Proving Correctness via Free Theorems

The Case of the destroy/build-Rule

Janis Voigtländer

Technische Universität Dresden

PEPM'08

Short Cut Fusion [Gill et al. 1993]

Example:
$$
\begin{aligned}
&\mathrm{fromTo}\ n\ m = \mathrm{go}\ n\\
&\qquad \textbf{where}\ \mathrm{go}\ i = \textbf{if}\ i > m\ \textbf{then}\ []\\
&\qquad\qquad\qquad\qquad\qquad \textbf{else}\ \ i : \mathrm{go}\ (\mathrm{succ}\ i)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{sum}\ []\qquad &= 0\\
\mathrm{sum}\ (x : xs) &= x + \mathrm{sum}\ xs
\end{aligned}
$$

Problem: Expressions like

$$\mathrm{sum}\ (\mathrm{fromTo}\ 1\ 10)$$

involve creating and consuming an intermediate list.

Short Cut Fusion [Gill et al. 1993]

Example:
$$\begin{aligned}&\text{fromTo}\ n\ m = \text{go}\ n\\ &\qquad \textbf{where}\ \text{go}\ i = \textbf{if}\ i > m\ \textbf{then}\ []\\ &\qquad\qquad\qquad\qquad\qquad\quad \textbf{else}\ \ i : \text{go}\ (\text{succ}\ i)\end{aligned}$$

$$\begin{aligned}\text{sum}\ [] &= 0\\ \text{sum}\ (x : xs) &= x + \text{sum}\ xs\end{aligned}$$

Problem: Expressions like

$$\text{sum}\ (\text{fromTo}\ 1\ 10)$$

involve creating and consuming an intermediate list.

Solution:
1. Write fromTo in terms of build.
2. Write sum in terms of foldr.
3. Use the following foldr/build-rule:

$$\text{foldr}\ c\ n\ (\text{build}\ \mathit{prod}) \quad \rightsquigarrow \quad \mathit{prod}\ c\ n$$

The Dual of Short Cut Fusion [Svenningsson 2002]

Example:
$$\begin{aligned}
&\text{fromTo } n\ m = \text{go } n\\
&\qquad \textbf{where } \text{go } i = \textbf{if } i > m \textbf{ then } []\\
&\qquad\qquad\qquad\qquad\quad \textbf{else } \ i : \text{go } (\text{succ } i)
\end{aligned}$$

$$\begin{aligned}
&\text{zip } []\ \ \ \qquad\quad []\qquad\quad\ \ = []\\
&\text{zip } (x : xs)\ (y : ys) = (x, y) : \text{zip } xs\ ys
\end{aligned}$$

Problem: Expressions like

$$\text{zip } (\text{fromTo } 1\ 10)\ (\text{fromTo } '\mathrm{a}'\ '\mathrm{j}')$$

involve two intermediate lists.

The Dual of Short Cut Fusion [Svenningsson 2002]

Example:

$$\begin{array}{l} \text{fromTo } n\ m = \text{go } n \\ \qquad \textbf{where } \text{go } i = \textbf{if } i > m \textbf{ then } [] \\ \qquad\qquad\qquad\qquad\quad \textbf{else } \ i : \text{go } (\text{succ } i) \end{array}$$

$$\begin{array}{lll} \text{zip } [] & [] & = [] \\ \text{zip } (x : xs) & (y : ys) & = (x, y) : \text{zip } xs\ ys \end{array}$$

Problem: Expressions like

$$\text{zip } (\text{fromTo } 1\ 10)\ (\text{fromTo } '\mathrm{a}'\ '\mathrm{j}')$$

involve two intermediate lists.

Solution:

1. Write fromTo in terms of unfoldr.
2. Write zip in terms of destroy.
3. Use the following destroy/unfoldr-rule:

$$\text{destroy } cons\ (\text{unfoldr } psi\ e) \quad \rightsquigarrow \quad cons\ psi\ e$$

Why a destroy/build-Rule?

Example:
$$\begin{aligned}&\text{fromTo } n\ m = \text{go } n\\ &\qquad \textbf{where } \text{go } i = \textbf{if } i > m \textbf{ then } []\\ &\qquad\qquad\qquad\qquad\qquad \textbf{else } \ i : \text{go } (\text{succ } i)\end{aligned}$$

$$\begin{aligned}&\text{zip } [] \quad\quad\ \ [] \quad\quad\ \ = []\\ &\text{zip } (x : xs)\ (y : ys) = (x, y) : \text{zip } xs\ ys\end{aligned}$$

Problem: What if we have

$$\text{zip } (\text{fromTo } 1\ 10)\ (\text{build } \mathit{prod})$$

where the producer of the second intermediate list cannot be expressed in terms of unfoldr?

Why a destroy/build-Rule?

