Type-Based Reasoning about Efficiency

D. Seidel* J. Voigtländer

University of Bonn

August 3rd, 2011

*Supported by the DFG under grant VO 1512/1-1.

Parametric Polymorphism in Haskell

A standard function:

$$
\begin{aligned}
\texttt{map}\ g\ [\,] &= [\,] \\
\texttt{map}\ g\ (a : as) &= (g\ a) : (\texttt{map}\ g\ as)
\end{aligned}
$$

Parametric Polymorphism in Haskell

A standard function:

$$\begin{aligned} \texttt{map}\ g\ [\,] &= [\,] \\ \texttt{map}\ g\ (a : as) &= (g\ a) : (\texttt{map}\ g\ as) \end{aligned}$$

Some invocations:

$$\texttt{map succ}\ [1, 2, 3] \qquad = [2, 3, 4]$$

Parametric Polymorphism in Haskell

A standard function:

$$\begin{aligned} \texttt{map}\ g\ [\,] &= [\,] \\ \texttt{map}\ g\ (a : as) &= (g\ a) : (\texttt{map}\ g\ as) \end{aligned}$$

Some invocations:

$$\begin{aligned} \texttt{map succ}\ [1, 2, 3] &= [2, 3, 4] \\ \texttt{map not}\ \ [\mathsf{True}, \mathsf{False}] &= [\mathsf{False}, \mathsf{True}] \end{aligned}$$

Parametric Polymorphism in Haskell

A standard function:

$$\begin{aligned} \texttt{map}\ g\ [\,] &= [\,] \\ \texttt{map}\ g\ (a : as) &= (g\ a) : (\texttt{map}\ g\ as) \end{aligned}$$

Some invocations:

$$\begin{aligned} \texttt{map succ}\ [1,2,3] &= [2,3,4] \\ \texttt{map not}\ \ [\text{True}, \text{False}] &= [\text{False}, \text{True}] \\ \texttt{map even}\ [1,2,3] &= [\text{False}, \text{True}, \text{False}] \end{aligned}$$

Parametric Polymorphism in Haskell

A standard function:

$$\begin{aligned} \texttt{map}\ g\ [\,] &= [\,] \\ \texttt{map}\ g\ (a : as) &= (g\ a) : (\texttt{map}\ g\ as) \end{aligned}$$

Some invocations:

$$\begin{aligned} &\texttt{map succ}\ [1,2,3] &&= [2,3,4] \\ &\texttt{map not}\ \ [\mathrm{True}, \mathrm{False}] &&= [\mathrm{False}, \mathrm{True}] \\ &\texttt{map even}\ [1,2,3] &&= [\mathrm{False}, \mathrm{True}, \mathrm{False}] \\ &\texttt{map not}\ \ [1,2,3] && \end{aligned}$$

Parametric Polymorphism in Haskell

A standard function:

$$\texttt{map} :: (\alpha \to \beta) \to [\alpha] \to [\beta]$$
$$\texttt{map}\ g\ [\,] = [\,]$$
$$\texttt{map}\ g\ (a : as) = (g\ a) : (\texttt{map}\ g\ as)$$

Some invocations:

$$\texttt{map succ}\ [1,2,3] = [2,3,4]$$
$$\texttt{map not}\ [\text{True}, \text{False}] = [\text{False}, \text{True}]$$
$$\texttt{map even}\ [1,2,3] = [\text{False}, \text{True}, \text{False}]$$
$$\texttt{map not}\ [1,2,3]$$

Parametric Polymorphism in Haskell

A standard function:

$$\begin{aligned}
&\texttt{map} :: (\alpha \to \beta) \to [\alpha] \to [\beta]\\
&\texttt{map}\ g\ [\,] \qquad\quad = [\,]\\
&\texttt{map}\ g\ (a : as) = (g\ a) : (\texttt{map}\ g\ as)
\end{aligned}$$

Some invocations:

$$\begin{aligned}
&\texttt{map succ}\ [1,2,3] &&= [2,3,4]\\
&\texttt{map not}\ \ [\text{True},\text{False}] &&= [\text{False},\text{True}]\\
&\texttt{map even}\ [1,2,3] &&= [\text{False},\text{True},\text{False}]\\
&\texttt{map not}\ \ [1,2,3] && ↯\ \text{rejected at compile-time}
\end{aligned}$$

Parametric Polymorphism in Haskell

A standard function:

$$\texttt{map} :: (\alpha \to \beta) \to [\alpha] \to [\beta]$$

Some invocations:

$$\begin{array}{lll} \texttt{map succ} & [1,2,3] & = [2,3,4] \\ \texttt{map not} & [\mathsf{True},\mathsf{False}] & = [\mathsf{False},\mathsf{True}] \\ \texttt{map even} & [1,2,3] & = [\mathsf{False},\mathsf{True},\mathsf{False}] \\ \texttt{map not} & [1,2,3] & \lightning \text{ rejected at compile-time} \end{array}$$

Another Example

$$\texttt{reverse} :: [\alpha] \rightarrow [\alpha]$$
$$\texttt{reverse}\ [\,] = [\,]$$
$$\texttt{reverse}\ (a : as) = (\texttt{reverse}\ as) +\!\!+ [a]$$

Another Example

$$
\begin{aligned}
&\texttt{reverse} :: [\alpha] \rightarrow [\alpha] \\
&\texttt{reverse}\ [\,] \qquad\quad = [\,] \\
&\texttt{reverse}\ (a : as) = (\texttt{reverse}\ as) +\!\!+ [a]
\end{aligned}
$$

For every choice of g and l:

$$\texttt{reverse}\ (\texttt{map}\ g\ l) \ = \ \texttt{map}\ g\ (\texttt{reverse}\ l)$$

Provable by induction.

