FEAT: Functional Enumeration of Algebraic Types

Jonas Duregård   Patrik Jansson   Meng Wang

Chalmers University of Technology

Get the Haskell lib.: `cabal install testing-feat`

Example

*Main> $\mathit{index}\ (10^{100}) :: ([[\mathit{Bool}]], [\mathit{Bool}])$
$([[], [], [\mathit{False}, \mathit{False}, \mathit{True}, \mathit{True}], \ldots [\mathit{False}], []],$
$[\mathit{True}, \mathit{True}, \mathit{False}, \mathit{False}])$

Property Based Testing of large AST types

Use case: test pretty-print + parse for Template Haskell.

$$\textbf{data}\ \mathit{Exp} \quad = \mathit{VarE}\ \mathit{Name} \mid \mathit{CaseE}\ \mathit{Exp}\ [\mathit{Match}] \mid \ldots \quad \text{-- 18 Con}$$

Property Based Testing of large AST types

Use case: test pretty-print + parse for Template Haskell.

data *Exp* $\quad = \mathit{VarE}\ \mathit{Name} \mid \mathit{CaseE}\ \mathit{Exp}\ [\mathit{Match}] \mid \ldots$ -- 18 Con

data *Match* $= \mathit{Match}\ \mathit{Pat}\ \mathit{Body}\ [\mathit{Dec}]$

Property Based Testing of large AST types

Use case: test pretty-print + parse for Template Haskell.

$$\textbf{data}\ \mathit{Exp} = \mathit{VarE}\ \mathit{Name} \mid \mathit{CaseE}\ \mathit{Exp}\ [\mathit{Match}] \mid \ldots \quad \text{-- 18 Con}$$

$$\textbf{data}\ \mathit{Match} = \mathit{Match}\ \mathit{Pat}\ \mathit{Body}\ [\mathit{Dec}]$$

$$\textbf{data}\ \mathit{Pat} = \mathit{LitP}\ \mathit{Lit} \mid \mathit{ViewP}\ \mathit{Exp}\ \mathit{Pat} \mid \ldots \quad \text{-- 14 Con}$$

$$\textbf{data}\ \mathit{Body} = \mathit{GuardedB}\ [(\mathit{Guard}, \mathit{Exp})] \mid \mathit{NormalB}\ \mathit{Exp}$$

$$\textbf{data}\ \mathit{Dec} = \mathit{FunD}\ \mathit{Name}\ [\mathit{Clause}] \mid \ldots \quad \text{-- 14 Con}$$

Property Based Testing of large AST types

Use case: test pretty-print + parse for Template Haskell.

$$
\begin{array}{llll}
\textbf{data}\ \mathit{Exp} & = \mathit{VarE}\ \mathit{Name} \mid \mathit{CaseE}\ \mathit{Exp}\ [\mathit{Match}] \mid \ldots & & \text{-- 18 Con} \\
\\
\textbf{data}\ \mathit{Match} & = \mathit{Match}\ \mathit{Pat}\ \mathit{Body}\ [\mathit{Dec}] & & \\
\\
\textbf{data}\ \mathit{Pat} & = \mathit{LitP}\ \mathit{Lit} \mid \mathit{ViewP}\ \mathit{Exp}\ \mathit{Pat} \mid \ldots & & \text{-- 14 Con} \\
\textbf{data}\ \mathit{Body} & = \mathit{GuardedB}\ [(\mathit{Guard}, \mathit{Exp})] \mid \mathit{NormalB}\ \mathit{Exp} & & \\
\textbf{data}\ \mathit{Dec} & = \mathit{FunD}\ \mathit{Name}\ [\mathit{Clause}] \mid \ldots & & \text{-- 14 Con} \\
\\
\textbf{data}\ \mathit{Clause} & = \mathit{Clause}\ [\mathit{Pat}]\ \mathit{Body}\ [\mathit{Dec}] & &
\end{array}
$$

Around 10 datatypes, around 100 constructors.

