathit{Empty} = \mathit{Empty}$$
$$\quad \mathit{fmap}\ f\ (\mathit{Branch}\ x\ t1\ t2) = \mathit{Branch}\ (f\ x)\ (\mathit{fmap}\ f\ t1)\ (\mathit{fmap}\ f\ t2)$$

Tüübiklassid

Listide funktoriaalus (defineeritud prelüüdis)

```
instance Functor [] where
  fmap = map
```

Maybe funktoriaalus (defineeritud prelüüdis)

```
instance Functor Maybe where
  fmap f Nothing = Nothing
  fmap f (Just x) = Just (f x)
```

Aksioomid

$$\begin{aligned} fmap\ id &= id \\ fmap\ f \circ fmap\ g &= fmap\ (f \circ g) \end{aligned}$$

Tüübiklassid

Klass *Monad*

$$
\begin{aligned}
&\textbf{class } \mathit{Monad}\ m \textbf{ where}\\
&\quad \mathit{return} :: a \to m\ a\\
&\quad (\gg\!=) \quad :: m\ a \to (a \to m\ b) \to m\ b\\
&\quad (\gg) \quad :: m\ a \to m\ b \to m\ b\\
&\quad \mathit{fail} \quad :: \mathit{String} \to m\ a\\
&\quad p \gg q = p \gg\!= \lambda\_ \to q\\
&\quad \mathit{fail}\ s = \mathit{error}\ s
\end{aligned}
$$

Listide monaad (defineeritud prelüüdis)

$$
\begin{aligned}
&\textbf{instance } \mathit{Monad}\ [\,] \textbf{ where}\\
&\quad (x : xs) \gg\!= f = f\ x +\!\!+ (xs \gg\!= f)\\
&\quad [\,] \quad\quad \gg\!= f = [\,]\\
&\quad \mathit{return}\ x \quad\quad = [x]\\
&\quad \mathit{fail}\ s \quad\quad = [\,]
\end{aligned}
$$

Tüübiklassid

Maybe monaad (defineeritud prelüüdis)

$$
\begin{aligned}
&\textbf{instance } \mathit{Monad}\ \mathit{Maybe} \textbf{ where} \\
&\quad \mathit{Just}\ x \gg\!= k = k\ x \\
&\quad \mathit{Nothing} \gg\!= k = \mathit{Nothing} \\
&\quad \mathit{return} = \mathit{Just} \\
&\quad \mathit{fail}\ s = \mathit{Nothing}
\end{aligned}
$$

Aksioomid

$$
\begin{aligned}
\mathit{return}\ x \gg\!= f &= f\ x \\
m \gg\!= \mathit{return} &= m \\
(m \gg\!= \lambda x \to k) \gg\!= f &= m \gg\!= \lambda x \to (k \gg\!= f)
\end{aligned}
$$

Aritmeetilste avaldiste väärtustamine

Aritmeetilste avaldised

$$\begin{aligned}\mathbf{data}\ Expr = {} & Num\ Int \\ & \mid Expr : + : Expr \\ & \mid Expr : - : Expr \\ & \mid Expr : * : Expr \\ & \mid Expr : / : Expr\end{aligned}$$

Näited

$$\begin{aligned}exp1 &= Num\ 1 : + : Num\ 2 \\ exp2 &= Num\ 2 : * : (Num\ 4 : - : Num\ 1) \\ exp3 &= Num\ 4 : * : Num\ 3 : / : Num\ 2\end{aligned}$$

Aritmeetilste avaldiste väärtustamine

Lihtne väärtustaja

$$
\begin{aligned}
&eval :: Expr \to Int \\
&eval\ (Num\ i) &&= i \\
&eval\ (e1 : + : e2) &&= (eval\ e1) + (eval\ e2) \\
&eval\ (e1 : - : e2) &&= (eval\ e1) - (eval\ e2) \\
&eval\ (e1 : * : e2) &&= (eval\ e1) * (eval\ e2) \\
&eval\ (e1 : / : e2) &&= (eval\ e1)\ `div`\ (eval\ e2)
\end{aligned}
$$

Näited

```
Main> eval exp2
6
Main> eval (Num 3 : / : Num 0)
Program error: divide by zero
```

Aritmeetilste avaldiste väärtustamine

Veatöötlusega väärtustaja

$$
\begin{aligned}
&\mathit{eval} :: \mathit{Expr} \to \mathit{Maybe\ Int} \\
&\mathit{eval}\ (\mathit{Num}\ i) \quad\quad = \mathit{Just}\ i \\
&\mathit{eval}\ (e1 : + : e2) = \mathbf{case}\ \mathit{eval}\ e1\ \mathbf{of} \\
&\qquad\qquad\qquad\qquad \mathit{Nothing} \to \mathit{Nothing} \\
&\qquad\qquad\qquad\qquad \mathit{Just}\ v1\ \to \mathbf{case}\ \mathit{eval}\ e2\ \mathbf{of} \\
&\qquad\qquad\qquad\qquad\quad \mathit{Nothing} \to \mathit{Nothing} \\
&\qquad\qquad\qquad\qquad\quad \mathit{Just}\ v2\ \to \mathit{Just}\ (v1 + v2) \\
&\mathit{eval}\ (e1 : - : e2) = \ldots \\
&\mathit{eval}\ (e1 : * : e2) = \ldots
\end{aligned}
$$

Aritmeetilste avaldiste väärtustamine

Veatöötlusega väärtustaja (järg)

$$
\begin{aligned}
\mathit{eval}\ (\mathit{e1} : / : \mathit{e2}) = \mathbf{case}\ \mathit{eval}\ \mathit{e1}\ \mathbf{of}& \\
\mathit{Nothing} &\rightarrow \mathit{Nothing} \\
\mathit{Just}\ \mathit{v1} &\rightarrow \mathbf{case}\ \mathit{eval}\ \mathit{e2}\ \mathbf{of} \\
\mathit{Nothing} &\rightarrow \mathit{Nothing} \\
\mathit{Just}\ \mathit{v2} &\rightarrow \mathbf{if}\ \mathit{v2} \equiv 0 \\
&\quad\ \mathbf{then}\ \mathit{Nothing} \\
&\quad\ \mathbf{else}\ \ \mathit{Just}\ (\mathit{v1}\ `\mathit{div}`\ \mathit{v2})
\end{aligned}
$$

Näited

```
Main> eval exp2
Just 6
Main> eval (Num 3 : / : Num 0)
Nothing
```

Aritmeetilste avaldiste väärtustamine

NB!

- Suur hulk koodi duplitseerimist!
- Mis on ühtne muster mida välja faktoriseerida?
- Paneme tähele, et:
  - toimub alamavaldiste väärtustamiste järjestikustamine;
  - kui üks ebaõnnestub, siis ebaõnnestub ka kogu avaldise väärtustamine;
  - väärtustamise õnnestumisel tehakse resultaat kättesaadavaks järgnevatele väärtustustele;
  - so. täpselt see mida ($\gg=$) operaator teeb!!

Aritmeetilste avaldiste väärtustamine

Veatöötlusega väärtustaja (monaadiline)

```
eval :: Expr → Maybe Int
eval (Num i)     = return i
eval (e1 :+: e2) = eval e1 >>= λv1 →
                   eval e2 >>= λv2 →
                   return (v1 + v2)
eval (e1 :-: e2) = ...
eval (e1 :*: e2) = ...
eval (e1 :/: e2) = eval e1 >>= λv1 →
                   eval e2 >>= λv2 →
                   if v2 ≡ 0
                     then fail "Divide by 0"
                     else return (v1 `div` v2)
```