Another Example

$$\begin{aligned}
&\texttt{reverse} :: [\alpha] \to [\alpha] \\
&\texttt{reverse}\ [\,] \qquad\quad = [\,] \\
&\texttt{reverse}\ (a : as) = (\texttt{reverse}\ as) \mathbin{+\!\!+} [a]
\end{aligned}$$

For every choice of g and l:

$$\texttt{reverse}\ (\texttt{map}\ g\ l) \;=\; \texttt{map}\ g\ (\texttt{reverse}\ l)$$

Provable by induction.

Or as a “free theorem” [Wadler, FPCA’89].

Another Example

$$\texttt{reverse} :: [\alpha] \to [\alpha]$$

For every choice of g and l:

$$\texttt{reverse}\ (\texttt{map}\ g\ l) \ = \ \texttt{map}\ g\ (\texttt{reverse}\ l)$$

Provable by induction.

Or as a "free theorem" [Wadler, FPCA'89].

Another Example

$$\texttt{reverse} :: [\alpha] \to [\alpha]$$

$$\texttt{tail} :: [\alpha] \to [\alpha]$$

For every choice of g and l:

$$\texttt{reverse}\ (\texttt{map}\ g\ l) = \texttt{map}\ g\ (\texttt{reverse}\ l)$$

$$\texttt{tail}\ (\texttt{map}\ g\ l) = \texttt{map}\ g\ (\texttt{tail}\ l)$$

Another Example

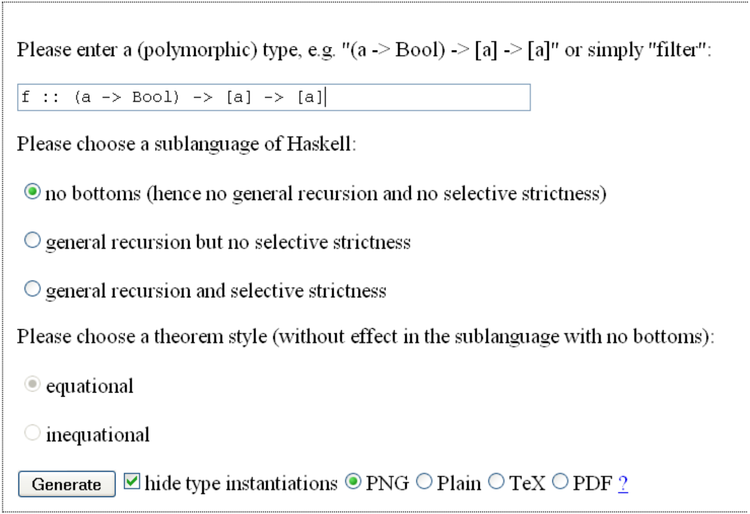
$$\texttt{reverse} :: [\alpha] \to [\alpha]$$
$$\texttt{tail} :: [\alpha] \to [\alpha]$$
$$\texttt{f} :: [\alpha] \to [\alpha]$$

For every choice of g and l:

$$\texttt{reverse}\ (\texttt{map}\ g\ l) = \texttt{map}\ g\ (\texttt{reverse}\ l)$$
$$\texttt{tail}\ (\texttt{map}\ g\ l) = \texttt{map}\ g\ (\texttt{tail}\ l)$$
$$\texttt{f}\ (\texttt{map}\ g\ l) = \texttt{map}\ g\ (\texttt{f}\ l)$$

Automatic Generation of Free Theorems

At `http://www-ps.iai.uni-bonn.de/ft`:



Automatic Generation of Free Theorems

The Free Theorem for "f :: forall a . (a -> Bool) -> [a] -> [a]"

$$\forall\, t_1,t_2 \in \text{TYPES},\ \text{R} \in \text{REL}(t_1,t_2).$$
$$\quad \forall\, p :: t_1 \to \textsc{Bool}.$$
$$\quad\quad \forall\, q :: t_2 \to \textsc{Bool}.$$
$$\quad\quad\quad (\forall\, (x, y) \in \text{R}.\ p\ x = q\ y)$$
$$\quad\quad\quad \Rightarrow (\forall\, (z, v) \in \text{LIFT}\{[]\}(\text{R}).\ (f\ p\ z, f\ q\ v) \in \text{LIFT}\{[]\}(\text{R}))$$

$$\text{LIFT}\{[]\}(\text{R})$$
$$\quad = \{([], [])\}$$
$$\quad \cup \{(x : xs, y : ys) \mid ((x, y) \in \text{R}) \wedge ((xs, ys) \in \text{LIFT}\{[]\}(\text{R}))\}$$

Reducing all permissable relation variables to functions

$$\forall\, t_1,t_2 \in \text{TYPES},\ g :: t_1 \to t_2.$$
$$\quad \forall\, p :: t_1 \to \textsc{Bool}.$$
$$\quad\quad \forall\, q :: t_2 \to \textsc{Bool}.$$
$$\quad\quad\quad (\forall\, x :: t_1.\ p\ x = q\ (g\ x))$$
$$\quad\quad\quad \Rightarrow (\forall\, y :: [t_1].\ \text{map}\ g\ (f\ p\ y) = f\ q\ (\text{map}\ g\ y))$$

Applications

- Short Cut Fusion [Gill et al., FPCA'93]

Applications

- Short Cut Fusion [Gill et al., FPCA'93]
- The Dual of Short Cut Fusion
  [Takano & Meijer, FPCA'95],
  [Svenningsson, ICFP'02]
- ...