Enumerating by size

Enumeration of $[\mathit{Bool}]$

$$\{\ [\,], [\mathit{False}], [\mathit{True}], [\mathit{False}, \mathit{False}], [\mathit{False}, \mathit{True}] \ldots$$

$$\begin{array}{l} \mathit{index}\ 0 \mapsto [\,] \\ \mathit{index}\ 1 \mapsto [\mathit{False}] \\ \mathit{index}\ 2 \mapsto [\mathit{True}] \\ \ldots \end{array}$$

Partitioning by size

Partitioning of $[Bool]$:

$$
\begin{array}{ll}
\{ & \\
\quad \{\}, & 0\ \mathit{Constructors} \\
\quad \{\ [\,]\ \}, & 1\ \mathit{Constructor} \\
\quad \{\}, & 2\ \mathit{Constructors} \\
\quad \{\ [\mathit{False}], [\mathit{True}]\ \}, & \ldots \\
\quad \{\}, & \\
\quad \{\ [\mathit{False}, \mathit{False}], [\mathit{False}, \mathit{True}] \ldots & \\
\ldots &
\end{array}
$$

An infinite sequence of finite sequences

Spec. of functional enumerations, part 1

type $E\ a$ -- fun. enum. of values of type a
type $F\ a$ -- similar but for finite enum.

$sel\ :: E\ a \to Part \to F\ a$ -- pick one finite enum.
$card\ :: F\ a \to \mathbb{N}$ -- get its cardinality
$index_F :: F\ a \to Index \to a$ -- ... and its elements
-- where $Part = Index = \mathbb{N}$

Spec. of functional enumerations, part 1

$$
\begin{array}{ll}
\textbf{type}\ E\ a & \text{-- fun. enum. of values of type } a \\
\textbf{type}\ F\ a & \text{-- similar but for finite enum.}
\end{array}
$$

$$
\begin{array}{lll}
sel & :: E\ a \to Part \to F\ a & \text{-- pick one finite enum.} \\
card & :: F\ a \to \mathbb{N} & \text{-- get its cardinality} \\
index_F & :: F\ a \to Index \to a & \text{-- ... and its elements}
\end{array}
$$

$$\text{-- where } Part = Index = \mathbb{N}$$

Possible implementations of F and E:

$$
\begin{array}{ll}
\textbf{type}\ F\ \ a = (\mathbb{N}, Index \to a) & \text{-- we will use this} \\
\textbf{type}\ F'\ a = [a] & \text{-- not this}
\end{array}
$$

Spec. of functional enumerations, part 1

$$
\begin{array}{ll}
\textbf{type}\ E\ a & \text{-- fun. enum. of values of type } a \\
\textbf{type}\ F\ a & \text{-- similar but for finite enum.}
\end{array}
$$

$$
\begin{array}{lll}
sel & :: E\ a \to Part \to F\ a & \text{-- pick one finite enum.} \\
card & :: F\ a \to \mathbb{N} & \text{-- get its cardinality} \\
index_F & :: F\ a \to Index \to a & \text{-- ... and its elements} \\
\end{array}
$$

$$\text{-- where } Part = Index = \mathbb{N}$$

Possible implementations of F and E:

$$
\begin{array}{ll}
\textbf{type}\ F\ \ a = (\mathbb{N}, Index \to a) & \text{-- we will use this} \\
\textbf{type}\ F'\ a = [a] & \text{-- not this} \\
\textbf{type}\ E\ \ a = [F\ a] & \text{-- but we use (roughly) this} \\
\textbf{type}\ E'\ a = Part \to F\ a & \text{-- not this}
\end{array}
$$

Spec. of functional enumerations, part 2

Set operations for E and for F:

$$
\begin{array}{lll}
\mathit{empty} & :: E\ a & \\
\mathit{sing} & :: a \to E\ a & \\
(\oplus) & :: E\ a \to E\ a \to E\ a & \text{-- merge two enum.s} \\
(\otimes) & :: E\ a \to E\ b \to E\ (a \times b) & \text{-- diagonalisation} \\
\mathit{pay} & :: E\ a \to E\ a & \text{-- increase cost (size)} \\
\mathit{fmap} & :: (a \to b) \to E\ a \to E\ b &
\end{array}
$$

$$
\begin{array}{ll}
\mathit{empty}_F & :: F\ a \\
\mathit{sing}_F & :: a \to F\ a \\
(\oplus_F) & :: F\ a \to F\ a \to F\ a \\
(\otimes_F) & :: F\ a \to F\ b \to F\ (a \times b)
\end{array}
$$

