Utrecht University

Faculty of Science

Dept. of Information and Computing Sciences

Thesis title

Author

C.R. van der Rest

Supervised by

Dr. W.S. Swierstra

Dr. M.M.T. Chakravarty

Dr. A. Serrano Mena

April 10, 2019

Contents

Declaration

Thanks to family, supervisor, friends and hops!

Abstract

Abstract

1
Introduction

1.1 Introduction

A common way of asserting a program's correctness is by defining properties that should universally hold, and asserting these properties over a range of random inputs. This technique is commonly referred to as *property based testing*, and generally consists of a two-step process. Defining properties that universally hold on all inputs, and defining *generators* that sample random values from the space of possible inputs. *QuickCheck* [?] is likely the most well known tool for performing property based tests on Haskell programs.

Although coming up with a set of properties that properly captures a program's behavior might initially seem to be the most involved part of the process, defining suitable generators for complex input data is actually quite difficult as well. Questions such as how to handle datatypes that are inhabited by an infinite number of values arise, or how to deal with constrained input data. The answers to these questions are reasonably well understood for *Algebraic datatypes (ADT's)*, but no general solution exists when more complex input data is required. In particular, little is known about enumerating and generating inhabitants of *Indexed datatypes*.

1.1.1 Problem Statement

Let us consider an example property in the context of QuickCheck:

$$reverse\_preserves\_length :: [a] \to Bool$$
$$reverse\_preserves\_length\ xs = length\ xs \equiv length\ (reverse\ xs)$$

We can *check* this property by taking a collection of lists, and asserting that *reverse_preserves_length* is *true* on all test inputs. Note that any inhabitant of the type $[a]$ can be used test data for *reverse_preserves_length*. However, a problem occurs when we want to test a conditional property:

$$insert\_preserves\_sorted :: Int \to [Int] \to Property$$
$$insert\_preserves\_sorted\ x\ xs = (sorted\ xs) ==> sorted\ (insert'\ x\ xs)$$

If we invoke QuickCheck on (`quickCheck insert_preserves_sorted`), we get the following output:

```
Test.QuickCheck> quickCheck prop_insertPreservesSorted
*** Gave up! Passed only 70 tests; 1000 discarded tests.
```

In essence, two things go wrong here. The obvious problem is that QuickCheck is unable to generate a sufficient amount of relevant testcases due to the sparseness of the precondition. The second and perhaps more subtle problem is that the generated test data for which the precondition holds almost exclusively consists of small values (that is, lists of 0, 1 or 2 elements). These problems make testing both inefficient (in terms of computational power required), as well as ineffective. Obviously, things will only get worse once we require more complex test data.

Data invariants, such as sortedness, can often be represented as an indexed datatype:

$$
\begin{array}{l}
\mathbf{data}\ \mathit{Sorted}\ \{\ \ell\}:\mathit{List}\ \mathbb{N}\rightarrow \mathit{Set}\ \ell\ \mathbf{where}\\
\quad \mathit{nil}\ :\ \ \mathit{Sorted}\ [\,]\\
\quad \mathit{single}:\ \forall\ \{n:\mathbb{N}\}\rightarrow \mathit{Sorted}\ (n::[\,])\\
\quad \mathit{step}:\ \ \forall\ \{n\ m:\mathbb{N}\}\ \{\mathit{xs}:\mathit{List}\ \mathbb{N}\}\rightarrow n\leq m\\
\qquad\qquad \rightarrow \mathit{Sorted}\ \{\ell\}\ (m::\mathit{xs})\rightarrow \mathit{Sorted}\ \{\ell\}\ (n::m::\mathit{xs})
\end{array}
$$

This means we can generate test data for properties with a precondition by generating values of a suitable indexed datatype. Or in this case, enumerating all indices for which the datatype is inhabited. A good understanding of how to generate inhabitants of indexed datatypes might aid in the generation of many types of complex test data, such as well-typed program terms.

1.1.2 Research Questions and Contributions

The general aim of this thesis is to work towards an answer to the following question:

How can we generically enumerate and/or sample values of indexed datatypes?

Obviously, this is quite a broad question, and as such answering it in its entirety is not realistic. Some subproblems worth considering are:

- We know that enumeration and sampling is possible for regular datatypes. QuickCheck [?] and SmallCheck [?] do this to generically derive test data generators. However, the question remains for which universes of indexed datatypes we can do the same.
- For more complex datatypes (such as ASTs or lambda terms), the number of values grows *extremely* fast with their size: there are only a few lambda terms (up to α-equivalence) with depth 1 or 2, but for depth 50 there are a little under 10^{66} [?] distinct terms. How can we efficiently sample or enumerate larger values of such datatypes? Can we apply techniques such memoization to extend our reach?
- How can insights gained into the enumeration and sampling of indexed datatypes aid in efficient generation of program terms?
- What guarantees about enumeration or sampling can we give? Can we exhaustively enumerate all datatypes, or are there some classes for which this is not possible (if not, why)?