Applications

- Short Cut Fusion [Gill et al., FPCA'93]
- The Dual of Short Cut Fusion
  [Takano & Meijer, FPCA'95],
  [Svenningsson, ICFP'02]
- ...
- Knuth's 0-1-principle and the like
  [Day et al., Haskell'99], [V., POPL'08]

Applications

- Short Cut Fusion [Gill et al., FPCA'93]
- The Dual of Short Cut Fusion [Takano & Meijer, FPCA'95], [Svenningsson, ICFP'02]
- ...
- Knuth's 0-1-principle and the like [Day et al., Haskell'99], [V., POPL'08]
- Bidirectionalization [V., POPL'09]

Applications

- Short Cut Fusion [Gill et al., FPCA'93]
- The Dual of Short Cut Fusion
  [Takano & Meijer, FPCA'95],
  [Svenningsson, ICFP'02]
- ...
- Knuth's 0-1-principle and the like
  [Day et al., Haskell'99], [V., POPL'08]
- Bidirectionalization [V., POPL'09]
- ...
- Testing polymorphic properties
  [Bernardy et al., ESOP'10]

What About Efficiency?

$\mathtt{f} :: \alpha \to \mathsf{Nat}$	$\mathtt{f} :: \alpha \to \alpha \to \alpha$	$\mathtt{f} :: \alpha \to (\alpha, \alpha)$
$\mathtt{f}\ (g\ x) = \mathtt{f}\ x$	$\mathtt{f}\ (g\ x)\ (g\ y) = g\ (\mathtt{f}\ x\ y)$	$\mathtt{f}\ (g\ x) = \textbf{let}\ y = \mathtt{f}\ x\ \textbf{in}\ (g\ (\mathtt{fst}\ y), g\ (\mathtt{snd}\ y))$

What About Efficiency?

	$\mathtt{f} :: \alpha \to \mathsf{Nat}$	$\mathtt{f} :: \alpha \to \alpha \to \alpha$	$\mathtt{f} :: \alpha \to (\alpha, \alpha)$
	$\mathtt{f}\ (g\ x) = \mathtt{f}\ x$	$\mathtt{f}\ (g\ x)\ (g\ y) = g\ (\mathtt{f}\ x\ y)$	$\mathtt{f}\ (g\ x) = \textbf{let}\ y = \mathtt{f}\ x\ \textbf{in}\ (g\ (\mathtt{fst}\ y), g\ (\mathtt{snd}\ y))$
call-by-value	$>$	$>$	$<$

What About Efficiency?

	$\mathtt{f} :: \alpha \to \mathrm{Nat}$	$\mathtt{f} :: \alpha \to \alpha \to \alpha$	$\mathtt{f} :: \alpha \to (\alpha, \alpha)$
	$\mathtt{f}\ (g\ x) = \mathtt{f}\ x$	$\mathtt{f}\ (g\ x)\ (g\ y) = g\ (\mathtt{f}\ x\ y)$	$\mathtt{f}\ (g\ x) = \mathbf{let}\ y = \mathtt{f}\ x\ \mathbf{in}\ (g\ (\mathtt{fst}\ y), g\ (\mathtt{snd}\ y))$
call-by-value	$>$	$>$	$<$
call-by-name	$=$	$=$	$<$ (?)

What About Efficiency?

	$\texttt{f} :: \alpha \to \mathsf{Nat}$	$\texttt{f} :: \alpha \to \alpha \to \alpha$	$\texttt{f} :: \alpha \to (\alpha, \alpha)$
	$\texttt{f}\ (g\ x) = \texttt{f}\ x$	$\texttt{f}\ (g\ x)\ (g\ y) = g\ (\texttt{f}\ x\ y)$	$\texttt{f}\ (g\ x) = \textbf{let}\ y = \texttt{f}\ x\ \textbf{in}\ (g\ (\texttt{fst}\ y), g\ (\texttt{snd}\ y))$
call-by-value	$>$	$>$	$<$
call-by-name	$=$	$=$	$<$ (?)
call-by-need	$=$	$=$	$<$

What About Efficiency?

	$\texttt{f} :: \alpha \to \mathsf{Nat}$	$\texttt{f} :: \alpha \to \alpha \to \alpha$	$\texttt{f} :: \alpha \to (\alpha, \alpha)$
	$\texttt{f}\ (g\ x) = \texttt{f}\ x$	$\texttt{f}\ (g\ x)\ (g\ y) = g\ (\texttt{f}\ x\ y)$	$\texttt{f}\ (g\ x) = \textbf{let}\ y = \texttt{f}\ x\ \textbf{in}\ (g\ (\texttt{fst}\ y), g\ (\texttt{snd}\ y))$
call-by-value	$>$	$>$	$<$
call-by-name	$=$	$=$	$<$ (?)
call-by-need	$=$	$=$	$<$

What About Efficiency?

Now, how about, say `f` $:: \alpha \to \alpha \to (\alpha, \alpha)$?

What About Efficiency?

Now, how about, say `f` $:: \alpha \to \alpha \to (\alpha, \alpha)$?

Or `f` $:: [\alpha] \to [\alpha]$?

What About Efficiency?

Now, how about, say $\texttt{f} :: \alpha \to \alpha \to (\alpha, \alpha)$?

Or $\texttt{f} :: [\alpha] \to [\alpha]$?

And how to actually establish such statements?

But isn't it Somehow Trivial?

Recall $\mathtt{f} :: \alpha \rightarrow \mathsf{Nat}$, with standard free theorem:

$$\mathtt{f}\ (g\ x) = \mathtt{f}\ x$$

for all choices of types τ_1, τ_2, function $g :: \tau_1 \rightarrow \tau_2$, and $x :: \tau_1$.

But isn't it Somehow Trivial?