Example enumeration of a simple datatype

$$\textbf{data}\ T = L \mid S\ T \mid B\ T\ T$$

$$\begin{aligned}
&\textbf{instance}\ \mathit{Enumerable}\ T\ \textbf{where}\\
&\quad \mathit{enumerate} = \mathit{pay}\ (\mathit{pure}\ L\\
&\qquad\qquad\qquad\qquad \oplus \quad \mathit{pure}\ S \circledast \mathit{enumerate}\\
&\qquad\qquad\qquad\qquad \oplus \quad \mathit{pure}\ B \circledast \mathit{enumerate} \circledast \mathit{enumerate})
\end{aligned}$$

$$\begin{aligned}
&[(0, [\,]),\\
&\ (1, [L]),\\
&\ (1, [S\ L]),\\
&\ (2, [S\ (S\ L), B\ L\ L]),\\
&\ (4, [S\ (S\ (S\ L)), S\ (B\ L\ L), B\ L\ (S\ L), B\ (S\ L)\ L])\\
&\ \ldots
\end{aligned}$$

Spec. of functional enumerations, part 3

$$
\begin{aligned}
&\mathit{sel} :: E\ a \to \mathit{Part} && \to F\ a \\
&\mathit{sel}\ (\mathit{pay}\ e)\ 0 && = \mathit{empty}_F \\
&\mathit{sel}\ (\mathit{pay}\ e)\ (p+1) && = \mathit{sel}\ e\ p \\
&\mathit{sel}\ \mathit{empty}\quad p && = \mathit{empty}_F \\
&\mathit{sel}\ (\mathit{sing}\ x)\ 0 && = \mathit{sing}_F\ x \\
&\mathit{sel}\ (\mathit{sing}\ x)\ (p+1) && = \mathit{empty}_F \\
&\mathit{sel}\ (a \oplus b)\quad p && = \mathit{sel}\ a\ p \oplus_F \mathit{sel}\ b\ p \\
&\mathit{sel}\ (a \otimes b)\quad p && = \bigoplus_{k=0}^{p} (\mathit{sel}\ a\ k \otimes_F \mathit{sel}\ b\ (p-k))
\end{aligned}
$$

Functional finite sequences

$$
\begin{array}{l}
\textbf{data}\ \mathit{Finite}\ a = F \\
\quad \{\, \mathit{card}_F \ :: \mathit{Integer} \\
\quad ,\, \mathit{index}_F :: \mathit{Integer} \to a \,\} \quad \text{-- between } 0 \text{ and } \mathit{card}_F - 1
\end{array}
$$

Functional finite sequences

$$
\begin{aligned}
&\textbf{data}\ \mathit{Finite}\ a = F \\
&\quad \{\, \mathit{card}_F \ \ :: \mathit{Integer} \\
&\quad ,\, \mathit{index}_F :: \mathit{Integer} \to a \,\} \quad \text{-- between } 0 \text{ and } \mathit{card}_F - 1
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{Finite}\ a \approx [a] \\
&\mathit{card}_F \quad\ \approx \mathit{length} \\
&\mathit{index}_F \quad \approx (!!)
\end{aligned}
$$

$$empty_F :: Finite\ a$$
$$empty_F = F\ 0\ (\lambda i \rightarrow \bot)$$

$$sing_F :: a \rightarrow Finite\ a$$
$$sing_F\ a = F\ 1\ one\ \textbf{where}$$
$$\quad one\ 0 = a$$
$$\quad one\ \_ = \bot$$

$$empty_F \approx [\,]$$
$$sing_F \approx (:[\,])$$

$$
\begin{aligned}
&map_F :: (a \rightarrow b) \rightarrow Finite\ a \rightarrow Finite\ b \\
&map_F\ f\ (F\ c\ ix) = F\ c\ (f \circ ix)
\end{aligned}
$$

$$
\begin{aligned}
&(\oplus_F) :: \mathit{Finite}\ a \to \mathit{Finite}\ a \to \mathit{Finite}\ a \\
&f1 \oplus_F f2 = F\ \mathit{car}\ \mathit{ix}\ \textbf{where} \\
&\quad \mathit{car} = \mathit{card}_F\ f1 + \mathit{card}_F\ f2 \\
&\quad \mathit{ix}\ i = \textbf{if}\ i < \mathit{card}_F\ f1 \\
&\qquad\qquad \textbf{then}\ \mathit{index}_F\ f1\ i \\
&\qquad\qquad \textbf{else}\ \ \mathit{index}_F\ f2\ (i - \mathit{card}_F\ f1)
\end{aligned}
$$