Example: $\text{fromTo}\ n\ m = \text{go}\ n$
$$\begin{aligned}&\textbf{where}\ \text{go}\ i = \textbf{if}\ i > m\ \textbf{then}\ []\\ &\qquad\qquad\qquad\qquad \textbf{else}\ \ i : \text{go}\ (\text{succ}\ i)\end{aligned}$$

$$\begin{aligned}&\text{zip}\ []\ \qquad\quad []\qquad\quad = []\\ &\text{zip}\ (x : xs)\ (y : ys) = (x, y) : \text{zip}\ xs\ ys\end{aligned}$$

Problem: What if we have

$$\text{zip}\ (\text{fromTo}\ 1\ 10)\ (\text{build}\ prod)$$

where the producer of the second intermediate list cannot be expressed in terms of unfoldr?

After fusion:

$$\begin{aligned}&\text{destroy}\ (\lambda psi\ xs \to \text{zipD}\ (\lambda i \to \textbf{if}\ \ i > 10\ \ \cdots)\ psi\ 1\ xs)\\ &\qquad\quad (\text{build}\ prod)\\ &\qquad \textbf{where}\ \text{zipD} = \cdots\end{aligned}$$

A destroy/build-Rule, How?

By the definitions,

$$\text{destroy } \mathit{cons} \text{ (build } \mathit{prod}\text{)}$$

is the same as

$$\mathit{cons} \text{ match } (\mathit{prod} \text{ (:) []})$$

where

$$\textbf{data} \text{ Maybe } \alpha = \text{Nothing} \mid \text{Just } \alpha$$

$$\begin{aligned}
&\text{match} :: [\alpha] \to \text{Maybe } (\alpha, [\alpha]) \\
&\text{match } [] \qquad\quad = \text{Nothing} \\
&\text{match } (x : \mathit{xs}) = \text{Just } (x, \mathit{xs})
\end{aligned}$$

A destroy/build-Rule, How?

By the definitions,

$$\text{destroy}\ \mathit{cons}\ (\text{build}\ \mathit{prod})$$

is the same as

$$\mathit{cons}\ \text{match}\ (\mathit{prod}\ (:)\ [])$$

where

$$\textbf{data}\ \text{Maybe}\ \alpha = \text{Nothing} \mid \text{Just}\ \alpha$$

$$\begin{aligned}
&\text{match} :: [\alpha] \to \text{Maybe}\ (\alpha, [\alpha])\\
&\text{match}\ [] \qquad\quad = \text{Nothing}\\
&\text{match}\ (x : xs) = \text{Just}\ (x, xs)
\end{aligned}$$

Why, then, not simply

$$\text{destroy}\ \mathit{cons}\ (\text{build}\ \mathit{prod})$$
$$\rightsquigarrow$$
$$\mathit{cons}\ \text{id}\ (\mathit{prod}\ (\lambda x\ xs \to \text{Just}\ (x, xs))\ \text{Nothing})\quad ?$$

Does it Preserve Semantics?

All we know about *cons* and *prod* are their types:

$$\mathit{cons} :: \forall\beta.\ (\beta \to \mathsf{Maybe}\ (\mathsf{T}_1, \beta)) \to \beta \to \mathsf{T}_2$$

and

$$\mathit{prod} :: \forall\beta.\ (\mathsf{T}_1 \to \beta \to \beta) \to \beta \to \beta$$

But that might be enough, thanks to free theorems [Wadler 1989]!

In the following, a proof sketch.

Where to Start?

The free theorem for

$$cons :: \forall \beta.\ (\beta \rightarrow \mathrm{Maybe}\ (\mathrm{T}_1, \beta)) \rightarrow \beta \rightarrow \mathrm{T}_2$$

is:

$$\begin{array}{l}
\forall \tau_1, \tau_2, \mathcal{R} \subseteq \tau_1 \times \tau_2, \mathcal{R} \text{ strict, continuous, and bottom-reflecting.} \\
\quad \forall p :: \tau_1 \rightarrow \mathrm{Maybe}\ (\mathrm{T}_1, \tau_1), q :: \tau_2 \rightarrow \mathrm{Maybe}\ (\mathrm{T}_1, \tau_2). \\
\qquad\qquad (p \neq \bot \Leftrightarrow q \neq \bot) \\
\qquad\ \wedge\ (\forall (x, y) \in \mathcal{R}.\ (p\ x, q\ y) \in \mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, \mathcal{R}))) \\
\qquad \Rightarrow \forall (z, v) \in \mathcal{R}.\ cons\ p\ z = cons\ q\ v
\end{array}$$

Where to Start?

The free theorem for

$$cons :: \forall \beta.\ (\beta \rightarrow \mathsf{Maybe}\ (\mathsf{T}_1, \beta)) \rightarrow \beta \rightarrow \mathsf{T}_2$$

is:

$$\begin{array}{l} \forall \tau_1, \tau_2, \mathcal{R} \subseteq \tau_1 \times \tau_2, \mathcal{R}\ \text{strict, continuous, and bottom-reflecting.} \\ \quad \forall p :: \tau_1 \rightarrow \mathsf{Maybe}\ (\mathsf{T}_1, \tau_1), q :: \tau_2 \rightarrow \mathsf{Maybe}\ (\mathsf{T}_1, \tau_2). \\ \qquad\qquad (p \neq \perp \Leftrightarrow q \neq \perp) \\ \qquad\quad \wedge\ (\forall (x, y) \in \mathcal{R}.\ (p\ x, q\ y) \in \mathrm{lift}_{\mathsf{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, \mathcal{R}))) \\ \qquad\quad \Rightarrow \forall (z, v) \in \mathcal{R}.\ cons\ p\ z = cons\ q\ v \end{array}$$

Recall that we want to prove

$$\begin{array}{c} cons\ \mathsf{match}\ (prod\ (:)\ []) \\ = \\ cons\ \mathsf{id}\ (prod\ (\lambda x\ xs \rightarrow \mathsf{Just}\ (x, xs))\ \mathsf{Nothing}) \end{array}$$

Where to Start?

