Improving Haskell

Haskell 1.0

Haskell 2.0

Martin Handley and Graham Hutton
University of Nottingham

Improving Haskell Programs

30+ seconds

≥ ?

2 seconds

The Problem

Reasoning about time efficiency in Haskell is notoriously difficult

Efficiency in a lazy setting
$\neq$
Steps taken to evaluate to some normal form

Example

`1 : []`

0 steps to WHNF

`1 :` *`eventually`* `[]`

0 steps to WHNF

`tail (1 : [])`

1 step

`tail (1 :` *`eventually`* `[])`

many steps

A Solution: Improvement Theory
(Moran and Sands, POPL 1999)

- Compare evaluation steps in all contexts:

$$\forall \mathbf{C}.\ \mathtt{steps}\ (\mathbf{C}[\mathtt{M}]) \geq \mathtt{steps}\ (\mathbf{C}[\mathtt{N}])$$

$\mathtt{M} \quad \underset{\sim}{\triangleright} \quad \mathtt{N}$ "M is improved by N"

- A context is a term with a "hole":

$$\mathbf{C} := \lambda x \rightarrow [-] \qquad \mathbf{C}[x] = \lambda x \rightarrow x$$

- Compositional: improve a program "bit by bit"

- Based on a language that is comparable to GHC Core

Renewed Interest!

- Formally shown to be time improvements:
  - Worker/wrapper transformation (ICFP 2014)
  - Common subexpression elimination (PPDP 2015)
  - Short cut fusion (ICFP 2018)

Example: Associativity of Append

```
(xs ++ ys) ++ zs   =   xs ++ (ys ++ zs)
```

- Correctness: simple inductive proof
- What about efficiency?

  The RHS is more efficient because…

  … “the LHS traverses `xs` twice”

Example: Associativity of Append

- Formally reason about efficiency:

$$\texttt{(xs ++ ys) ++ zs} \;\; \trianglerighteq\!\!\!\sim \;\; \texttt{xs ++ (ys ++ zs)}$$

- Prove using *improvement induction:*

$$\texttt{(xs ++ ys) ++ zs} \;\; \trianglerighteq\!\!\!\sim \;\; \checkmark\mathbf{C}\texttt{[(xs ++ ys) ++ zs)]}$$

$$\checkmark\mathbf{C}\texttt{[xs ++ (ys ++ zs)]} \;\; \trianglelefteq\!\trianglerighteq\!\!\!\sim \;\; \texttt{xs ++ (ys ++ zs)}$$

✓ represents a unit time cost

$$(xs \mathbin{+\!\!+} ys) \mathbin{+\!\!+} zs$$

$\equiv$ { syntactic sugar }

$$\mathbf{let}\ ws = xs \mathbin{+\!\!+} ys\ \mathbf{in}\ ws \mathbin{+\!\!+} zs$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { unfold $+\!\!+$ }

$$\begin{array}{l}
\mathbf{let}\ ws = \checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\qquad\qquad [\,] \qquad\ \ \to ys \\
\qquad\qquad (u : us) \to u : (us \mathbin{+\!\!+} ys) \\
\mathbf{in}\ ws \mathbin{+\!\!+} zs
\end{array}$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { unfold $+\!\!+$ }

$$\begin{array}{l}
\mathbf{let}\ ws = \checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\qquad\qquad [\,] \qquad\ \ \to ys \\
\qquad\qquad (u : us) \to u : (us \mathbin{+\!\!+} ys) \\
\mathbf{in}\ \checkmark\mathbf{case}\ ws\ \mathbf{of} \\
\quad [\,] \qquad\ \ \to zs \\
\quad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)
\end{array}$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { move tick inside $\mathbb{D}$'s hole, where

$$\begin{array}{l}
\mathbb{D} \equiv \mathbf{case}\ [-]\ \mathbf{of} \\
\qquad [\,] \qquad\ \ \to ys \\
\qquad (u : us) \to u : (us \mathbin{+\!\!+} ys)\ \}
\end{array}$$

$$\begin{array}{l}
\mathbf{let}\ ws = \mathbf{case}\ \checkmark xs\ \mathbf{of} \\
\qquad\qquad [\,] \qquad\ \ \to ys \\
\qquad\qquad (u : us) \to u : (us \mathbin{+\!\!+} ys) \\
\mathbf{in}\ \checkmark\mathbf{case}\ ws\ \mathbf{of} \\
\quad [\,] \qquad\ \ \to zs \\
\quad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)
\end{array}$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { move $\mathbb{D}$ inside case, where

