Transformation and Analysis of Functional Programs

Neil Mitchell
The Thesis

- Uniplate
  - Generic, shorter

- Supero
  - Optimisation, faster

- Catch
  - Checking, safer

- Firstify
  - Defunctionalisation
Uniplate

• Generics library (similar to SYB1)
• Write concise traversals over structures
• Collect all variables in an expression:
  \[ v \mid \text{Var } v \leftarrow \text{universe } x \]
• Make all variables title-case:
  \[
  \text{transform } f \\
  \quad \text{where } f (\text{Var } (x:xs)) = \text{Var } (\text{toUpper } x : xs) \\
  \quad f x = x
  \]
Uniplate advantages

• No/few extensions
• Simple types
• Concise (~40% shorter than SYB)
• Fast (at least 50% faster than SYB)
• Makes use of compiler support

• Does common stuff well
Supero

• A supercompiler
  – Evaluate the program at compile time
  – Resituate so you terminate
• Old ideas, but rarely implemented
  – Supercompilation from Turchin
  – Homeomorphic embedding from Glück
• First attempt for Haskell
Supero improvements

• New contributions
  – A strategy for let bindings
  – A better generalisation

• Competitive with C (microbenchmarks)
• Faster than GHC (small benchmarks)
• Still a prototype, lots of choices to make
Firstify

• Partial defunctionalisation + First-order analysis method = Higher-order analysis method

• Take several ingredients
  – Arity raising, inlining, specialisation
  – Add termination bounds
Firstify results

• Very practically motivated
  – Works well on the nofib suite

• Stress test: \texttt{print (0 :: Double)}
  – Makes use of Arrays, IO Monad, IO Function, Show Continuation, list comprehensions..
Catch

• Automatic safety proof
  – If Catch says “Safe”, your program will not crash by calling error
• First-order language (needs Firstify)
• Division into two parts
  – An algorithm (deals with Core)
  – A constraint language (the lossy bit)
Catch constraints

• Constraints must:
  – Be finite (for a given type)
  – Provide three operations
  – The operations must be consistent

• MP-constraints represent data type patterns in a finite way
Catch results

• Tried on HsColour
  – Real program, real users
  – Even a web service (hpaste.org)
  – Found 3 real bugs, now fixed
  – 1 false positive, but a nice refactoring

• Very automatic, but still powerful
Current status

- Uniplate: already widely used
- Supero: proof of concept
- Firstify: works well enough for Catch
- Catch: useful in some situations