The free theorem for

$$cons :: \forall \beta.\ (\beta \to \text{Maybe}\ (\mathsf{T}_1, \beta)) \to \beta \to \mathsf{T}_2$$

is:

$$\begin{aligned}
&\forall \tau_1, \tau_2, \mathcal{R} \subseteq \tau_1 \times \tau_2, \mathcal{R}\ \text{strict, continuous, and bottom-reflecting}.\\
&\quad \forall p :: \tau_1 \to \text{Maybe}\ (\mathsf{T}_1, \tau_1), q :: \tau_2 \to \text{Maybe}\ (\mathsf{T}_1, \tau_2).\\
&\qquad\quad (p \neq \bot \Leftrightarrow q \neq \bot)\\
&\qquad \wedge\ (\forall (x, y) \in \mathcal{R}.\ (p\ x, q\ y) \in \text{lift}_{\text{Maybe}}(\text{lift}_{(,)}(\text{id}, \mathcal{R})))\\
&\qquad \Rightarrow \forall (z, v) \in \mathcal{R}.\ cons\ p\ z = cons\ q\ v
\end{aligned}$$

Recall that we want to prove

$$\begin{gathered}
cons\ \text{match}\ (prod\ (:)\ [\,])\\
=\\
cons\ \text{id}\ (prod\ (\lambda x\ xs \to \text{Just}\ (x, xs))\ \text{Nothing})
\end{gathered}$$

Where to Start?

The free theorem for

$$cons :: \forall\beta.\ (\beta \rightarrow \mathsf{Maybe}\ (\mathsf{T}_1, \beta)) \rightarrow \beta \rightarrow \mathsf{T}_2$$

is:

$$\begin{array}{l}
\forall \tau_1, \tau_2, \mathcal{R} \subseteq \tau_1 \times \tau_2, \mathcal{R} \text{ strict, continuous, and bottom-reflecting}.\\
\quad \forall p :: \tau_1 \rightarrow \mathsf{Maybe}\ (\mathsf{T}_1, \tau_1), q :: \tau_2 \rightarrow \mathsf{Maybe}\ (\mathsf{T}_1, \tau_2).\\
\qquad\qquad (p \neq \bot \Leftrightarrow q \neq \bot)\\
\qquad\quad \wedge\ (\forall (x, y) \in \mathcal{R}.\ (p\ x, q\ y) \in \mathsf{lift}_{\mathsf{Maybe}}(\mathsf{lift}_{(,)}(\mathsf{id}, \mathcal{R})))\\
\qquad\quad \Rightarrow \forall (z, v) \in \mathcal{R}.\ cons\ p\ z = cons\ q\ v
\end{array}$$

Recall that we want to prove

$$\begin{array}{c}
cons\ \mathsf{match}\ (prod\ (:)\ [])\\
=\\
cons\ \mathsf{id}\ (prod\ (\lambda x\ xs \rightarrow \mathsf{Just}\ (x, xs))\ \mathsf{Nothing})
\end{array}$$

Where to Start?

The free theorem for

$$cons :: \forall\beta.\ (\beta \to \mathsf{Maybe}\ (\mathsf{T}_1, \beta)) \to \beta \to \mathsf{T}_2\ ,$$

specialized down to function level, is:

$$\begin{array}{l}
\forall \tau_1, \tau_2, f :: \tau_1 \to \tau_2, f \text{ strict and total.} \\
\quad \forall p :: \tau_1 \to \mathsf{Maybe}\ (\mathsf{T}_1, \tau_1), q :: \tau_2 \to \mathsf{Maybe}\ (\mathsf{T}_1, \tau_2). \\
\qquad\qquad (p \neq \bot \Leftrightarrow q \neq \bot) \\
\qquad\quad \wedge\ (\forall x :: \tau_1.\ (p\ x, q\ (f\ x)) \in \mathrm{lift}_{\mathsf{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f))) \\
\qquad\quad \Rightarrow \forall y :: \tau_1.\ cons\ p\ y = cons\ q\ (f\ y)
\end{array}$$

Recall that we want to prove

$$\begin{array}{c}
cons\ \mathrm{match}\ (prod\ (:)\ [\,]) \\
= \\
cons\ \mathrm{id}\ (prod\ (\lambda x\ xs \to \mathsf{Just}\ (x, xs))\ \mathsf{Nothing})
\end{array}$$

Where to Start?