Recall $\texttt{f} :: \alpha \to \mathsf{Nat}$, with standard free theorem:

$$\texttt{f}\ (g\ x) = \texttt{f}\ x$$

for all choices of types τ_1, τ_2, function $g :: \tau_1 \to \tau_2$, and $x :: \tau_1$. Clearly, this means that $\texttt{f}$ is a constant function, i.e., for all x and y:

$$\texttt{f}\ y = \texttt{f}\ x$$

But isn't it Somehow Trivial?

Recall $\texttt{f} :: \alpha \rightarrow \mathsf{Nat}$, with standard free theorem:

$$\texttt{f}\ (g\ x) = \texttt{f}\ x$$

for all choices of types τ_1, τ_2, function $g :: \tau_1 \rightarrow \tau_2$, and $x :: \tau_1$. Clearly, this means that $\texttt{f}$ is a constant function, i.e., for all x and y:

$$\texttt{f}\ y = \texttt{f}\ x$$

So, "obviously",

$$\texttt{f}\ (g\ x) \sqsupseteq \texttt{f}\ x$$

where $\sqsupseteq$ means "the same result, and equally fast or slower".

But isn't it Somehow Trivial?

Recall $\texttt{f} :: \alpha \to \text{Nat}$, with standard free theorem:

$$\texttt{f}\ (g\ x) = \texttt{f}\ x$$

for all choices of types τ_1, τ_2, function $g :: \tau_1 \to \tau_2$, and $x :: \tau_1$. Clearly, this means that $\texttt{f}$ is a constant function, i.e., for all x and y:

$$\texttt{f}\ y = \texttt{f}\ x$$

So, "obviously",

$$\texttt{f}\ (g\ x) \sqsupseteq \texttt{f}\ x$$

where $\sqsupseteq$ means "the same result, and equally fast or slower". What's wrong with this reasoning?

But isn't it Somehow Trivial?

Consider:

$$\texttt{f}\ x = \textbf{if}\ x == 0\ \textbf{then}\ 0\ \textbf{else}\ \texttt{f}\ (x-1) \qquad\qquad g\ x = 0$$

But isn't it Somehow Trivial?

Consider:

$$\texttt{f}\ x = \textbf{if}\ x == 0 \qquad\qquad g\ x = 0$$
$$\textbf{then}\ 0\ \textbf{else}\ \texttt{f}\ (x - 1)$$

Certainly, for all x and y:

$$\texttt{f}\ y = \texttt{f}\ x$$

But isn't it Somehow Trivial?

Consider:

$$\texttt{f}\ x = \textbf{if}\ x == 0\ \textbf{then}\ 0\ \textbf{else}\ \texttt{f}\ (x-1) \qquad g\ x = 0$$

Certainly, for all x and y:

$$\texttt{f}\ y = \texttt{f}\ x$$

But certainly not, for $x > 0$:

$$\texttt{f}\ (g\ x) \sqsupseteq \texttt{f}\ x$$

But isn't it Somehow Trivial?

Consider:

$$\texttt{f} :: \text{Nat} \rightarrow \text{Nat} \qquad\qquad g :: \alpha \rightarrow \text{Nat}$$

$$\texttt{f}\ x = \textbf{if}\ x == 0 \qquad\qquad g\ x = 0$$

$$\qquad \textbf{then}\ 0\ \textbf{else}\ \texttt{f}\ (x - 1)$$

Certainly, for all x and y:

$$\texttt{f}\ y = \texttt{f}\ x$$

But certainly not, for $x > 0$:

$$\texttt{f}\ (g\ x) \sqsupseteq \texttt{f}\ x$$

But isn't it Somehow Trivial?

Consider:

$$\texttt{f} :: \text{Nat} \to \text{Nat} \qquad\qquad g :: \alpha \to \text{Nat}$$

$$\texttt{f}\ x = \textbf{if}\ x == 0\ \textbf{then}\ 0\ \textbf{else}\ \texttt{f}\ (x - 1) \qquad\qquad g\ x = 0$$

Certainly, for all x and y:

$$\texttt{f}\ y = \texttt{f}\ x$$

But certainly not, for $x > 0$:

$$\texttt{f}\ (g\ x) \sqsupseteq \texttt{f}\ x$$

Exploiting polymorphism is really essential, and not just for the extensional statements!

Free Theorems, Formally — In a Nutshell

Syntax for a typed λ-calculus:

$$\tau ::= \alpha \mid \mathsf{Nat} \mid \tau \rightarrow \tau \mid \ldots$$
$$t ::= x \mid n \mid t + t \mid \lambda x :: \tau.t \mid t\ t \mid \ldots$$

Free Theorems, Formally — In a Nutshell

Syntax for a typed λ-calculus:

$$\tau ::= \alpha \mid \mathsf{Nat} \mid \tau \to \tau \mid \ldots$$
$$t ::= x \mid n \mid t + t \mid \lambda x :: \tau.t \mid t\ t \mid \ldots$$

Semantics:

$$\begin{aligned}
[\![\alpha]\!]_\theta &= \theta(\alpha) \\
[\![\mathsf{Nat}]\!]_\theta &= \mathbb{N} \\
[\![\tau_1 \to \tau_2]\!]_\theta &= {[\![\tau_2]\!]_\theta}^{[\![\tau_1]\!]_\theta} \\
\theta &\in \mathsf{Set}^{\mathsf{TVar}}
\end{aligned}$$

Free Theorems, Formally — In a Nutshell

Syntax for a typed λ-calculus:

$$\tau ::= \alpha \mid \mathsf{Nat} \mid \tau \to \tau \mid \ldots$$
$$t ::= x \mid n \mid t + t \mid \lambda x :: \tau.t \mid t\ t \mid \ldots$$

