Programación Funcional

Instituto de Computación, Facultad de Ingeniería
Universidad de la República, Uruguay

Tipos Algebraicos

Tipos Algebraicos Polimórficos

- La clase pasada dimos una primera definición...

$$\begin{aligned} \text{data } T = & \; C_1 \; t_{11} \; ... \; t_{1k_1} \\ & \; ... \\ | & \; C_n \; t_{n1} \; ... \; t_{nk_n} \end{aligned}$$

  donde T es el nuevo tipo a introducir y cada C_i es un constructor de dicho tipo, que toma k_i parámetros.

- ... y algunos ejemplos:
  - data $LChar = Nil \mid Cons \; Char \; LChar$
  - data $Tree = Empty \mid Branch \; Int \; Tree \; Tree$

- ¿Pero qué pasa si queremos definir listas o árboles de elementos de un tipo arbitrario?

Tipos Algebraicos Polimórficos

- Las definiciones pueden contener variables de tipos

$$\begin{aligned} \text{data } T\ a_1 \ldots a_m = {} & C_1\ t_{11} \ldots t_{1k_1} \\ & \ldots \\ & \mid\ C_n\ t_{n1} \ldots t_{nk_n} \end{aligned}$$

  donde las variables a_i pueden ser usadas en la definición de los constructores. Ejemplo:

  - data $Pair\ a\ b = Pair\ a\ b$

- Como los tipos algebraicos pueden ser recursivos, el tipo $(T\ a_1 \ldots a_m)$ también puede usarse como componente en la definición. Ejemplos:

  - data $List\ a = Nil \mid Cons\ a\ (List\ a)$
  - data $Tree\ a = Empty \mid Branch\ a\ (Tree\ a)\ (Tree\ a)$

Tipos Algebraicos Polimórficos

- Ejemplos de Instancias:
  - *Cons* `'a'` (*Cons* `'b'` (*Cons* `'c'` *Nil*))
    Tiene tipo *List Char*
  - *Branch* `"tito"` (*Branch* `"toto"` *Empty Empty*) *Empty*
    Tiene tipo *Tree String*
  - *Cons* `'a'` (*Cons True Nil*)
    Incorrecto

Definición de Funciones

- Se pueden definir funciones polimórficas

$$
\begin{aligned}
&\mathit{length} :: \mathit{List}\ a \to \mathit{Int} \\
&\mathit{length}\ \mathit{Nil} &&= 0 \\
&\mathit{length}\ (\mathit{Cons}\ \_\ xs) &&= 1 + \mathit{length}\ xs
\end{aligned}
$$

- Funciones con polimorfismo ad-hoc

$$
\begin{aligned}
&\mathit{maximum} :: \mathit{Ord}\ a \Rightarrow \mathit{List}\ a \to \mathit{Int} \\
&\mathit{maximum}\ (\mathit{Cons}\ x\ \mathit{Nil}) &&= x \\
&\mathit{maximum}\ (\mathit{Cons}\ x\ xs) &&= \mathit{max}\ x\ (\mathit{maximum}\ xs)
\end{aligned}
$$

- Funciones no polimórficas

$$
\begin{aligned}
&\mathit{sumOrd} :: \mathit{List}\ \mathit{Char} \to \mathit{Int} \\
&\mathit{sumOrd}\ \mathit{Nil} &&= 0 \\
&\mathit{sumOrd}\ (\mathit{Cons}\ x\ xs) &&= \mathit{ord}\ x + \mathit{sumOrd}\ xs
\end{aligned}
$$

Preguntas

Manejo de Errores

- Tenemos un tipo algebraico que representa expresiones numéricas

$$\begin{aligned}\text{data } Expr = {} & Lit \quad Int \\ & \mid Add\ Expr\ Expr \mid Sub\ Expr\ Expr \\ & \mid Mul\ Expr\ Expr \mid Div\ Expr\ Expr\end{aligned}$$

- Queremos implementar su evaluador:

$$\begin{aligned}eval\ (Lit\ \ v) &= v \\ eval\ (Add\ e1\ e2) &= eval\ e1 + eval\ e2 \\ eval\ (Sub\ e1\ e2) &= eval\ e1 - eval\ e2 \\ eval\ (Mul\ e1\ e2) &= eval\ e1 * eval\ e2 \\ eval\ (Div\ e1\ e2) &= eval\ e1\ `div`\ eval\ e2\end{aligned}$$

- ¿Qué hacemos en el caso de división por cero?

Manejo de Errores - Opciones

- Función Parcial

$$
\begin{aligned}
&myDiv1 :: Int \to Int \to Int \\
&myDiv1\ x\ y \mid y\ /= 0 = div\ x\ y
\end{aligned}
$$

- Utilizar la función $error :: String \to a$

$$
\begin{aligned}
&myDiv2 :: Int \to Int \to Int \\
&myDiv2\ x\ y \mid y\ /= 0 \quad\quad = div\ x\ y \\
&\qquad\qquad\ \mid otherwise = error\ \texttt{"Division por cero"}
\end{aligned}
$$

- Evitar el error usando un valor "dummy"

$$
\begin{aligned}
&myDiv3 :: Int \to Int \to Int \\
&myDiv3\ x\ y \mid y\ /= 0 \quad\quad = div\ x\ y \\
&\qquad\qquad\ \mid otherwise = 0
\end{aligned}
$$

Manejo de Errores - Opciones

- Que el usuario elija el valor a asignar en caso de error

$$
\begin{aligned}
&myDiv4 :: Int \to Int \to Int \to Int \\
&myDiv4\ x\ y\ z \mid y\ /= 0 \quad = div\ x\ y \\
&\qquad\qquad\quad \mid otherwise = z
\end{aligned}
$$

- Tipo de Error : *Maybe*

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

$$
\begin{aligned}
&myDiv5 :: Int \to Int \to Maybe\ Int \\
&myDiv5\ x\ y \mid y\ /= 0 \quad = Just\ (div\ x\ y) \\
&\qquad\qquad\quad \mid otherwise = Nothing
\end{aligned}
$$

$$
\begin{aligned}
&fromJust \quad :: Maybe\ a \to a \\
&fromMaybe :: a \to Maybe\ a \to a \\
&maybe \quad\ :: b \to (a \to b) \to Maybe\ a \to b
\end{aligned}
$$

Manejo de Errores - Opciones

- Tipo de Error (Suma) : *Either*

$$\text{data } \mathit{Either}\ a\ b = \mathit{Left}\ a \mid \mathit{Right}\ b$$

$$\begin{aligned}
&\mathit{myDiv6} :: \mathit{Int} \to \mathit{Int} \to \mathit{Either\ String\ Int} \\
&\mathit{myDiv6}\ x\ y \mid y\ /= 0 \quad\ = \mathit{Right}\ (\mathit{div}\ x\ y) \\
&\qquad\qquad\ \mid \mathit{otherwise} = \mathit{Left}\ \texttt{"Division por Cero"}
\end{aligned}$$

$$\mathit{either} :: (a \to c) \to (b \to c) \to \mathit{Either}\ a\ b \to c$$

Preguntas