The free theorem for

$$cons :: \forall \beta.\ (\beta \to \text{Maybe}\ (\mathsf{T}_1, \beta)) \to \beta \to \mathsf{T}_2\ ,$$

specialized down to function level, is:

$$\begin{aligned}
&\forall \tau_1, \tau_2, f :: \tau_1 \to \tau_2, f \text{ strict and total.}\\
&\quad \forall p :: \tau_1 \to \text{Maybe}\ (\mathsf{T}_1, \tau_1), q :: \tau_2 \to \text{Maybe}\ (\mathsf{T}_1, \tau_2).\\
&\qquad\qquad (p \neq \bot \Leftrightarrow q \neq \bot)\\
&\qquad\quad \wedge\ (\forall x :: \tau_1.\ (p\ x, q\ (f\ x)) \in \text{lift}_{\text{Maybe}}(\text{lift}_{(,)}(\text{id}, f)))\\
&\qquad\quad \Rightarrow \forall y :: \tau_1.\ cons\ p\ y = cons\ q\ (f\ y)
\end{aligned}$$

Recall that we want to prove

$$\begin{gathered}
cons\ \text{match}\ (prod\ (:)\ [])\\
=\\
cons\ \text{id}\ (prod\ (\lambda x\ xs \to \text{Just}\ (x, xs))\ \text{Nothing})
\end{gathered}$$

Where to Start?

The free theorem for

$$cons :: \forall\beta.\ (\beta \to \mathrm{Maybe}\ (\mathrm{T}_1, \beta)) \to \beta \to \mathrm{T}_2\,,$$

specialized down to function level, is:

$$\begin{array}{l}\forall\tau_1,\tau_2, f :: \tau_1 \to \tau_2, f \text{ strict and total.}\\ \quad \forall p :: \tau_1 \to \mathrm{Maybe}\ (\mathrm{T}_1,\tau_1), q :: \tau_2 \to \mathrm{Maybe}\ (\mathrm{T}_1,\tau_2).\\ \qquad\quad (p \neq \bot \Leftrightarrow q \neq \bot)\\ \qquad \wedge\ (\forall x :: \tau_1.\ (p\ x, q\ (f\ x)) \in \mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)))\\ \qquad \Rightarrow \forall y :: \tau_1.\ cons\ p\ y = cons\ q\ (f\ y)\end{array}$$

Recall that we want to prove

$$\begin{array}{c} cons\ \mathrm{match}\ (prod\ (:)\ [])\\ =\\ cons\ \mathrm{id}\ (prod\ (\lambda x\ xs \to \mathrm{Just}\ (x, xs))\ \mathrm{Nothing})\end{array}$$

Where to Start?

The free theorem for

$$\mathit{cons} :: \forall \beta.\ (\beta \to \mathsf{Maybe}\ (\mathsf{T}_1, \beta)) \to \beta \to \mathsf{T}_2\,,$$

specialized down to function level, is:

$$\begin{array}{l} \forall \tau_1, \tau_2, f :: \tau_1 \to \tau_2, f \text{ strict and total.} \\ \quad \forall p :: \tau_1 \to \mathsf{Maybe}\ (\mathsf{T}_1, \tau_1), q :: \tau_2 \to \mathsf{Maybe}\ (\mathsf{T}_1, \tau_2). \\ \qquad\quad (p \neq \bot \Leftrightarrow q \neq \bot) \\ \qquad \wedge\ (\forall x :: \tau_1.\ (p\ x, q\ (f\ x)) \in \mathrm{lift}_{\mathsf{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f))) \\ \qquad \Rightarrow \forall y :: \tau_1.\ \mathit{cons}\ p\ y = \mathit{cons}\ q\ (f\ y) \end{array}$$

Recall that we want to prove

$$\begin{array}{c} \mathit{cons}\ \mathsf{match}\ (\mathit{prod}\ (:)\ [\,]) \\ = \\ \mathit{cons}\ \mathsf{id}\ (\mathit{prod}\ (\lambda x\ xs \to \mathsf{Just}\ (x, xs))\ \mathsf{Nothing}) \end{array}$$

How to Continue?

All we need is a function f such that:

1. f is strict and total
2. $\forall x :: [\mathrm{T}_1].\ (\mathrm{match}\ x, \mathrm{id}\ (f\ x)) \in \mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f))$
3. $f\ (\mathit{prod}\ (:)\ []) = \mathit{prod}\ (\lambda x\ xs \rightarrow \mathrm{Just}\ (x, xs))\ \mathrm{Nothing}$

(Note that the condition $\mathrm{match} \neq \bot \Leftrightarrow \mathrm{id} \neq \bot$ is trivially fulfilled.)

How to Continue?

All we need is a function f such that:

1. f is strict and total
2. $\forall x :: [\mathsf{T}_1].\ (\mathsf{match}\ x, \mathsf{id}\ (f\ x)) \in \mathsf{lift}_{\mathsf{Maybe}}(\mathsf{lift}_{(,)}(\mathsf{id}, f))$
3. $f\ (\mathit{prod}\ (:)\ []) = \mathit{prod}\ (\lambda x\ xs \to \mathsf{Just}\ (x, xs))\ \mathsf{Nothing}$

(Note that the condition $\mathsf{match} \neq \perp \Leftrightarrow \mathsf{id} \neq \perp$ is trivially fulfilled.)