Intended research contributions automatic derivation of generators for at least a subset of indexed datatypes and an implementation in Haskell showing how such derivations can be applied to practical problems.

1.1.3 Methodology

We use the programming language/proof assistant Agda [?] as our vehicle of choice, with the intention to eventually backport our development to Haskell in order to be able to investigate the practical applications of our insights in the context of program term generation.

2
Conclusion

2.1 One Section

Lorem ipsum dolor **sit amet**, *consectetur adipiscing **elit***. Cras est purus, egestas ac quam id, dapibus suscipit tellus. Sed quis tincidunt dolor. Donec efficitur rutrum tincidunt. Praesent convallis commodo nisi, tempus congue justo imperdiet id. Donec congue nulla fermentum ipsum commodo semper. Duis eget imperdiet felis. Aliquam scelerisque efficitur sem nec faucibus. Duis molestie tortor in volutpat facilisis. Cras ligula dolor, aliquet ac turpis nec, aliquam ullamcorper nisl.

$$f(n) = \begin{cases} n/2 & \text{, if } n \text{ is even} \\ 3n+1 & \text{otherwise} \end{cases}$$

Sed libero sapien, consectetur quis magna in, tempor vehicula sapien. Suspendisse dignissim aliquam lobortis. Praesent vel felis ac nulla luctus varius suscipit nec ipsum. Duis purus mauris, lobortis sit amet placerat ut, tincidunt quis felis. Praesent efficitur ex dictum odio ornare vulputate. Sed eu felis eget ante aliquet interdum. Mauris at lobortis nisl.

$$\sum_{x=1}^{j} \sqrt{\frac{x+1}{\mathcal{A}}}$$

Pellentesque diam felis, vulputate quis dui vel, lacinia ultricies justo. Nunc sit amet mattis ligula, suscipit tempor orci. Donec pretium et ipsum at tempor. Pellentesque ac ligula faucibus, auctor risus sed, mollis tortor. Sed accumsan sit amet enim tempor mattis. Aliquam erat volutpat. Etiam vitae varius arcu. Nulla ullamcorper suscipit nisl et placerat. Sed ut venenatis neque. In sapien libero, feugiat id iaculis quis, finibus non ante. Vivamus placerat facilisis massa nec congue. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Sed pulvinar magna ac sodales sollicitudin. Sed auctor mauris feugiat metus imperdiet congue.

2.1.1 With a subsection

Nullam mauris quam, pharetra eu nisl id, facilisis efficitur purus. Morbi sem dui, mollis in augue at, sodales pharetra nisl. Sed posuere lacus justo, ut dignissim eros varius sed. Nam facilisis condimentum turpis. Curabitur vel diam eget justo interdum vestibulum vitae ac turpis. Morbi eget condimentum diam. Etiam efficitur efficitur tincidunt. Maecenas pellentesque, risus quis pellentesque consequat, nisl nulla ornare lectus, nec accumsan nisi quam tincidunt arcu. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce mattis, odio sit amet luctus hendrerit, velit nisl pulvinar turpis, id consectetur nisi massa in enim. Proin sodales pellentesque velit, id egestas nunc porttitor porttitor. Integer vel elit commodo neque vestibulum fringilla. Vivamus turpis urna, pharetra in venenatis at, mattis nec dolor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec nec diam neque. Phasellus id quam vel erat egestas pharetra.

And a para. Nulla ut dui eget dui scelerisque vulputate eget quis tortor. Nullam quis volutpat tortor. Ut euismod tortor at nunc scelerisque aliquet. In id ante sed magna ultrices posuere. Vestibulum ullamcorper mauris ligula, vitae faucibus neque placerat nec. Integer lobortis at lorem eu condimentum. Vestibulum ut hendrerit turpis, eu tincidunt ipsum. Aenean a consectetur risus. Donec dignissim molestie erat eu facilisis. Aliquam facilisis ullamcorper lectus a elementum. Vivamus eget justo quam.

2.1.2 With a subsection

Nullam mauris quam, pharetra eu nisl id, facilisis efficitur purus. Morbi sem dui, mollis in augue at, sodales pharetra nisl. Sed posuere lacus justo, ut dignissim eros varius sed. Nam facilisis condimentum turpis. Curabitur vel diam eget justo interdum vestibulum vitae ac turpis. Morbi eget condimentum diam. Etiam efficitur efficitur tincidunt. Maecenas pellentesque, risus quis pellentesque consequat, nisl nulla ornare lectus, nec accumsan nisi quam tincidunt arcu. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce mattis, odio sit amet luctus hendrerit, velit nisl pulvinar turpis, id consectetur nisi massa in enim. Proin sodales pellentesque velit, id egestas nunc porttitor porttitor. Integer vel elit commodo neque vestibulum fringilla. Vivamus turpis urna, pharetra in venenatis at, mattis nec dolor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec nec diam neque. Phasellus id quam vel erat egestas pharetra.