$$(\oplus_F) \approx (+\!\!+)$$

$$
\begin{aligned}
(xs +\!\!+ ys)\,!!\,i &\equiv xs\,!!\,i && \text{for } i < \mathit{length}\ xs \\
&\equiv ys\,!!\,(i - \mathit{length}\ xs) && \text{for } i \geqslant \mathit{length}\ xs
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{f1} \otimes_F \mathit{f2} = F\ \mathit{car}\ \mathit{ix}\ \textbf{where} \\
&\quad \mathit{car} = \mathit{card}_F\ \mathit{f1} * \mathit{card}_F\ \mathit{f2} \\
&\quad \mathit{ix}\ i = \textbf{let}\ (d, m) = (i\ `\mathit{divMod}`\ \mathit{card}_F\ \mathit{f2}) \\
&\qquad\qquad \textbf{in}\ \ (\mathit{index}_F\ \mathit{f1}\ d, \mathit{index}_F\ \mathit{f2}\ m)
\end{aligned}
$$

$$
[(x, y) \mid x \leftarrow \mathit{xs}, y \leftarrow \mathit{ys}] \approx \mathit{xs} \otimes_F \mathit{ys}
$$

$$
\begin{aligned}
&[(x, y) \mid x \leftarrow \mathit{xs}, y \leftarrow \mathit{ys}]\, !!\, i \equiv \\
&\quad (\mathit{xs}\, !!\, \mathit{div}\ i\ \mathit{lys}, \mathit{ys}\, !!\, \mathit{mod}\ i\ \mathit{lys}) \\
&\qquad \textbf{where}\ \mathit{lys} = \mathit{length}\ \mathit{ys}
\end{aligned}
$$

Infinite enumerations

$\textbf{newtype}\ \mathit{Enumerate}\ a = E\ \{\mathit{parts} :: [\mathit{Finite}\ a]\}$

-- Actual implementation also store

-- the reversals of all initial segments of the list

$$
\begin{aligned}
&\mathit{empty} :: \mathit{Enumerate}\ a\\
&\mathit{empty} = E\ (\mathit{repeat}\ \mathit{empty}_F)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{sing} :: a \rightarrow \mathit{Enumerate}\ a\\
&\mathit{sing}\ a = E\ ([\mathit{sing}_F\ a] +\!\!+\ \mathit{repeat}\ \mathit{empty}_F)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{fmap} :: (a \rightarrow b) \rightarrow \mathit{Enumerate}\ a \rightarrow \mathit{Enumerate}\ b\\
&\mathit{fmap}\ f = E \circ \mathit{fmap}\ (\mathit{map}_F\ f) \circ \mathit{parts}
\end{aligned}
$$

$$
\begin{aligned}
&(\oplus) :: \mathit{Enumerate}\ a \rightarrow \mathit{Enumerate}\ a \rightarrow \mathit{Enumerate}\ a \\
&(\oplus)\ \mathit{e1}\ \mathit{e2} = E\ \$\ \mathit{zipWith}\ (\oplus_F)\ (\mathit{parts}\ \mathit{e1})\ (\mathit{parts}\ \mathit{e2})
\end{aligned}
$$

Each part of a product is a finite convolution sum:

$$
\begin{aligned}
&(e_1 \otimes e_2) \,!\, n \\
&\quad \equiv \quad\quad e_1 \,!\, 0 \otimes_F e_2 \,!\, n \\
&\quad\quad\quad \oplus_F e_1 \,!\, 1 \otimes_F e_2 \,!\, (n-1) \\
&\quad\quad\quad \oplus_F \ldots \\
&\quad\quad\quad\quad e_1 \,!\, n \otimes_F e_2 \,!\, 0 \\
&\quad \equiv \mathit{concat}_F \\
&\quad\quad (\mathit{zipWith}\ (\otimes_F) \\
&\quad\quad\quad (\mathit{parts}\ e_1) \\
&\quad\quad\quad (\mathit{reverse}\ (\mathit{take}\ n\ (\mathit{parts}\ e_2))))
\end{aligned}
$$

$$
\begin{aligned}
&e \,!\, n = \mathit{parts}\ e \,!!\, n \\
&\mathit{concat}_F = \mathit{foldr}\ (\oplus_F)\ \mathit{empty}_F
\end{aligned}
$$