The free theorem for

$$\mathit{prod} :: \forall \beta.\ (\mathsf{T}_1 \to \beta \to \beta) \to \beta \to \beta$$

is:

$$\begin{array}{l}
\forall \tau_1, \tau_2, \mathcal{R} \subseteq \tau_1 \times \tau_2, \mathcal{R} \text{ strict, continuous, and bottom-reflecting.} \\
\quad \forall p :: \mathsf{T}_1 \to \tau_1 \to \tau_1, q :: \mathsf{T}_1 \to \tau_2 \to \tau_2. \\
\qquad\quad (p \neq \perp \Leftrightarrow q \neq \perp) \\
\qquad \wedge\ (\forall x :: \mathsf{T}_1.\ (p\ x \neq \perp \Leftrightarrow q\ x \neq \perp) \\
\qquad\qquad\qquad\qquad \wedge\ \forall (y, z) \in \mathcal{R}.\ (p\ x\ y, q\ x\ z) \in \mathcal{R}) \\
\qquad \Rightarrow \forall (v, w) \in \mathcal{R}.\ (\mathit{prod}\ p\ v, \mathit{prod}\ q\ w) \in \mathcal{R}
\end{array}$$

How to Continue?

All we need is a function f such that:

1. f is strict and total
2. $\forall x :: [\mathrm{T}_1].\ (\mathrm{match}\ x, \mathrm{id}\ (f\ x)) \in \mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f))$
3. $f\ (\mathit{prod}\ (:)\ [\,]) = \mathit{prod}\ (\lambda x\ xs \rightarrow \mathrm{Just}\ (x, xs))\ \mathrm{Nothing}$

(Note that the condition $\mathrm{match} \neq \bot \Leftrightarrow \mathrm{id} \neq \bot$ is trivially fulfilled.)

The free theorem for

$$\mathit{prod} :: \forall \beta.\ (\mathrm{T}_1 \rightarrow \beta \rightarrow \beta) \rightarrow \beta \rightarrow \beta\,,$$

specialized down to function level, is:

$$\begin{array}{l}
\forall \tau_1, \tau_2, f :: \tau_1 \rightarrow \tau_2, f \text{ strict and total.} \\
\quad \forall p :: \mathrm{T}_1 \rightarrow \tau_1 \rightarrow \tau_1, q :: \mathrm{T}_1 \rightarrow \tau_2 \rightarrow \tau_2. \\
\qquad\quad (p \neq \bot \Leftrightarrow q \neq \bot) \\
\qquad \wedge\ (\forall x :: \mathrm{T}_1.\ (p\ x \neq \bot \Leftrightarrow q\ x \neq \bot) \\
\qquad\qquad\qquad\qquad \wedge\ \forall y :: \tau_1.\ f\ (p\ x\ y) = q\ x\ (f\ y)) \\
\qquad \Rightarrow \forall z :: \tau_1.\ f\ (\mathit{prod}\ p\ z) = \mathit{prod}\ q\ (f\ z)
\end{array}$$

How to Continue?

All we need is a function f such that:

1. f is strict and total
2. $\forall x :: [\mathsf{T}_1].\ (\mathsf{match}\ x, \mathsf{id}\ (f\ x)) \in \mathsf{lift}_{\mathsf{Maybe}}(\mathsf{lift}_{(,)}(\mathsf{id}, f))$
3. $f\ (\mathit{prod}\ (:)\ [\,]) = \mathit{prod}\ (\lambda x\ \mathit{xs} \rightarrow \mathsf{Just}\ (x, \mathit{xs}))\ \mathsf{Nothing}$

(Note that the condition $\mathsf{match} \neq \perp \Leftrightarrow \mathsf{id} \neq \perp$ is trivially fulfilled.)

The free theorem for

$$\mathit{prod} :: \forall\beta.\ (\mathsf{T}_1 \rightarrow \beta \rightarrow \beta) \rightarrow \beta \rightarrow \beta\,,$$

specialized down to function level, is:

$$\begin{array}{l}
\forall \tau_1, \tau_2, f :: \tau_1 \rightarrow \tau_2, f \text{ strict and total.} \\
\quad \forall p :: \mathsf{T}_1 \rightarrow \tau_1 \rightarrow \tau_1, q :: \mathsf{T}_1 \rightarrow \tau_2 \rightarrow \tau_2. \\
\qquad\quad (p \neq \perp \Leftrightarrow q \neq \perp) \\
\qquad \wedge\ (\forall x :: \mathsf{T}_1.\ (p\ x \neq \perp \Leftrightarrow q\ x \neq \perp) \\
\qquad\qquad\qquad\quad \wedge\ \forall y :: \tau_1.\ f\ (p\ x\ y) = q\ x\ (f\ y)) \\
\qquad \Rightarrow \forall z :: \tau_1.\ f\ (\mathit{prod}\ p\ z) = \mathit{prod}\ q\ (f\ z)
\end{array}$$

How to Continue?