$$\begin{array}{l}
\mathbb{D} \equiv \mathbf{let}\ ws = [-] \\
\qquad\ \mathbf{in}\ \checkmark\mathbf{case}\ ws\ \mathbf{of} \\
\qquad\qquad [\,] \qquad\ \ \to zs \\
\qquad\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)\ \}
\end{array}$$

$$\begin{array}{l}
\mathbf{case}\ \checkmark xs\ \mathbf{of} \\
\ [\,] \qquad\ \ \to \mathbf{let}\ ws = ys\ \mathbf{in}\ \checkmark\mathbf{case}\ ws\ \mathbf{of} \\
\qquad\qquad\qquad [\,] \qquad\ \ \to zs \\
\qquad\qquad\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs) \\
\ (u : us) \to \mathbf{let}\ ws = u : (us \mathbin{+\!\!+} ys)\ \mathbf{in}\ \checkmark\mathbf{case}\ ws\ \mathbf{of} \\
\qquad\qquad\qquad [\,] \qquad\ \ \to zs \\
\qquad\qquad\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)
\end{array}$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { move tick outside $\mathbb{D}$'s hole, where

$$\begin{array}{l}
\mathbb{D} \equiv \mathbf{case}\ [-]\ \mathbf{of} \\
\qquad [\,] \qquad\ \ \to \ldots \\
\qquad (u : us) \to \ldots\ \}
\end{array}$$

$$\begin{array}{l}
\checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\ [\,] \qquad\ \ \to \mathbf{let}\ ws = ys\ \mathbf{in}\ \checkmark\mathbf{case}\ ws\ \mathbf{of} \\
\qquad\qquad\qquad [\,] \qquad\ \ \to zs \\
\qquad\qquad\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs) \\
\ (u : us) \to \mathbf{let}\ ws = u : (us \mathbin{+\!\!+} ys)\ \mathbf{in}\ \checkmark\mathbf{case}\ ws\ \mathbf{of} \\
\qquad\qquad\qquad [\,] \qquad\ \ \to zs \\
\qquad\qquad\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)
\end{array}$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { fold $+\!\!+$ }

$$\begin{array}{l}
\checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\ [\,] \qquad\ \ \to \mathbf{let}\ ws = ys\ \mathbf{in}\ ws \mathbin{+\!\!+} zs \\
\ (u : us) \to \mathbf{let}\ ws = u : (us \mathbin{+\!\!+} ys)\ \mathbf{in}\ \checkmark\mathbf{case}\ ws\ \mathbf{of} \\
\qquad\qquad\qquad [\,] \qquad\ \ \to zs \\
\qquad\qquad\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)
\end{array}$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { inline ws and remove unused binding }

$$\begin{array}{l}
\checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\ [\,] \qquad\ \ \to \checkmark ys \mathbin{+\!\!+} zs \\
\ (u : us) \to \mathbf{let}\ ws = u : (us \mathbin{+\!\!+} ys)\ \mathbf{in}\ \checkmark\mathbf{case}\ ws\ \mathbf{of} \\
\qquad\qquad\qquad [\,] \qquad\ \ \to zs \\
\qquad\qquad\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)
\end{array}$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { move tick outside $\mathbb{D}$'s hole, where $\mathbb{D} \equiv [-] \mathbin{+\!\!+} zs$ }

$$\begin{array}{l}
\checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\ [\,] \qquad\ \ \to \checkmark(ys \mathbin{+\!\!+} zs) \\
\ (u : us) \to \mathbf{let}\ ws = u : (us \mathbin{+\!\!+} ys)\ \mathbf{in}\ \checkmark\mathbf{case}\ ws\ \mathbf{of} \\
\qquad\qquad\qquad [\,] \qquad\ \ \to zs \\
\qquad\qquad\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)
\end{array}$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { inline ws and remove unused binding }

