CSc 520
Principles of Programming Languages
18: Haskell — Data Types

Christian Collberg
collberg@cs.arizona.edu

Department of Computer Science
University of Arizona

Copyright © 2005 Christian Collberg
User-defined Datatypes

Haskell allows the definition of new datatypes:

\[
\text{data } \textit{Datatype} \ a_1 \ldots a_n = \textit{constr}_1 \mid \ldots \mid \textit{constr}_m
\]

where

1. \textit{Datatype} is the name of a new type constructor of
   \textit{arity} \( n \geq 0 \),
2. \( a_1, \ldots, a_n \) are distinct type variables representing the
   arguments of \textit{DatatypeName} and
3. \( \textit{constr}_1, \ldots, \textit{constr}_m \) (\( m \geq 1 \)) describe the way in which
   elements of the new datatype are constructed.
User-defined Datatypes...

Each \textit{constr} can take one of two forms:

1. \textit{Name type}_1 \ldots \textit{type}_r where \textit{Name} is a previously unused constructor function name (i.e. an identifier beginning with a capital letter). This declaration introduces \textit{Name} as a new constructor function of type:

\[
\text{type}_1 \rightarrow \ldots \rightarrow \text{type}_r \rightarrow \text{Datatype } a_1 \ldots a_n
\]

2. \textit{type}_1 \oplus \textit{type}_2 where \oplus is a previously unused constructor function operator (i.e. an operator symbol beginning with a colon). This declaration introduces (\oplus) as a new constructor function of type:

\[
\text{type}_1 \rightarrow \text{type}_2 \rightarrow \text{Datatype } a_1 \ldots a_n
\]
User-defined Datatypes...  

The following definition introduces a new type \texttt{Day} with elements \texttt{Sun}, \texttt{Mon}, \texttt{Tue},...:

\begin{verbatim}
data Day = Sun | Mon | Tue | Wed | Thu | Fri | Sat
\end{verbatim}

Simple functions manipulating elements of type \texttt{Day} can be defined using pattern matching:

\begin{verbatim}
what_shall_I_do Sun = "relax"
what_shall_I_do Sat = "go shopping"
what_shall_I_do _ = "go to work"
\end{verbatim}
Another example uses a pair of constructors to provide a representation for temperatures which may be given using either of the centigrade or fahrenheit scales:

```haskell
data Temp = Centigrade Float | Fahrenheit Float

freezing :: Temp -> Bool
freezing (Centigrade temp) = temp <= 0.0
freezing (Fahrenheit temp) = temp <= 32.0
```
Datatype definitions may also be recursive.

The following example defines a type representing binary trees with values of a particular type at their leaves:

```
data Tree a = Lf a  |  Tree a :^: Tree a
```

For example,

```
(Lf 12 :^: (Lf 23 :^: Lf 13)) :^: Lf 10
```

has type `Tree Int` and represents the binary tree:

```
10
  /  \
12   23
    /   /
   13   10
```
User-defined Datatypes...

Calculate the list of elements at the leaves of a tree traversing the branches of the tree from left to right.

leaves :: Tree a -> [a]
leaves (Lf l) = [l]
leaves (l:^:r) = leaves l ++ leaves r

Using the binary tree above as an example:

[12, 23, 13, 10]
(24 reductions, 73 cells)
Acknowledgements

These slides were derived directly from the Gofer manual.

Functional programming environment, Version 2.20
© Copyright Mark P. Jones 1991.

A copy of the Gofer manual can be found in
/home/cs520/2003/gofer/docs/goferdoc.ps.