All we need is a function f such that:

1. f is strict and total
2. $\forall x :: [\mathsf{T}_1].\ (\mathsf{match}\ x, \mathsf{id}\ (f\ x)) \in \mathsf{lift}_{\mathsf{Maybe}}(\mathsf{lift}_{(,)}(\mathsf{id}, f))$
3. $f\ (\mathit{prod}\ (:)\ [\,]) = \mathit{prod}\ (\lambda x\ \mathit{xs} \rightarrow \mathsf{Just}\ (x, \mathit{xs}))\ \mathsf{Nothing}$

(Note that the condition $\mathsf{match} \neq \bot \Leftrightarrow \mathsf{id} \neq \bot$ is trivially fulfilled.)

The free theorem for

$$\mathit{prod} :: \forall \beta.\ (\mathsf{T}_1 \rightarrow \beta \rightarrow \beta) \rightarrow \beta \rightarrow \beta\,,$$

specialized down to function level, is:

$$\begin{array}{l}
\forall \tau_1, \tau_2, f :: \tau_1 \rightarrow \tau_2, f \text{ strict and total.} \\
\quad \forall p :: \mathsf{T}_1 \rightarrow \tau_1 \rightarrow \tau_1, q :: \mathsf{T}_1 \rightarrow \tau_2 \rightarrow \tau_2. \\
\qquad\quad (p \neq \bot \Leftrightarrow q \neq \bot) \\
\qquad \wedge\ (\forall x :: \mathsf{T}_1.\ (p\ x \neq \bot \Leftrightarrow q\ x \neq \bot) \\
\qquad\qquad\qquad\quad \wedge\ \forall y :: \tau_1.\ f\ (p\ x\ y) = q\ x\ (f\ y)) \\
\qquad \Rightarrow \forall z :: \tau_1.\ f\ (\mathit{prod}\ p\ z) = \mathit{prod}\ q\ (f\ z)
\end{array}$$

How to Continue?

All we need is a function f such that:

1. f is strict and total
2. $\forall x :: [\mathsf{T}_1].\ (\mathsf{match}\ x, \mathsf{id}\ (f\ \ x)) \in \mathsf{lift}_{\mathsf{Maybe}}(\mathsf{lift}_{(,)}(\mathsf{id}, f))$
3. $f\ (\mathit{prod}\ (:)\ [\,])=\mathit{prod}\ (\lambda x\ \mathit{xs} \to \mathsf{Just}\ (x, \mathit{xs}))\ \mathsf{Nothing}$

(Note that the condition $\mathsf{match} \neq \bot \Leftrightarrow \mathsf{id} \neq \bot$ is trivially fulfilled.)

The free theorem for

$$\mathit{prod} :: \forall \beta.\ (\mathsf{T}_1 \to \beta \to \beta) \to \beta \to \beta\,,$$

specialized down to function level, is:

$$\begin{array}{l}
\forall \tau_1, \tau_2, f :: \tau_1 \to \tau_2, f \text{ strict and total.} \\
\quad \forall p :: \mathsf{T}_1 \to \tau_1 \to \tau_1, q :: \mathsf{T}_1 \to \tau_2 \to \tau_2. \\
\qquad\quad (p \neq \bot \Leftrightarrow q \neq \bot) \\
\qquad \wedge\ (\forall x :: \mathsf{T}_1.\ (p\ x \neq \bot \Leftrightarrow q\ x \neq \bot) \\
\qquad\qquad\qquad\quad \wedge\ \forall y :: \tau_1.\ f\ (p\ x\ y) = q\ x\ (f\ y)) \\
\qquad \Rightarrow \forall z :: \tau_1.\ f\ (\mathit{prod}\ p\ z) = \mathit{prod}\ q\ (f\ z)
\end{array}$$

Almost There

All we need is a function f such that:

1. f is strict and total
2. $\forall x :: [\mathsf{T}_1].\ (\mathsf{match}\ x, \mathsf{id}\ (f\ x)) \in \mathsf{lift}_{\mathsf{Maybe}}(\mathsf{lift}_{(,)}(\mathsf{id}, f))$
3. $\forall x :: \mathsf{T}_1, y :: [\mathsf{T}_1].\ f\ ((:)\ x\ y) = (\lambda x\ xs \rightarrow \mathsf{Just}\ (x, xs))\ x\ (f\ y)$
4. $f\ [] = \mathsf{Nothing}$

(Note that the "$\neq \bot$"-conditions are again trivially fulfilled.)

Almost There

All we need is a function f such that:

1. f is strict and total
2. $\forall x :: [\mathsf{T}_1].\ (\mathsf{match}\ x, \mathsf{id}\ (f\ \ x)) \in \mathsf{lift}_{\mathsf{Maybe}}(\mathsf{lift}_{(,)}(\mathsf{id}, f))$
3. $\forall x :: \mathsf{T}_1, y :: [\mathsf{T}_1].\ f\ ((:)\ x\ y) = (\lambda x\ xs \rightarrow \mathsf{Just}\ (x, xs))\ x\ (f\ \ y)$
4. $f\ [] = \mathsf{Nothing}$

(Note that the "$\neq \perp$"-conditions are again trivially fulfilled.)