Semantics:

$$\begin{aligned} [\![\alpha]\!]_\theta &= \theta(\alpha) \\ [\![\mathsf{Nat}]\!]_\theta &= \mathbb{N} \\ [\![\tau_1 \to \tau_2]\!]_\theta &= {[\![\tau_2]\!]_\theta}^{[\![\tau_1]\!]_\theta} \\ \theta &\in \mathsf{Set}^{\mathsf{TVar}} \end{aligned}
\qquad
\begin{aligned} [\![x]\!]_\sigma &= \sigma(x) \\ [\![n]\!]_\sigma &= \mathbf{n} \\ [\![\lambda x :: \tau.t]\!]_\sigma &= \lambda \mathbf{v}.[\![t]\!]_{\sigma[x \mapsto \mathbf{v}]} \\ [\![t_1\ t_2]\!]_\sigma &= [\![t_1]\!]_\sigma\ [\![t_2]\!]_\sigma \end{aligned}$$

Free Theorems, Formally — In a Nutshell

Syntax for a typed λ-calculus:

$$\tau ::= \alpha \mid \mathsf{Nat} \mid \tau \to \tau \mid \ldots$$
$$t ::= x \mid n \mid t + t \mid \lambda x :: \tau . t \mid t\ t \mid \ldots$$

Semantics:

$$\begin{aligned}
[\![\alpha]\!]_\theta &= \theta(\alpha) & [\![x]\!]_\sigma &= \sigma(x) \\
[\![\mathsf{Nat}]\!]_\theta &= \mathbb{N} & [\![n]\!]_\sigma &= \mathbf{n} \\
[\![\tau_1 \to \tau_2]\!]_\theta &= {[\![\tau_2]\!]_\theta}^{[\![\tau_1]\!]_\theta} & [\![\lambda x :: \tau . t]\!]_\sigma &= \lambda \mathbf{v}. [\![t]\!]_{\sigma[x \mapsto \mathbf{v}]} \\
\theta &\in \mathsf{Set}^{\mathsf{TVar}} & [\![t_1\ t_2]\!]_\sigma &= [\![t_1]\!]_\sigma\ [\![t_2]\!]_\sigma
\end{aligned}$$

Logical relation:

$$\begin{aligned}
\Delta_{\alpha,\rho} &= \rho(\alpha) \qquad\qquad \Delta_{\mathsf{Nat},\rho} = id_{\mathbb{N}} \\
\Delta_{\tau_1 \to \tau_2,\rho} &= \{(\mathbf{f}, \mathbf{g}) \mid \forall (\mathbf{x}, \mathbf{y}) \in \Delta_{\tau_1,\rho}.\ (\mathbf{f}\ \mathbf{x}, \mathbf{g}\ \mathbf{y}) \in \Delta_{\tau_2,\rho}\}
\end{aligned}$$

with $\rho \in \mathsf{Rel}^{\mathsf{TVar}}$

Free Theorems, Formally — In a Nutshell

Syntax for a typed λ-calculus:

$$\tau ::= \alpha \mid \mathsf{Nat} \mid \tau \to \tau \mid \ldots$$
$$t ::= x \mid n \mid t + t \mid \lambda x :: \tau . t \mid t\ t \mid \ldots$$

Semantics:

$$\begin{array}{ll} [\![\alpha]\!]_\theta = \theta(\alpha) & [\![x]\!]_\sigma = \sigma(x) \\ [\![\mathsf{Nat}]\!]_\theta = \mathbb{N} & [\![n]\!]_\sigma = \mathbf{n} \\ [\![\tau_1 \to \tau_2]\!]_\theta = {[\![\tau_2]\!]_\theta}^{[\![\tau_1]\!]_\theta} & [\![\lambda x :: \tau . t]\!]_\sigma = \lambda \mathbf{v}. [\![t]\!]_{\sigma[x \mapsto \mathbf{v}]} \\ \theta \in \mathsf{Set}^{\mathsf{TVar}} & [\![t_1\ t_2]\!]_\sigma = [\![t_1]\!]_\sigma\ [\![t_2]\!]_\sigma \end{array}$$

Logical relation:

$$\Delta_{\alpha,\rho} = \rho(\alpha) \qquad \Delta_{\mathsf{Nat},\rho} = id_{\mathbb{N}}$$
$$\Delta_{\tau_1 \to \tau_2,\rho} = \{(\mathbf{f}, \mathbf{g}) \mid \forall (\mathbf{x}, \mathbf{y}) \in \Delta_{\tau_1,\rho}.\ (\mathbf{f}\ \mathbf{x}, \mathbf{g}\ \mathbf{y}) \in \Delta_{\tau_2,\rho}\}$$

with $\rho \in \mathsf{Rel}^{\mathsf{TVar}}$

Theorem: for closed term t of type τ, $([\![t]\!]_\emptyset, [\![t]\!]_\emptyset) \in \Delta_{\tau,\rho}$.

An Example Derivation, for `f` $:: \alpha \to \mathsf{Nat}$

$$\forall \rho, t \text{ closed with } t :: \tau.\ (\llbracket t \rrbracket_\emptyset, \llbracket t \rrbracket_\emptyset) \in \Delta_{\tau,\rho}$$

An Example Derivation, for $\texttt{f} :: \alpha \to \mathsf{Nat}$

$$\forall \rho, t \text{ closed with } t :: \tau.\ ([\![t]\!]_\emptyset, [\![t]\!]_\emptyset) \in \Delta_{\tau,\rho}$$

$\Rightarrow$ ($t = \texttt{f}$ and $\tau = \alpha \to \mathsf{Nat}$)

$$\forall \rho.\ ([\![\texttt{f}]\!]_\emptyset, [\![\texttt{f}]\!]_\emptyset) \in \Delta_{\alpha \to \mathsf{Nat},\rho}$$

