Testing properties of generic functions

Patrik Jansson, Johan Jeuring et al.

Chalmers University of Technology, Sweden

061212 (IFIP WG 2.1)

- Generic Haskell library: 3 bugs fixed
- Generic generators explored

CHALMERS

Patrik Jansson (cs.chalmers.se)

My research:

- Dependent types (Agda, Alfa, cayenne)
- Program Correctness (Fast'n Loose reasoning, Chasing Bottoms, Haskell semantics, . . .)
- Generic Programming (PolyP, Generic Haskell)

"Where type theory meets functional programming"

Research Environment

J. Hughes, T. Coquand, P. Dybjer, K. Claessen

Background

Automatic Testing:

Unit testing, *property testing*, regression testing, contract checking, etc.
Tools: GAST, *QuickCheck*, HUnit, etc.

Generic Programming:

Standard Template Library (C++), Generic Clean, *Generic Haskell*, Scrap your boilerplate, etc.

CHALMERS

Property testing tools

QuickCheck (for Haskell, by Hughes, Claessen)

An embedded language for

- properties (specifications)
- test-data generators (sets)
- coverage measurements (random distribution)

GAST (for Clean, by Koopman et al.)

Generic Automated Software Testing — adds

- generic generators
- complete enumeration

Coverage and Generators: QuickCheck

$$\mathit{prop\_SmallPrime} :: \text{Integer} \rightarrow \text{Property}$$
$$\mathit{prop\_SmallPrime}\ x = \mathit{prime}\ x \Longrightarrow x < 100$$

CHALMERS

Coverage and Generators: QuickCheck

$prop\_SmallPrime :: \text{Integer} \rightarrow \text{Property}$
$prop\_SmallPrime\ x = prime\ x \Longrightarrow x < 100$

```
Main> test prop_SmallPrime
OK, passed 100 successful tests.
```

CHALMERS

Coverage and Generators: QuickCheck

Custom generator:

$$\begin{aligned} &\mathit{primeNumbers} :: \mathsf{Gen\ Integer} \\ &\mathit{primeNumbers} = \mathbf{do}\ n \leftarrow \mathit{arbitrary} \\ &\qquad\qquad\qquad\qquad\quad \mathit{return}\ (\mathit{primes}\ !!\ \mathit{abs}\ n) \end{aligned}$$

$$\begin{aligned} &\mathit{prop\_SmallPrime} :: \mathsf{Integer} \rightarrow \mathsf{Property} \\ &\mathit{prop\_SmallPrime}\ x = \mathit{prime}\ x \Longrightarrow x < 100 \end{aligned}$$

CHALMERS

Coverage and Generators

Custom generator:

$$\begin{aligned}
&\mathit{primeNumbers} :: \mathsf{Gen\ Integer} \\
&\mathit{primeNumbers} = \mathbf{do}\ n \leftarrow \mathit{arbitrary} \\
&\qquad\qquad\qquad\qquad\quad \mathit{return}\ (\mathit{primes}\ !!\ \mathit{abs}\ n)
\end{aligned}$$

$$\begin{aligned}
&\mathit{prop_SmallPrime} :: \mathsf{Integer} \rightarrow \mathsf{Property} \\
&\mathit{prop_SmallPrime}\ x = \mathit{prime}\ x \Longrightarrow x < 100
\end{aligned}$$

$$\begin{aligned}
&\mathit{prop_SmallPrime2} :: \mathsf{Property} \\
&\mathit{prop_SmallPrime2} = \mathit{forAll}\ \mathit{primeNumbers} \\
&\qquad\qquad\qquad\qquad\qquad\quad (\lambda x \rightarrow x < 100)
\end{aligned}$$

CHALMERS

Coverage and Generators

Custom generator:

$$primeNumbers :: \text{Gen Integer}$$
$$primeNumbers = \textbf{do}\ n \leftarrow arbitrary$$
$$\qquad\qquad\qquad\qquad return\ (primes\ !!\ abs\ n)$$

$$prop\_SmallPrime2 :: \text{Property}$$
$$prop\_SmallPrime2 = forAll\ primeNumbers$$
$$\qquad\qquad\qquad\qquad (\lambda x \rightarrow x < 100)$$

```
Main> test prop_SmallPrime2
Falsifiable, after 26 successful tests
107
```