The last two conditions leave no room other than to consider:

$$
\begin{aligned}
f\ [] \quad\quad &= \mathsf{Nothing} \\
f\ (x : y) &= \mathsf{Just}\ (x, f\ \ y)
\end{aligned}
$$

Almost There

All we need is a function f such that:

1. f is strict and total
2. $\forall x :: [\mathrm{T}_1].\ (\mathrm{match}\ x, \mathrm{id}\ (f\ x)) \in \mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f))$
3. $\forall x :: \mathrm{T}_1, y :: [\mathrm{T}_1].\ f\ ((:)\ x\ y) = (\lambda x\ xs \rightarrow \mathrm{Just}\ (x, xs))\ x\ (f\ y)$
4. $f\ [] = \mathrm{Nothing}$

(Note that the "$\neq \perp$"-conditions are again trivially fulfilled.)

The last two conditions leave no room other than to consider:

$$\begin{aligned} f\ [] &= \mathrm{Nothing} \\ f\ (x : y) &= \mathrm{Just}\ (x, f\ y) \end{aligned}$$

This f is strict and total!

Almost There

2. $\forall x :: [\mathsf{T}_1].\ (\mathsf{match}\ x, \mathsf{id}\ (f\ \ x)) \in \mathsf{lift}_{\mathsf{Maybe}}(\mathsf{lift}_{(,)}(\mathsf{id}, f))$?

$$\begin{aligned} f\ \ [] \qquad\quad &= \mathsf{Nothing} \\ f\ \ (x : y) &= \mathsf{Just}\ (x, f\ \ y) \end{aligned}$$

Finishing Up

We have:

$$\mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)) = \{(\bot, \bot), (\mathrm{Nothing}, \mathrm{Nothing})\} \cup \{(\mathrm{Just}\ x_1, \mathrm{Just}\ y_1) \mid (x_1, y_1) \in \mathrm{lift}_{(,)}(\mathrm{id}, f)\}$$

$$\mathrm{lift}_{(,)}(\mathrm{id}, f) = \{(\bot, \bot)\} \ \cup \ \{((x_1, x_2), (y_1, y_2)) \mid x_1 = y_1 \wedge f\ x_2 = y_2\}$$

Finishing Up

We have:

$$\mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)) = \{(\bot, \bot), (\mathrm{Nothing}, \mathrm{Nothing})\} \cup \{(\mathrm{Just}\ x_1, \mathrm{Just}\ y_1) \mid (x_1, y_1) \in \mathrm{lift}_{(,)}(\mathrm{id}, f)\}$$

$$\mathrm{lift}_{(,)}(\mathrm{id}, f) = \{(\bot, \bot)\} \ \cup\ \{((x_1, x_2), (y_1, y_2)) \mid x_1 = y_1 \wedge f\ x_2 = y_2\}$$

To establish

$$\forall x :: [\mathrm{T}_1].\ (\mathrm{match}\ x, \mathrm{id}\ (f\ x)) \in \mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f))\,,$$

we check against the definitions:

$$\begin{array}{ll} \mathrm{match}\ [\,] = \mathrm{Nothing} & f\ [\,] = \mathrm{Nothing} \\ \mathrm{match}\ (x : y) = \mathrm{Just}\ (x, y) & f\ (x : y) = \mathrm{Just}\ (x, f\ y) \end{array}$$

Finishing Up

We have:

$$\begin{aligned}\mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)) = {} & \{(\bot, \bot), (\mathrm{Nothing}, \mathrm{Nothing})\} \cup \\ & \{(\mathrm{Just}\ x_1, \mathrm{Just}\ y_1) \mid (x_1, y_1) \in \mathrm{lift}_{(,)}(\mathrm{id}, f)\}\end{aligned}$$

$$\mathrm{lift}_{(,)}(\mathrm{id}, f) = \{(\bot, \bot)\} \ \cup \ \{((x_1, x_2), (y_1, y_2)) \mid x_1 = y_1 \wedge f\ x_2 = y_2\}$$

To establish

$$\forall x :: [\mathrm{T}_1].\ (\mathrm{match}\ x, \mathrm{id}\ (f\ x)) \in \mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)),$$

we check against the definitions:

$$\begin{aligned}&\mathrm{match}\ [] &&= \mathrm{Nothing} \qquad\qquad && f\ [] &&= \mathrm{Nothing}\\ &\mathrm{match}\ (x : y) &&= \mathrm{Just}\ (x, y) && f\ (x : y) &&= \mathrm{Just}\ (x, f\ y)\end{aligned}$$

Finishing Up

We have:

$$\mathrm{lift}_{\mathsf{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)) = \{(\bot,\bot), (\mathrm{Nothing}, \mathrm{Nothing})\} \cup \{(\mathrm{Just}\ x_1, \mathrm{Just}\ y_1) \mid (x_1, y_1) \in \mathrm{lift}_{(,)}(\mathrm{id}, f)\}$$

$$\mathrm{lift}_{(,)}(\mathrm{id}, f) = \{(\bot,\bot)\} \ \cup\ \{((x_1, x_2), (y_1, y_2)) \mid x_1 = y_1 \wedge f\ x_2 = y_2\}$$