$$
\begin{aligned}
&(\otimes) :: \mathit{Enumerate}\ a \to \mathit{Enumerate}\ b \to \mathit{Enumerate}\ (a, b)\\
&\mathit{xs} \otimes \mathit{ys} = E\ \$\ \mathit{map}\ (\mathit{conv}\ (\mathit{parts}\ \mathit{xs}))\\
&\qquad\qquad\qquad\qquad (\mathit{reversals}\ (\mathit{parts}\ \mathit{ys}))
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{conv} :: [\mathit{Finite}\ a] \to [\mathit{Finite}\ b] \to \mathit{Finite}\ (a, b)\\
&\mathit{conv}\ \mathit{xs}\ \mathit{ys} = \mathit{concat}_F\ (\mathit{zipWith}\ (\otimes_F)\ \mathit{xs}\ \mathit{ys})
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{reversals} :: [a] \to [[a]]\\
&\mathit{reversals} = \mathit{go}\ [\,]\ \mathbf{where}\\
&\quad \mathit{go}\ \mathit{rev}\ (x : \mathit{xs}) = \mathbf{let}\ \mathit{rev}' = x : \mathit{rev}\\
&\qquad\qquad\qquad\qquad\ \ \mathbf{in}\ \ \mathit{rev}' : \mathit{go}\ \mathit{rev}'\ \mathit{xs}
\end{aligned}
$$

Assigning costs

$$
\begin{aligned}
&\mathit{pay} :: \mathit{Enumerate}\ a \rightarrow \mathit{Enumerate}\ a\\
&\mathit{pay}\ e = E\ \$\ \mathit{empty}_F : \mathit{parts}\ e
\end{aligned}
$$

Assigning costs

$$pay :: Enumerate\ a \rightarrow Enumerate\ a$$
$$pay\ e = E\ \$\ empty_F : parts\ e$$

Guarded recursion operator:

$$fix\ pay \equiv empty$$

Examples

$$enum_{Bool} = pay \, \$ \, sing \; False \oplus sing \; True$$

`*Main>` $map \; card_F \; (parts \; enum_{Bool})$
$[0, 2, 0, 0, 0, 0 \ldots$

Examples

$\mathbf{data}\ N = Ze \mid Su\ N\ \mathbf{deriving}\ Show$
$enum_N = pay \,\$\, sing\ Ze \oplus fmap\ Su\ enum_N$

`*Main>` $map\ card_F\ (parts\ enum_N)$
$[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 \ldots$

Examples

$$
\begin{aligned}
&\mathit{enum}_{[\mathit{Bool}]} = \mathit{pay}\ \$ \\
&\quad \mathit{sing}\ [\,] \oplus \\
&\quad \mathit{fmap}\ (\mathit{uncurry}\ (:))\ (\mathit{enum}_{\mathit{Bool}} \otimes \mathit{enum}_{[\mathit{Bool}]})
\end{aligned}
$$

`*Main>` $\mathit{map}\ \mathit{card}_F\ (\mathit{parts}\ \mathit{enum}_{[\mathit{Bool}]})$

$[0, 1, 0, 2, 0, 4, 0, 8, 0, 16, 0, 32, 0, 64, 0, 128, 0, 256, 0 \ldots$

Absolute indexing

$$
\begin{aligned}
&\mathit{index}' :: \mathit{Enumerate}\ a \to \mathit{Integer} \to a \\
&\mathit{index}' = \mathit{index}_F \circ \mathit{concat}_F \circ \mathit{parts}
\end{aligned}
$$

Absolute indexing

$$index' :: Enumerate\ a \rightarrow Integer \rightarrow a$$
$$index' = index_F \circ concat_F \circ parts$$

Consider:

$$\begin{aligned} length\ (repeat\ ()) &\equiv \bot \\ repeat\ ()\ !!\ i &\equiv () \end{aligned}$$

Random selection

$$uniform_F :: Finite\ a \rightarrow Gen\ a$$
$$uniform_F\ f = liftM\ (index_F\ f)\ (choose\ (0, card_F\ f - 1))$$

Random selection

$$uniform_F :: Finite\ a \rightarrow Gen\ a$$
$$uniform_F\ f = liftM\ (index_F\ f)\ (choose\ (0, card_F\ f - 1))$$

$$uniform :: Enumerate\ a \rightarrow Int \rightarrow Gen\ a$$
$$uniform\ e\ n = uniform_F\ \$\ concat_F\ \$\ take\ n\ \$\ parts\ e$$

- An algebra of Functional Enumerations
    - Concise definition (complete implementation in slides)
    - Nice algebraic properties
    - Automatically derivable for algebraic data types

- An algebra of Functional Enumerations
  - Concise definition (complete implementation in slides)
  - Nice algebraic properties
  - Automatically derivable for algebraic data types
- Actually finds bugs
  - Particularly useful for large abstract syntax tree types
  - Found errors in haskell-src-exts and template-haskell
  - Neither SmallCheck nor QuickCheck finds these bugs

Try it

Have some untested code lying around?

- cabal install testing-feat
- **import** *Test.Feat*

Try it

Have some untested code lying around?

- cabal install testing-feat
- **import** *Test.Feat*

Thank you!