CHALMERS

Coverage and Generators: GAST

GAST syntax is very close to QuickCheck:

$$\begin{aligned} &\mathit{listsAreShort} &&:: [\mathsf{Int}] \to \mathsf{Bool} \\ &\mathit{listsAreShort}\ \mathit{xs} &&= \mathit{length}\ \mathit{xs} < 5 \end{aligned}$$

CHALMERS

Coverage and Generators: GAST

GAST syntax is very close to QuickCheck:

$$\begin{aligned} &\mathit{listsAreShort} &&:: [\mathsf{Int}] \to \mathsf{Bool} \\ &\mathit{listsAreShort}\ \mathit{xs} &&= \mathit{length}\ \mathit{xs} < 5 \end{aligned}$$

and tests can be run by calling the function *test*:

$$\mathit{Start} = \mathit{test}\ \mathit{listsAreShort}$$

Coverage and Generators: GAST

GAST syntax is very close to QuickCheck:

$$\begin{aligned} &\mathit{listsAreShort} &&:: [\mathsf{Int}] \to \mathsf{Bool} \\ &\mathit{listsAreShort}\ \mathit{xs} &&= \mathit{length}\ \mathit{xs} < 5 \end{aligned}$$

and tests can be run by calling the function *test*:

$$\mathit{Start} = \mathit{test}\ \mathit{listsAreShort}$$

which in this case results in the answer

```
Passed after 500 tests.
```

CHALMERS

Combining GAST and QuickCheck

- GAST supplies generators for all datatypes automatically
- Port to Generic Haskell and QuickCheck done
- Both enumeration based and sized generators.
- See paper for details.

CHALMERS

Generic properties

$$
\begin{aligned}
&\text{-- } \mathit{read} \,.\, \mathit{show} = \mathit{id} \\
&\mathit{readshow}\{\!|\mathrm{t} :: *|\!\} \;::\; (\mathit{eq}\{\!|\mathrm{t}|\!\}, \mathit{greadsPrec}\{\!|\mathrm{t}|\!\}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathit{gshowsPrec}\{\!|\mathrm{t}|\!\}) \Rightarrow \\
&\qquad\qquad\qquad\qquad\quad \mathrm{t} \to \mathrm{Bool} \\
&\mathit{readshow}\{\!|\mathrm{t}|\!\}\ x \quad = \mathit{eq}\{\!|\mathrm{t}|\!\}\ x\ (\mathit{gread}\{\!|\mathrm{t}|\!\}\ (\mathit{gshow}\{\!|\mathrm{t}|\!\}\ x))
\end{aligned}
$$

CHALMERS

Generic properties

```
   -- read . show = id
readshow{|t :: *|} :: (eq{|t|}, greadsPrec{|t|},
                                gshowsPrec{|t|}) =>
                      t -> Bool
readshow{|t|} x    = eq{|t|} x (gread{|t|} (gshow{|t|} x))

data Unit    = U
data a :+: b = Inl a | Inr b
data a :*: b = a :*: b
```

CHALMERS

Generic properties

$$readshow\{|t|\}\ x = eq\{|t|\}\ x\ (gread\{|t|\}\ (gshow\{|t|\}\ x))$$

```
data Unit    = U
data a :+: b = Inl a | Inr b
data a :*: b = a :*: b
```

$$
\begin{array}{lllll}
eq\{|\mathsf{Unit}|\} & U & U & = & True \\
eq\{|\alpha :+: \beta|\} & (Inl\ x) & (Inl\ y) & = & eq\{|\alpha|\}\ x\ y \\
eq\{|\alpha :+: \beta|\} & (Inr\ x) & (Inr\ y) & = & eq\{|\beta|\}\ x\ y \\
eq\{|\alpha :+: \beta|\} & \_ & \_ & = & False \\
eq\{|\alpha :*: \beta|\} & (x_1 :*: y_1) & (x_2 :*: y_2) & = & eq\{|\alpha|\}\ x_1\ x_2 \wedge \\
 & & & & \quad eq\{|\beta|\}\ y_1\ y_2 \\
eq\{|\mathsf{Int}|\} & n_1 & n_2 & = & eqInt\ n_1\ n_2
\end{array}
$$

CHALMERS

Generic properties

$$readshow\{|t|\}\ x = eq\{|t|\}\ x\ (gread\{|t|\}\ (gshow\{|t|\}\ x))$$

Testing is monomorphic — which type should be used?