An Example Derivation, for $\texttt{f} :: \alpha \to \mathsf{Nat}$

$$\forall \rho, t \text{ closed with } t :: \tau.\ (\llbracket t \rrbracket_\emptyset, \llbracket t \rrbracket_\emptyset) \in \Delta_{\tau,\rho}$$

$\Rightarrow$ $(t = \texttt{f} \text{ and } \tau = \alpha \to \mathsf{Nat})$

$$\forall \rho.\ (\llbracket \texttt{f} \rrbracket_\emptyset, \llbracket \texttt{f} \rrbracket_\emptyset) \in \Delta_{\alpha \to \mathsf{Nat},\rho}$$

$\Leftrightarrow$ $(\Delta_{\tau_1 \to \tau_2,\rho} = \{(\mathbf{f}, \mathbf{g}) \mid \forall (\mathbf{x}, \mathbf{y}) \in \Delta_{\tau_1,\rho}.\ (\mathbf{f}\ \mathbf{x}, \mathbf{g}\ \mathbf{y}) \in \Delta_{\tau_2,\rho}\})$

$$\forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta_{\alpha,\rho}.\ (\llbracket \texttt{f} \rrbracket_\emptyset\ \mathbf{x}, \llbracket \texttt{f} \rrbracket_\emptyset\ \mathbf{y}) \in \Delta_{\mathsf{Nat},\rho}$$

An Example Derivation, for $\mathtt{f} :: \alpha \to \mathsf{Nat}$

$$\forall \rho, t \text{ closed with } t :: \tau.\ ([\![t]\!]_\emptyset, [\![t]\!]_\emptyset) \in \Delta_{\tau,\rho}$$

$\Rightarrow$ $(t = \mathtt{f} \text{ and } \tau = \alpha \to \mathsf{Nat})$

$$\forall \rho.\ ([\![\mathtt{f}]\!]_\emptyset, [\![\mathtt{f}]\!]_\emptyset) \in \Delta_{\alpha \to \mathsf{Nat},\rho}$$

$\Leftrightarrow$ $(\Delta_{\tau_1 \to \tau_2,\rho} = \{(\mathbf{f}, \mathbf{g}) \mid \forall (\mathbf{x}, \mathbf{y}) \in \Delta_{\tau_1,\rho}.\ (\mathbf{f}\ \mathbf{x}, \mathbf{g}\ \mathbf{y}) \in \Delta_{\tau_2,\rho}\})$

$$\forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta_{\alpha,\rho}.\ ([\![\mathtt{f}]\!]_\emptyset\ \mathbf{x}, [\![\mathtt{f}]\!]_\emptyset\ \mathbf{y}) \in \Delta_{\mathsf{Nat},\rho}$$

$\Rightarrow$ $(\Delta_{\alpha,\rho} = \rho(\alpha),\ \rho(\alpha) := \mathbf{g} \in [\![\tau_2]\!]_\emptyset{}^{[\![\tau_1]\!]_\emptyset},\ \Delta_{\mathsf{Nat},\rho} = id_{\mathbb{N}})$

$$\forall \mathbf{g} \in [\![\tau_2]\!]_\emptyset{}^{[\![\tau_1]\!]_\emptyset}, \mathbf{x} \in [\![\tau_1]\!]_\emptyset.\ [\![\mathtt{f}]\!]_\emptyset\ \mathbf{x} = [\![\mathtt{f}]\!]_\emptyset\ (\mathbf{g}\ \mathbf{x})$$

An Example Derivation, for $\texttt{f} :: \alpha \to \mathsf{Nat}$

$$\forall \rho, t \text{ closed with } t :: \tau.\ ([\![t]\!]_\emptyset, [\![t]\!]_\emptyset) \in \Delta_{\tau,\rho}$$

$\Rightarrow$ ($t = \texttt{f}$ and $\tau = \alpha \to \mathsf{Nat}$)

$$\forall \rho.\ ([\![\texttt{f}]\!]_\emptyset, [\![\texttt{f}]\!]_\emptyset) \in \Delta_{\alpha\to\mathsf{Nat},\rho}$$

$\Leftrightarrow$ ($\Delta_{\tau_1\to\tau_2,\rho} = \{(\mathbf{f}, \mathbf{g}) \mid \forall(\mathbf{x}, \mathbf{y}) \in \Delta_{\tau_1,\rho}.\ (\mathbf{f}\ \mathbf{x}, \mathbf{g}\ \mathbf{y}) \in \Delta_{\tau_2,\rho}\}$)

$$\forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta_{\alpha,\rho}.\ ([\![\texttt{f}]\!]_\emptyset\ \mathbf{x}, [\![\texttt{f}]\!]_\emptyset\ \mathbf{y}) \in \Delta_{\mathsf{Nat},\rho}$$

$\Rightarrow$ ($\Delta_{\alpha,\rho} = \rho(\alpha)$, $\rho(\alpha) := \mathbf{g} \in [\![\tau_2]\!]_\emptyset{}^{[\![\tau_1]\!]_\emptyset}$, $\Delta_{\mathsf{Nat},\rho} = id_{\mathbb{N}}$)

$$\forall \mathbf{g} \in [\![\tau_2]\!]_\emptyset{}^{[\![\tau_1]\!]_\emptyset}, \mathbf{x} \in [\![\tau_1]\!]_\emptyset.\ [\![\texttt{f}]\!]_\emptyset\ \mathbf{x} = [\![\texttt{f}]\!]_\emptyset\ (\mathbf{g}\ \mathbf{x})$$

$\Rightarrow$ (term semantics)

$$\forall g :: \tau_1 \to \tau_2, x :: \tau_1.\ [\![\texttt{f}\ x]\!]_\emptyset = [\![\texttt{f}\ (g\ x)]\!]_\emptyset$$

