Uute tüüpide defineerimine

- Tüübisünonüümid:

$$\texttt{type}\ tcon\ tv_1 \ldots tv_n \quad = \quad texpr \qquad\qquad n \geqslant 0$$

- Algebralised andmetüübid:

$$\begin{array}{lcll} \texttt{data}\ tcon\ tv_1 \ldots tv_n & = & dcon_1\ texpr_{11} \ldots texpr_{1m_1} & \\ & | & \ldots & n,m_1,\ldots,m_n \geqslant 0 \\ & | & dcon_n\ texpr_{n1} \ldots texpr_{nm_n} & \end{array}$$

- “Uustüübid”:

$$\texttt{newtype}\ tcon\ tv_1 \ldots tv_n \quad = \quad dcon\ texpr \qquad n \geqslant 0$$

Uute tüüpide defineerimine

Tüübisünonüümid

$$\textbf{type}\ tcon\ tv_1 \ldots tv_n = texpr \qquad n >= 0$$

- Annab tüübiavaldisele nime, mis on esialgese tüübiavaldisega täiesti ekvivalentne
- Tüüp võib olla polümorfne, kuid mitte rekursiivne
- Tüübikonstruktorit ei saa osaliselt rakendada

Näited

$$\begin{aligned}
&\textbf{type}\ String &&= [Char] && \text{-- eeldefineeritud}\\
&\textbf{type}\ AssocList\ a\ b &&= [(a, b)]\\
&\textbf{type}\ Predicate\ a &&= a \to Bool
\end{aligned}$$

Algebralised andmetüübid

Loenditüübid

$$\begin{aligned}\textbf{data}\ Day = {} & Mon \mid Tue \mid Wed \mid Thu \\ & \mid Fri \mid Sat \mid Sun \\ \textbf{deriving}\ & Show\end{aligned}$$

- Nimed Mon – Sun on konstruktor-konstandid (kuna neil ei ole argumente) ja on tüübi Day ainsad elemendid.

Näide

$$\begin{aligned}& valday :: Int \to Day \\ & valday\ n = days\ !!\ (n-1) \\ & \quad \textbf{where}\ days = [Mon, Tue, Wed, Thu, Fri, Sat, Sun]\end{aligned}$$

Algebralised andmetüübid

Funktsioonid defineeritakse näidiste sobitamise abil

$$
\begin{aligned}
&workday :: Day \to Bool \\
&workday\ Mon = True \\
&workday\ Tue\ = True \\
&workday\ Wed = True \\
&workday\ Thu = True \\
&workday\ Fri\ = True \\
&workday\ Sat\ = False \\
&workday\ Sun\ = False
\end{aligned}
$$

Algebralised andmetüübid

Variant-kirjed

$$\mathbf{data}\ \mathit{Tagger} = \mathit{Tagn}\ \mathit{Integer} \mid \mathit{Tagb}\ \mathit{Bool}$$

- Konstruktorid Tagn ja Tagb on funktsioonid

$$\mathit{Tagn} :: \mathit{Integer} \to \mathit{Tagger}$$
$$\mathit{Tagb} :: \mathit{Bool} \to \mathit{Tagger}$$

- Erinevalt (tavalistest) funktsioonidest
  - on konstruktor-aplikatsioonid (näit. $\mathit{Tagn}\ 7$) kanoonilisel kujul (so. neid ei saa edasi lihtsustada);
  - saab konstruktoreid kasutada näidiste sobitamisel

Algebralised andmetüübid

Maybe tüüp (eeldefineeritud)

$$\mathbf{data}\ Maybe\ a = Just\ a \mid Nothing$$

- Konstruktorid *Just* ja *Nothing* on polümorfset tüüpi

$$\begin{aligned} Just &:: a \to Maybe\ a \\ Nothing &:: Maybe\ a \end{aligned}$$

- *Maybe* tüüpi kasutatakse ebaõnnestumiste modelleerimiseks

Algebralised andmetüübid

Ebaõnnestumise modelleerimine

Otsida tabelist võtmele vastav väärtus

$$\begin{aligned}
&\textbf{type } Table = [(String, Int)] \\
&lookup :: String \to Table \to Int \\
&lookup\ key\ ((k, v) : xs) \mid key == k \quad = v \\
&\qquad\qquad\qquad\qquad\quad \mid otherwise = lookup\ key\ xs
\end{aligned}$$

NB!

Kui võtmele vastavat elementi pole, siis on viga!