CHALMERS

Structure types combined

```
data StrT a = STUnit
            | STInt Int
            | STChar Char
            | STProd (StrT a) (StrT a)
            | STLabel{anA :: a}
```

CHALMERS

Structure types combined

$$
\begin{aligned}
\textbf{data}\ \mathrm{StrT\ a} = {} & \mathit{STUnit} \\
\mid {} & \mathit{STInt}\ \mathrm{Int} \\
\mid {} & \mathit{STChar}\ \mathrm{Char} \\
\mid {} & \mathit{STProd}\ (\mathrm{StrT\ a})\ (\mathrm{StrT\ a}) \\
\mid {} & \mathit{STLabel}\{\,\mathit{anA} :: \mathrm{a}\,\}
\end{aligned}
$$

$$
\mathit{readshow}\{\!|\mathrm{t}|\!\}\ x = \mathit{eq}\{\!|\mathrm{t}|\!\}\ x\ (\mathit{gread}\{\!|\mathrm{t}|\!\}\ (\mathit{gshow}\{\!|\mathrm{t}|\!\}\ x))
$$

$$
\mathit{readshow\_StrTInt} :: \mathrm{StrT\ Int} \to \mathrm{Bool}
$$

$$
\mathit{readshow\_StrTInt} = \mathit{readshow}\{\!|\mathrm{StrT\ Int}|\!\}
$$

CHALMERS

Structure types combined

$$\begin{aligned}&\textbf{data}\ \mathrm{StrT\ a} = \ \ldots \\ &\qquad\qquad\quad \mid\ \mathit{STLabel}\{\,\mathit{anA} :: \mathrm{a}\,\}\end{aligned}$$

$$\mathit{readshow}\{\!|\mathrm{t}|\!\}\ x = \mathit{eq}\{\!|\mathrm{t}|\!\}\ x\ (\mathit{gread}\{\!|\mathrm{t}|\!\}\ (\mathit{gshow}\{\!|\mathrm{t}|\!\}\ x))$$

$$\mathit{readshow\_StrTInt} :: \mathrm{StrT\ Int} \to \mathrm{Bool}$$

$$\mathit{readshow\_StrTInt} = \mathit{readshow}\{\!|\mathrm{StrT\ Int}|\!\}$$

```
Main> test readshow_STInt
*** Exception: gread: no parse
```

CHALMERS

Structure types combined

$$\textbf{data}\ \mathsf{StrT\ a} = \ldots \mid \mathit{STLabel}\{\,\mathit{anA} :: \mathsf{a}\,\}$$

$$\mathit{readshow}\{\!|\mathsf{t}|\!\}\ x = \mathit{eq}\{\!|\mathsf{t}|\!\}\ x\ (\mathit{gread}\{\!|\mathsf{t}|\!\}\ (\mathit{gshow}\{\!|\mathsf{t}|\!\}\ x))$$

$$\mathit{readshow\_StrTInt} :: \mathsf{StrT\ Int} \to \mathsf{Bool}$$

$$\mathit{readshow\_StrTInt} = \mathit{readshow}\{\!|\mathsf{StrT\ Int}|\!\}$$

```
Main> test (protect readshow_STInt)
Falsifiable, after 3 successful tests
(shrunk failing case 3 times):
STLabel {anA =-2}
```

CHALMERS

Bugs found

Tested parts of the Generic Haskell library corresponding to the derivable Haskell classes *Eq*, *Ord*, *Enum*, *Bounded*, *Read*, and *Show*, and the generic map function *gmap*.

- Show: labeled fields + negative numbers
- Read: to few parantheses allowed
- Enum: unfair merge of infinite lists

Located and fixed.

CHALMERS

QuickCheck and GAST experience

Great tools!
But:

- monomorphic
- need to protect against exceptions, non-termination
- GAST needs “QuickCheck-style” random generation
- GAST needs “QuickCheck-style” shrinking
- QuickCheck needs “GAST-style” generics
- QuickCheck needs “GAST-style” enumeration

CHALMERS

Conclusions

- Generic Haskell library: bugs fixed (Show, Read, Enum)
- QuickCheck and GAST should cross-breed
- Generic Generators implemented for Haskell

Future work:

- Better generic generators
- QuickCheck + ghc generics
- Proofs of Generic Haskell library properties

CHALMERS