Bringing Costs into the Picture

New semantics:

$$
\begin{aligned}
[\![x]\!]^{¢}_{\sigma} &= (\sigma(x), 0) \\
[\![n]\!]^{¢}_{\sigma} &= (\mathbf{n}, 0) \\
[\![\lambda x :: \tau . t]\!]^{¢}_{\sigma} &= (\lambda \mathbf{v}. 1 \triangleright [\![t]\!]^{¢}_{\sigma[x \mapsto \mathbf{v}]}, 0) \\
[\![t_1 \ \ t_2]\!]^{¢}_{\sigma} &= [\![t_1]\!]^{¢}_{\sigma} \ ¢ \ [\![t_2]\!]^{¢}_{\sigma}
\end{aligned}
$$

where: $c \triangleright (\mathbf{v}, c') = (\mathbf{v}, c + c')$ and

$$
\mathbf{f} \ ¢ \ \mathbf{x} = (c + c') \triangleright (\mathbf{g} \ \mathbf{v}) \quad \text{if } \mathbf{f} = (\mathbf{g}, c), \ \mathbf{x} = (\mathbf{v}, c')
$$

Bringing Costs into the Picture

New semantics:

$$
\begin{aligned}
[\![x]\!]^{¢}_{\sigma} &= (\sigma(x), 0) \\
[\![n]\!]^{¢}_{\sigma} &= (\mathbf{n}, 0) \\
[\![\lambda x :: \tau . t]\!]^{¢}_{\sigma} &= (\lambda \mathbf{v}. 1 \rhd [\![t]\!]^{¢}_{\sigma[x \mapsto \mathbf{v}]}, 0) \\
[\![t_1 \;\; t_2]\!]^{¢}_{\sigma} &= [\![t_1]\!]^{¢}_{\sigma} \; ¢ \; [\![t_2]\!]^{¢}_{\sigma}
\end{aligned}
$$

where: $c \rhd (\mathbf{v}, c') = (\mathbf{v}, c + c')$ and

$$\mathbf{f} \; ¢ \; \mathbf{x} = (c + c') \rhd (\mathbf{g} \;\; \mathbf{v}) \;\; \text{if } \mathbf{f} = (\mathbf{g}, c), \; \mathbf{x} = (\mathbf{v}, c')$$

Note: $t :: \alpha \to \text{Nat}$ implies $[\![t]\!]^{¢}_{\emptyset} \in \mathcal{C}(\mathcal{C}(\mathbb{N})^{\theta(\alpha)})$,
for every $\theta \in \text{Set}^{\text{TVar}}$,
where $\mathcal{C}(S) = \{(\mathbf{v}, c) \mid \mathbf{v} \in S, c \in \mathbb{Z}\}$.

How about the Logical Relation?

Tempting would be:

$$\Delta^{¢}_{\alpha,\rho} = \rho(\alpha) \qquad \Delta^{¢}_{\mathsf{Nat},\rho} = id_{\mathbb{N}\times\mathbb{Z}}$$

$$\Delta^{¢}_{\tau_1\to\tau_2,\rho} = \{(\mathbf{f},\mathbf{g}) \mid \forall(\mathbf{x},\mathbf{y}) \in \Delta^{¢}_{\tau_1,\rho}.\ (\mathbf{f}\ ¢\ \mathbf{x}, \mathbf{g}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\tau_2,\rho}\}$$

with $\rho(\alpha_1), \rho(\alpha_2), \ldots \subseteq \mathcal{C}(S_i) \times \mathcal{C}(T_i)$

How about the Logical Relation?

Tempting would be:

$$\Delta^{¢}_{\alpha,\rho} = \rho(\alpha) \qquad \Delta^{¢}_{\mathsf{Nat},\rho} = id_{\mathbb{N}\times\mathbb{Z}}$$

$$\Delta^{¢}_{\tau_1\to\tau_2,\rho} = \{(\mathbf{f},\mathbf{g}) \mid \forall(\mathbf{x},\mathbf{y}) \in \Delta^{¢}_{\tau_1,\rho}.\ (\mathbf{f} \mathbin{¢} \mathbf{x}, \mathbf{g} \mathbin{¢} \mathbf{y}) \in \Delta^{¢}_{\tau_2,\rho}\}$$

with $\rho(\alpha_1), \rho(\alpha_2), \ldots \subseteq \mathcal{C}(S_i) \times \mathcal{C}(T_i)$

But NO, does not work!

How about the Logical Relation?

Tempting would be:

$$\Delta^{¢}_{\alpha,\rho} = \rho(\alpha) \qquad \Delta^{¢}_{\mathsf{Nat},\rho} = id_{\mathbb{N}\times\mathbb{Z}}$$

$$\Delta^{¢}_{\tau_1\to\tau_2,\rho} = \{(\mathbf{f},\mathbf{g}) \mid \forall(\mathbf{x},\mathbf{y}) \in \Delta^{¢}_{\tau_1,\rho}.\ (\mathbf{f}\ ¢\ \mathbf{x}, \mathbf{g}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\tau_2,\rho}\}$$

with $\rho(\alpha_1), \rho(\alpha_2), \ldots \subseteq \mathcal{C}(S_i) \times \mathcal{C}(T_i)$

But NO, does not work! It would allow us to conclude from:

$$\forall\rho, \mathtt{f} :: \alpha \to \mathsf{Nat}.\ ([\![\mathtt{f}]\!]^{¢}_{\emptyset}, [\![\mathtt{f}]\!]^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha\to\mathsf{Nat},\rho}$$

that:

$$\forall g :: \tau_1 \to \tau_2, x :: \tau_1.\ [\![\mathtt{f}\ x]\!]^{¢}_{\emptyset} = [\![\mathtt{f}\ (g\ x)]\!]^{¢}_{\emptyset}$$

How about the Logical Relation?