Algebralised andmetüübid

Ebaõnnestumise modelleerimine

Otsida tabelist võtmele vastav väärtus

$$
\begin{aligned}
&\textbf{type } Table = [(String, Int)] \\
&lookup :: String \to Table \to Maybe\ Int \\
&lookup\ key\ [\,] &&= Nothing \\
&lookup\ key\ ((k, v) : xs) \mid key == k &&= Just\ v \\
&\qquad\qquad\qquad\qquad\ \mid otherwise &&= lookup\ key\ xs
\end{aligned}
$$

Algebralised andmetüübid

Rekursiivsed andemtüübid

$$\begin{aligned}\mathbf{data}\ \mathit{Tree}\ a = {} & \mathit{Empty} \\ & \mid \mathit{Branch}\ a\ (\mathit{Tree}\ a)\ (\mathit{Tree}\ a)\end{aligned}$$

Näide — puu “lamendamine”

$$\begin{aligned}&\mathit{flatten} :: \mathit{Tree}\ a \to [a] \\ &\mathit{flatten}\ \mathit{Empty} = [\,] \\ &\mathit{flatten}\ (\mathit{Branch}\ x\ t1\ t2) = \mathit{flatten}\ t1 +\!\!+ [x] +\!\!+ \\ &\qquad\qquad\qquad\qquad\quad \mathit{flatten}\ t2\end{aligned}$$

Algebralised andmetüübid

Kirjed

Konstruktori argumendid võivad olla "märgendatud"

$$\begin{aligned}\textbf{data}\ \mathit{Tree}\ a = {} & \mathit{Empty} \\ \mid {} & \mathit{Branch}\{\mathit{val} :: a, \mathit{left}, \mathit{right} :: \mathit{Tree}\ a\}\end{aligned}$$

- *val*, *left* ja *right* on selektorfunktsioonid

$$\begin{aligned}\mathit{val} &:: \mathit{Tree}\ a \to a \\ \mathit{left} &:: \mathit{Tree}\ a \to \mathit{Tree}\ a \\ \mathit{right} &:: \mathit{Tree}\ a \to \mathit{Tree}\ a\end{aligned}$$

- Väärtusi võib defineerida tavaliselt (identifitseerides argumendi positsiooni kaudu) või kasutades märgendeid

$$\begin{aligned}\mathit{tree1} &= \mathit{Branch}\ 1\ \mathit{Empty}\ \mathit{Empty} \\ \mathit{tree2} &= \mathit{Branch}\{\mathit{left} = \mathit{Empty} \\ &\qquad\qquad , \mathit{val} = 1 \\ &\qquad\qquad , \mathit{right} = \mathit{Empty}\}\end{aligned}$$

Uute tüüpide defineerimine

”Uustüübid”

- Ühe unaarse konstruktoriga tüüp

$$\textbf{newtype}\ \mathit{Table}\ a\ b = \mathit{MkTable}\ [(a, b)]$$

- Erinevalt tüübisünonüümist võib olla rekursiivne ja saab osaliselt rakendada
- Erinevalt ühe unaarse konstruktoriga andmetüübist on konstruktor “virtuaalne”
- Konstruktori argument võib olla “märgendatud”

Näide: geomeetrilised kujundid

Kujundite esitamine

$$
\begin{aligned}
&\textbf{data}\ \mathit{Shape} = \mathit{Rectangle}\ \mathit{Side}\ \mathit{Side} \\
&\qquad\qquad \mid \mathit{Ellipse}\ \mathit{Radius}\ \mathit{Radius} \\
&\qquad\qquad \mid \mathit{RtTriangle}\ \mathit{Side}\ \mathit{Side} \\
&\qquad\qquad \mid \mathit{Polygon}\ [\mathit{Vertex}] \\
&\qquad\qquad \textbf{deriving}\ \mathit{Show} \\
&\textbf{type}\ \mathit{Radius} = \mathit{Float} \\
&\textbf{type}\ \mathit{Side}\ \ = \mathit{Float} \\
&\textbf{type}\ \mathit{Vertex} = (\mathit{Float}, \mathit{Float})
\end{aligned}
$$

Näide: geomeetrilised kujundid

Uute kujundite defineerimine

$$
\begin{aligned}
square\ s &= Rectangle\ s\ s \\
circle\ r &= Ellipse\ r\ r
\end{aligned}
$$

Kujundite pindala

$$
\begin{aligned}
&area :: Shape \to Float \\
&area\ (Rectangle\ s1\ s2) &&= s1 * s2 \\
&area\ (RtTriangle\ s1\ s2) &&= s1 * s2\ /\ 2 \\
&area\ (Ellipse\ r1\ r2) &&= pi * r1 * r2
\end{aligned}
$$

Näide: geomeetrilised kujundid

Kolmnurga pindala

$$
\begin{aligned}
&triArea :: Vertex \to Vertex \to Vertex \to Float \\
&triArea\ v1\ v2\ v3 = \mathbf{let}\ a = distBetween\ v1\ v2 \\
&\qquad\qquad\qquad\qquad\quad\ b = distBetween\ v2\ v3 \\
&\qquad\qquad\qquad\qquad\quad\ c = distBetween\ v3\ v1 \\
&\qquad\qquad\qquad\qquad\quad\ s = 0.5 * (a + b + c) \\
&\qquad\qquad\qquad\qquad\ \mathbf{in}\ sqrt\ (s * (s - a) * (s - b) * (s - c))
\end{aligned}
$$