To establish

$$\forall x :: [\mathrm{T}_1].\ (\mathrm{match}\ x, \mathrm{id}\ (f\ x)) \in \mathrm{lift}_{\mathsf{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f))\,,$$

we check against the definitions:

$$\begin{aligned} \mathrm{match}\ [\,] &= \mathrm{Nothing} & \qquad f\ [\,] &= \mathrm{Nothing} \\ \mathrm{match}\ (x : y) &= \mathrm{Just}\ (x, y) & \qquad f\ (x : y) &= \mathrm{Just}\ (x, f\ y) \end{aligned}$$

Finishing Up

We have:

$$\begin{aligned}\mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)) = {} & \{(\bot, \bot), (\mathrm{Nothing}, \mathrm{Nothing})\} \cup \\ & \{(\mathrm{Just}\ x_1, \mathrm{Just}\ y_1) \mid (x_1, y_1) \in \mathrm{lift}_{(,)}(\mathrm{id}, f)\}\end{aligned}$$

$$\mathrm{lift}_{(,)}(\mathrm{id}, f) = \{(\bot, \bot)\} \ \cup \ \{((x_1, x_2), (y_1, y_2)) \mid x_1 = y_1 \wedge f\ \ x_2 = y_2\}$$

To establish

$$\forall x :: [\mathrm{T}_1].\ (\mathrm{match}\ x, \mathrm{id}\ (f\ \ x)) \in \mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)),$$

we check against the definitions:

$$\begin{aligned} \mathrm{match}\ [] &= \mathrm{Nothing} & \qquad f\ \ [] &= \mathrm{Nothing} \\ \mathrm{match}\ (x : y) &= \mathrm{Just}\ (x, y) & \qquad f\ \ (x : y) &= \mathrm{Just}\ (x, f\ \ y) \end{aligned}$$

Finishing Up

We have:

$$\mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)) = \{(\bot, \bot), (\mathrm{Nothing}, \mathrm{Nothing})\} \cup \{(\mathrm{Just}\ x_1, \mathrm{Just}\ y_1) \mid (x_1, y_1) \in \mathrm{lift}_{(,)}(\mathrm{id}, f)\}$$

$$\mathrm{lift}_{(,)}(\mathrm{id}, f) = \{(\bot, \bot)\} \ \cup\ \{((x_1, x_2), (y_1, y_2)) \mid x_1 = y_1 \wedge f\ \ x_2 = y_2\}$$

To establish

$$\forall x :: [\mathrm{T}_1].\ (\mathrm{match}\ x, \mathrm{id}\ (f\ \ x)) \in \mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f))\,,$$

we check against the definitions:

$$\begin{aligned} &\mathrm{match}\ [\,] &&= \mathrm{Nothing} \qquad\qquad && f\ \ [\,] &&= \mathrm{Nothing} \\ &\mathrm{match}\ (x : y) &&= \mathrm{Just}\ (x, y) && f\ \ (x : y) &&= \mathrm{Just}\ (x, f\ \ y) \end{aligned}$$

Finishing Up

We have:

$$\begin{aligned}\mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)) = {} & \{(\bot, \bot), (\mathrm{Nothing}, \mathrm{Nothing})\} \cup \\ & \{(\mathrm{Just}\ x_1, \mathrm{Just}\ y_1) \mid (x_1, y_1) \in \mathrm{lift}_{(,)}(\mathrm{id}, f)\}\end{aligned}$$

$$\mathrm{lift}_{(,)}(\mathrm{id}, f) = \{(\bot, \bot)\} \ \cup \ \{((x_1, x_2), (y_1, y_2)) \mid x_1 = y_1 \wedge f\ x_2 = y_2\}$$

To establish

$$\forall x :: [\mathrm{T}_1].\ (\mathrm{match}\ x, \mathrm{id}\ (f\ x)) \in \mathrm{lift}_{\mathrm{Maybe}}(\mathrm{lift}_{(,)}(\mathrm{id}, f)),$$

we check against the definitions:

$$\begin{aligned}\mathrm{match}\ [] &= \mathrm{Nothing} & \qquad f\ [] &= \mathrm{Nothing} \\ \mathrm{match}\ (x : y) &= \mathrm{Just}\ (x, y) & \qquad f\ (x : y) &= \mathrm{Just}\ (x, f\ y)\end{aligned}$$

Done!

Conclusion

- The destroy/build-rule holds unconditionally.
- Part of the proof work was push-the-button.
- The remainder was very much goal-driven.
- The approach scales to other transformation rules as well.
- Sascha Böhme implemented a great tool!
- Go play with it:
  `http://linux.tcs.inf.tu-dresden.de/~voigt/ft/`

References

- A. Gill, J. Launchbury, and S.L. Peyton Jones.
  A short cut to deforestation.
  In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 223–232. ACM Press, 1993.

- J. Svenningsson.
  Shortcut fusion for accumulating parameters & zip-like functions.
  In *International Conference on Functional Programming, Proceedings*, pages 124–132. ACM Press, 2002.

- P. Wadler.
  Theorems for free!
  In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 347–359. ACM Press, 1989.