Deriving Generic Functions by Example (+10 years)

Neil Mitchell

http://ndmitchell.com
Guess the function

Input: aBc(

• (cBa
• bCd)
• ABC(
• aaBBcc((
• ac
Basic idea

\[ f :: A \rightarrow B \]

- I pick: \( a \in A \)
- You pick \( f \), give me \( b \) (where \( b = f \ a \))
- I infer \( f \)
  - Correct for \( a \) (\( b = f \ a \))
  - Correct for all \( A \) (predictable)
Concrete example

Let’s derive ‘is’ functions for Haskell types

\(a\): data MyType = Foo | Bar

\(b\): isFoo Foo{} = True; isFoo _ = False

\hspace{1cm}isBar Bar{} = True; isBar _ = False

You do not need to write down \(f\).
Want to be sure \(f\) is what you wanted.
And the result...

MapCtor (App "FunBind" (List [List [App "Match" (List [App "Ident" (List [Concat (List [String "is", CtorName])]), List [App "PParen" (List [App "PRec" (List [App "UnQual" (List [App "Ident" (List [CtorName])]), List [[]]]), App "Nothing" (List []), App "UnGuardedRhs" (List [App "Con" (List [App "UnQual" (List [App "Ident" (List [String "True"]))]), App "BDecls" (List [List []])]), App "Match" (List [App "Ident" (List [Concat (List [String "is", CtorName])]), List [App "PWildCard" (List []), App "Nothing" (List []), App "UnGuardedRhs" (List [App "Con" (List [App "UnQual" (List [App "Ident" (List [String "False"]))]), App "BDecls" (List [List []])])])])]), App "BDecls" (List [List []])])])])])]))

Important to be *predictable* to treat f as a black box
What happened to this work?
Where this work went

• 2007: York Doctoral Symposium (YDS) paper/talk
• 2008: York Programming Languages and Systems (PLASMA) talk
• 2009: Approaches and Applications of Inductive Programming (AAIP) keynote talk and reviewed post-publication
• 2007-2017: DERIVE open source project
2007: YDS

• My PhD involved learning to write English
  – With much thanks and credit to Colin Runciman
• YDS was a paper I wrote without Colin reading
  – Reading back, it’s not too bad (6 small pages)
• All about an algorithm for inferring ‘f’ for one specific use case
2008: PLASMA

• More theory about how the algorithm worked, a bit more principled
  – f now quantified, can lift between quantifiers

• A sales pitch for the associated open-source DERIVE tool
2009: AAIP

• Invited to give a talk at a workshop
  – They’d seen my YDS work through my blog posts
• More formal and generic – less intuition
• Reviewed post-submission, 12 pages in 2-column style
Formal setup

We pick all of:

• Input  \textit{the input type}
• Output  \textit{the output type}
• DSL  \textit{type of things describing functions}
• sample :: Input \textit{chosen input}
• apply :: DSL \rightarrow Input \rightarrow Output \textit{apply } f
• derive :: Output \rightarrow Maybe DSL \textit{guess } f
Correctness

∀o ∈ Output, 
d ∈ derive o, 
apply d sample ≡ o

If derive succeeds, 
it must work for the example
∀i ∈ Input,
d₁, d₂ ∈ DSL,
apply d₁ sample ≡ apply d₂ sample ⇒
apply d₁ i ≡ apply d₂ i

If any input can distinguish two DSLs it must be sample

Predictability not influenced by derive!
Guess the function solved

Input: aBc(
Output: ac
Function: Pick odd indicies, filter isLower

Option 1: Change sample to aBcd(
Option 2: Only permit one of those in DSL
2007-2017: Derive tool

• Generates instances
  – 60% of instances defined by example
  – Some instances have been moved into GHC

• Moderately successful Haskell tool
  – https://github.com/ndmitchell/derive
  – 843 commits
  – 10 forks, 15 stars, 3 watchers
  – 14 contributors (most a couple of patches)
DERIVE: End of the line

• There are lots of newer instances it can’t do
  – Projects now ship an instance deriver with the instance, rather than centrally

• New way to define generic instances with GHC

• Examples define simple instances, which are the easiest ones anyway

• I don’t personally use it anymore
What happened to me?
Personal life

- 10Y Move to Cambridge
- 8Y Got married (Emily)
- 5Y Had child (Henry)
Jobs

• 3 month Google Summer of Code
• 3 month internship at Credit Suisse
• 8 years at Standard Chartered
• 1 year at Barclays

Expected to have to abandon Haskell, instead been programming it for a decade, and also learnt finance
Academic

• Supercompilation
  – Extension of my PhD, paper in ICFP 2010
  – Had a few PhD students follow my work
  – Mostly fizzled out (apart from Russia)

• Build systems (Shake)
  – Required by Standard Chartered
  – Papers at ICFP 2012, Haskell Symposium 2016
  – Going strong: GHC switching, companies use it
Open Source

• Lots and lots of projects (too many)
  – Biggest: Hoogle, Shake, HLint, Ghcid
  – Recent: Hexml, Weeder, Profiterole
  – Contribute: Foundation, Alga

• All on GitHub [https://github.com/ndmitchell/](https://github.com/ndmitchell/)
Talks/Blog

• Still talk at user groups/conferences
  – 46 talks since 2004, recently 2-4 a year
  – All on http://ndmitchell.com
  – Where I got all the material for this talk from

• Blog with 307 posts
  – I write 4-8 posts a year (should do more)
  – 976,729 views (not including aggregator sites)
  – Initially just writing practice
• Haskell EDSL for writing build systems – alternative to Make
  – Monadic dependencies
  – Unchanging dependencies
  – Non-file dependencies
  – Lots of engineering
• Vastly better for generated files
import Development.Shake
import System.FilePath

main = shakeArgs shakeOptions $ do
  want ["result.tar"]
  ".tar" %> \out -> do
  need [out -<.> "lst"]
  contents <- readFileLines $ out -<.> "lst"
  need contents
  cmd "tar -cf" [out] contents
Shake users

• **Standard Chartered** have been using Shake since 2009, 1000’s of compiles per day.
• **factis research GmbH** use Shake to compile their Checkpad MED application.
• **Samplecount** using Shake since 2012, producing several open-source projects for working with Shake.
• **CovenantEyes** use Shake to build their Windows client.
• **Keystone Tower Systems** has a robotic welder with a Shake build system.
• **FP Complete** use Shake to build Docker images.
• **Genomics Plc** use Shake for the build system, their first major use of Haskell in the company.
Conclusions

• YDS was fun, resulted in my first invited talk

• Suggestions:
  – Do lots of things that interest you
  – Make some of those things good
  – Tell people what you are doing (blogs, talk etc)
  – Be open about your work
  – Start your website/blog now