And a para. Nulla ut dui eget dui scelerisque vulputate eget quis tortor. Nullam quis volutpat tortor. Ut euismod tortor at nunc scelerisque aliquet. In id ante sed magna ultrices posuere. Vestibulum ullamcorper mauris ligula, vitae faucibus neque placerat nec. Integer lobortis at lorem eu condimentum. Vestibulum ut hendrerit turpis, eu tincidunt ipsum. Aenean a consectetur risus. Donec dignissim molestie erat eu facilisis. Aliquam facilisis ullamcorper lectus a elementum. Vivamus eget justo quam.

2.1.3 With a subsection

Nullam mauris quam, pharetra eu nisl id, facilisis efficitur purus. Morbi sem dui, mollis in augue at, sodales pharetra nisl. Sed posuere lacus justo, ut dignissim eros varius sed. Nam facilisis condimentum turpis. Curabitur vel diam eget justo interdum vestibulum vitae ac turpis. Morbi eget condimentum diam. Etiam efficitur efficitur tincidunt. Maecenas pellentesque, risus quis pellentesque consequat, nisl nulla ornare lectus, nec accumsan nisi quam tincidunt arcu. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce mattis, odio sit amet luctus hendrerit, velit nisl pulvinar turpis, id consectetur nisi massa in enim. Proin sodales pellentesque velit, id egestas nunc porttitor porttitor. Integer vel elit commodo neque vestibulum fringilla. Vivamus turpis urna, pharetra in venenatis at, mattis nec dolor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec nec diam neque. Phasellus id quam vel erat egestas pharetra.

And a para. Nulla ut dui eget dui scelerisque vulputate eget quis tortor. Nullam quis volutpat tortor. Ut euismod tortor at nunc scelerisque aliquet. In id ante sed magna ultrices posuere. Vestibulum ullamcorper mauris ligula, vitae faucibus neque placerat nec. Integer lobortis at lorem eu condimentum. Vestibulum ut hendrerit turpis, eu tincidunt ipsum. Aenean a consectetur risus. Donec dignissim molestie erat eu facilisis. Aliquam facilisis ullamcorper lectus a elementum. Vivamus eget justo quam.

2.1.4 With a subsection

Nullam mauris quam, pharetra eu nisl id, facilisis efficitur purus. Morbi sem dui, mollis in augue at, sodales pharetra nisl. Sed posuere lacus justo, ut dignissim eros varius sed. Nam facilisis condimentum turpis. Curabitur vel diam eget justo interdum vestibulum vitae ac turpis. Morbi eget condimentum diam. Etiam efficitur efficitur tincidunt. Maecenas pellentesque, risus quis pellentesque consequat, nisl nulla ornare lectus, nec accumsan nisi quam tincidunt arcu. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce mattis, odio sit amet luctus hendrerit, velit nisl pulvinar turpis, id consectetur nisi massa in enim. Proin sodales pellentesque velit, id egestas nunc porttitor porttitor. Integer vel elit commodo neque vestibulum fringilla. Vivamus turpis urna, pharetra in venenatis at, mattis nec dolor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec nec diam neque. Phasellus id quam vel erat egestas pharetra.

And a para. Nulla ut dui eget dui scelerisque vulputate eget quis tortor. Nullam quis volutpat tortor. Ut euismod tortor at nunc scelerisque aliquet. In id ante sed magna ultrices posuere. Vestibulum ullamcorper mauris ligula, vitae faucibus neque placerat nec. Integer lobortis at lorem eu condimentum. Vestibulum ut hendrerit turpis, eu tincidunt ipsum. Aenean a consectetur risus. Donec dignissim molestie erat eu facilisis. Aliquam facilisis ullamcorper lectus a elementum. Vivamus eget justo quam.

3

By the way ...

3.1 Testing Literate Agda

3.1.1 Describing Well-Typed Λ terms

The following inductive relation can be used to describe all well-typed terms under a certain context, given a goal type:

$$
\begin{array}{l}
\textbf{data}\ \_\vdash\_\ (\mathit{\Gamma} : \mathit{Ctx}) : \mathit{Ty} \to \mathit{Set}\ \textbf{where} \\
\quad [\mathit{Var}] : \ \forall\ \{\tau\} \to \mathit{\Gamma} \ni \tau \\
\quad\qquad\qquad \to \mathit{\Gamma} \vdash \tau \\
\quad [\mathit{Abs}] : \ \forall\ \{\alpha\ \tau\ \sigma\} \to \mathit{\Gamma}, \alpha : \sigma \vdash \tau \\
\quad\qquad\qquad \to \mathit{\Gamma} \vdash \sigma \Rightarrow \tau \\
\quad [\mathit{App}] : \forall\ \{\tau\ \sigma\} \to \mathit{\Gamma} \vdash \sigma \Rightarrow \tau \to \mathit{\Gamma} \vdash \sigma \\
\quad\qquad\qquad \to \mathit{\Gamma} \vdash \tau
\end{array}
$$

A
Some Formulas

Listing of figures

Listing of tables