Lõigu pikkus

$$
\begin{aligned}
&distBetween :: Vertex \to Vertex \to Float \\
&distBetween\ (x1, y1)\ (x2, y2) \\
&\quad = sqrt\ ((x1 - x2)\ \hat{}\ 2 + (y1 - y2)\ \hat{}\ 2)
\end{aligned}
$$

Näide: geomeetrilised kujundid

Polügoni pindala

$$
\begin{aligned}
&area\ (Polygon\ (v1 : vs)) = polyArea\ vs \\
&\quad \textbf{where}\ polyArea :: [Vertex] \to Float \\
&\qquad polyArea\ (v2 : v3 : vs') = triArea\ v1\ v2\ v3 \\
&\qquad\qquad\qquad\qquad\qquad\quad + polyArea\ (v3 : vs') \\
&\qquad polyArea\ \_\qquad\qquad\qquad\quad = 0
\end{aligned}
$$

Näide: aritmeetilised avaldised

Arimteetiliste avaldiste esitamine

$$\begin{aligned}
&\textbf{type } \mathit{Name} = \mathit{String} \\
&\textbf{data } \mathit{Expr} \;\; = \mathit{Num}\ \mathit{Double} \\
&\qquad\qquad\;\; \mid \mathit{Var}\ \mathit{Name} \\
&\qquad\qquad\;\; \mid \mathit{Expr}\ \mathbin{:+:}\ \mathit{Expr} \\
&\qquad\qquad\;\; \mid \mathit{Expr}\ \mathbin{:*:}\ \mathit{Expr}
\end{aligned}$$

Näited

$$\begin{aligned}
&\mathit{expr1} = \mathit{Num}\ 3 \mathbin{:+:} (\mathit{Num}\ 2 \mathbin{:*:} \mathit{Num}\ 7) \\
&\mathit{expr2} = (\mathit{Var}\ \texttt{"x"} \mathbin{:+:} \mathit{Num}\ 2) \mathbin{:*:} \mathit{Num}\ 4
\end{aligned}$$

Näide: aritmeetilised avaldised

Arimteetiliste avaldiste väärtustamine

$$
\begin{aligned}
&eval :: Expr \to Double \\
&eval\ (Num\ n) \quad\ = n \\
&eval\ (e1\ :+:\ e2) = eval\ e1 + eval\ e2 \\
&eval\ (e1\ :*:\ e2) = eval\ e1\ *\ eval\ e2
\end{aligned}
$$

NB!

Muutujate korral ei ole defineeritud!

Näide: aritmeetilised avaldised

Arimteetiliste avaldiste väärtustamine

$$
\begin{aligned}
&evalE :: Expr \to [(Name, Double)] \to Double \\
&evalE\ (Num\ n)\ env \quad = n \\
&evalE\ (e1\ :+:\ e2)\ env = evalE\ e1\ env + evalE\ e2\ env \\
&evalE\ (e1\ :*:\ e2)\ env = evalE\ e1\ env * evalE\ e2\ env \\
&evalE\ (Var\ x)\ env \quad = \mathbf{case}\ lookup\ x\ env\ \mathbf{of} \\
&\qquad\qquad\qquad\qquad\qquad\quad Just\ v \quad \to v \\
&\qquad\qquad\qquad\qquad\qquad\quad Nothing \to error\ (msg\ x) \\
&\quad \mathbf{where}\ msg\ x = \texttt{"Unbound variable: "} +\!\!+ x
\end{aligned}
$$

Näide: aritmeetilised avaldised

Arimteetiliste avaldiste näitamine

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

NB!

Sulud puudu!

```
Hugs> showExp expr2
"x+2.0*4.0"
```

Näide: aritmeetilised avaldised

Arimteetiliste avaldiste näitamine

```
showExp :: Expr → String
showExp (Num n)    = show n
showExp (Var v)    = v
showExp (e1 :+: e2) = "(" ++ showExp e1 ++ "+" ++
                      showExp e2 ++ ")"
showExp (e1 :*: e2) = "(" ++ showExp e1 ++ "*" ++
                      showExp e2 ++ ")"
```

NB!

Liiga palju sulge!

```
Hugs> showExp expr1
"(3.0+(2.0*7.0))"
```

Näide: aritmeetilised avaldised

Arimteetiliste avaldiste näitamine

```
showExp :: Expr → String
showExp (Num n)    = show n
showExp (Var v)    = v
showExp (e1 :+: e2) = showExp e1 ++ "+" ++
                      showExp e2
showExp (e1 :*: e2) = showArg e1 ++ "*" ++
                      showArg e2
showArg (e1 :+: e2) = "(" ++
                      showExp (e1 :+: e2) ++
                      ")"
showArg e          = showExp e
```