$$\begin{array}{l}
\checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\ [\,] \qquad\ \ \to \checkmark(ys \mathbin{+\!\!+} zs) \\
\ (u : us) \to \checkmark\mathbf{case}\ \checkmark(u : (us \mathbin{+\!\!+} ys))\ \mathbf{of} \\
\qquad\qquad\qquad [\,] \qquad\ \ \to zs \\
\qquad\qquad\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)
\end{array}$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { move tick outside $\mathbb{D}$'s hole, where

$$\begin{array}{l}
\mathbb{D} \equiv \mathbf{case}\ [-]\ \mathbf{of} \\
\qquad [\,] \qquad\ \ \to zs \\
\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)\ \}
\end{array}$$

$$\begin{array}{l}
\checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\ [\,] \qquad\ \ \to \checkmark(ys \mathbin{+\!\!+} zs) \\
\ (u : us) \to \checkmark\checkmark\mathbf{case}\ u : (us \mathbin{+\!\!+} ys)\ \mathbf{of} \\
\qquad\qquad\qquad [\,] \qquad\ \ \to zs \\
\qquad\qquad\qquad (v : vs) \to v : (vs \mathbin{+\!\!+} zs)
\end{array}$$

$\underset{\sim}{\triangleleft\!\triangleright}$ { evaluate case }

$$\begin{array}{l}
\checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\ [\,] \qquad\ \ \to \checkmark(ys \mathbin{+\!\!+} zs) \\
\ (u : us) \to \checkmark\checkmark(u : ((us \mathbin{+\!\!+} ys) \mathbin{+\!\!+} zs))
\end{array}$$

$\underset{\sim}{\triangleright}$ { remove ticks }

$$\begin{array}{l}
\checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\ [\,] \qquad\ \ \to \checkmark(ys \mathbin{+\!\!+} zs) \\
\ (u : us) \to u : ((us \mathbin{+\!\!+} ys) \mathbin{+\!\!+} zs)
\end{array}$$

$\equiv$ { renaming }

$$\begin{array}{l}
\checkmark\mathbf{case}\ xs\ \mathbf{of} \\
\ [\,] \qquad\ \ \to \checkmark(ys \mathbin{+\!\!+} zs) \\
\ (x : xs) \to x : ((xs \mathbin{+\!\!+} ys) \mathbin{+\!\!+} zs)
\end{array}$$

$\equiv$ { define $\mathbb{C}$, where

$$\begin{array}{l}
\mathbb{C} \equiv \mathbf{case}\ xs\ \mathbf{of} \\
\qquad [\,] \qquad\ \ \to \checkmark(ys \mathbin{+\!\!+} zs) \\
\qquad (x : xs) \to x : [-]\ \}
\end{array}$$

$$\checkmark\mathbb{C}[(xs \mathbin{+\!\!+} ys) \mathbin{+\!\!+} zs]$$

Example: Associativity of Append

- One step of the proof:

$$\boxed{\mathbf{C}[\texttt{case M of \{ }\texttt{pat}_i \rightarrow \texttt{N}_i\texttt{ \}}] \quad \underset{\sim}{\triangleleft\!\triangleright} \quad \texttt{case M of \{ }\texttt{pat}_i \rightarrow \mathbf{C}[\texttt{N}_i]\texttt{ \}}}$$

1. $\texttt{T} = \mathbf{C}[\texttt{case M of \{ }\texttt{pat}_i \rightarrow \texttt{N}_i\texttt{ \}}]$
2. $\mathbf{C}$ is an *evaluation* context
3. $\texttt{FV(M)} \cap \texttt{BV}(\mathbf{C}) = \varnothing$
4. $\texttt{FV}(\mathbf{C}) \cap \texttt{BV}(\texttt{pat}_i) = \varnothing$
5. $\underset{\sim}{\triangleleft\!\triangleright} \quad \Rightarrow \quad \underset{\sim}{\triangleright}$

University of Nottingham Improvement Engine (Unie)

- *Inequational* reasoning assistant written in Haskell, ~12,000 lines of code (available on GitHub)
- Supports mechanised improvement proofs
- Designed to allow users to focus on high-level proof structure by handling technical details
- Inspired by the Hermit system (Farmer 2015), and uses the Kure library (JFP 2014)

Demo

Summary

- First system to support mechanised improvements
- Next: interface with Agda/Coq/Idris so that proofs can be formally verified
- Comments and feedback welcome!

https://github.com/mathandley/unie