Much more disciplined:

$$\Delta^{¢}_{\alpha,\rho} = \mathcal{C}(\rho(\alpha)) \qquad \Delta^{¢}_{\mathsf{Nat},\rho} = id_{\mathbb{N}\times\mathbb{Z}}$$

$$\Delta^{¢}_{\tau_1\to\tau_2,\rho} = \{(\mathbf{f},\mathbf{g}) \mid cost(\mathbf{f}) = cost(\mathbf{g}) \land \forall(\mathbf{x},\mathbf{y}) \in \Delta^{¢}_{\tau_1,\rho}.\ (\mathbf{f}\ ¢\ \mathbf{x}, \mathbf{g}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\tau_2,\rho}\}$$

with $\rho(\alpha_1), \rho(\alpha_2), \ldots \subseteq S_i \times T_i$,

where $\mathcal{C}(R) = \{((\mathbf{u}, c), (\mathbf{v}, c) \mid (\mathbf{u}, \mathbf{v}) \in R, c \in \mathbb{Z}\}$

How about the Logical Relation?

Much more disciplined:

$$\Delta^{¢}_{\alpha,\rho} = \mathcal{C}(\rho(\alpha)) \qquad \Delta^{¢}_{\mathsf{Nat},\rho} = id_{\mathbb{N}\times\mathbb{Z}}$$

$$\Delta^{¢}_{\tau_1\to\tau_2,\rho} = \{(\mathbf{f},\mathbf{g}) \mid cost(\mathbf{f}) = cost(\mathbf{g}) \wedge \forall(\mathbf{x},\mathbf{y}) \in \Delta^{¢}_{\tau_1,\rho}.\ (\mathbf{f}\ ¢\ \mathbf{x}, \mathbf{g}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\tau_2,\rho}\}$$

with $\rho(\alpha_1), \rho(\alpha_2), \ldots \subseteq S_i \times T_i$,

where $\mathcal{C}(R) = \{((\mathbf{u}, c), (\mathbf{v}, c) \mid (\mathbf{u}, \mathbf{v}) \in R, c \in \mathbb{Z}\}$

Now indeed ...

Theorem: for closed term t of type τ,

$$([\![t]\!]^{¢}_{\emptyset}, [\![t]\!]^{¢}_{\emptyset}) \in \Delta^{¢}_{\tau,\rho}$$

Now, What about our Example $\mathtt{f} :: \alpha \to \mathsf{Nat}$?

$$\forall \rho.\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha \to \mathsf{Nat}, \rho}$$

Now, What about our Example $\texttt{f} :: \alpha \to \mathsf{Nat}$?

$$\forall \rho.\ (\llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}, \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha \to \mathsf{Nat},\rho}$$

$$\Rightarrow \ (\Delta^{¢}_{\tau_1 \to \tau_2,\rho} = \{(\mathbf{f}, \mathbf{g}) \mid \mathit{cost}(\mathbf{f}) = \mathit{cost}(\mathbf{g}) \land \forall (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\tau_1,\rho}.\ (\mathbf{f} \ ¢ \ \mathbf{x}, \mathbf{g} \ ¢ \ \mathbf{y}) \in \Delta^{¢}_{\tau_2,\rho}\})$$

$$\forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\alpha,\rho}.\ (\llbracket \texttt{f} \rrbracket^{¢}_{\emptyset} \ ¢ \ \mathbf{x}, \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset} \ ¢ \ \mathbf{y}) \in \Delta^{¢}_{\mathsf{Nat},\rho}$$

Now, What about our Example $\texttt{f} :: \alpha \to \mathsf{Nat}$?

$$\forall \rho.\ (\llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}, \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha \to \mathsf{Nat}, \rho}$$

$$\Rightarrow\ (\Delta^{¢}_{\tau_1 \to \tau_2, \rho} = \{(\mathbf{f}, \mathbf{g}) \mid cost(\mathbf{f}) = cost(\mathbf{g}) \wedge \forall (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\tau_1, \rho}.\ (\mathbf{f} \ ¢\ \mathbf{x}, \mathbf{g} \ ¢\ \mathbf{y}) \in \Delta^{¢}_{\tau_2, \rho}\})$$

$$\forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\alpha, \rho}.\ (\llbracket \texttt{f} \rrbracket^{¢}_{\emptyset} \ ¢\ \mathbf{x}, \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset} \ ¢\ \mathbf{y}) \in \Delta^{¢}_{\mathsf{Nat}, \rho}$$

$$\Rightarrow\ (\Delta^{¢}_{\alpha, \rho} = \mathcal{C}(\rho(\alpha)),\ \rho(\alpha) := R \in \mathsf{Rel},\ \Delta^{¢}_{\mathsf{Nat}, \rho} = id_{\mathbb{N} \times \mathbb{Z}})$$

$$\forall R \in \mathsf{Rel}, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R).\ \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset} \ ¢\ \mathbf{x} = \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset} \ ¢\ \mathbf{y}$$

Now, What about our Example $\mathtt{f} :: \alpha \to \mathsf{Nat}$?

$$\forall \rho.\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha \to \mathsf{Nat}, \rho}$$

$$\Rightarrow\ (\Delta^{¢}_{\tau_1 \to \tau_2, \rho} = \{(\mathbf{f}, \mathbf{g}) \mid \mathit{cost}(\mathbf{f}) = \mathit{cost}(\mathbf{g}) \wedge \forall (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\tau_1, \rho}.\ (\mathbf{f}\ ¢\ \mathbf{x}, \mathbf{g}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\tau_2, \rho}\})$$

$$\forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\alpha, \rho}.\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\mathsf{Nat}, \rho}$$

$$\Rightarrow\ (\Delta^{¢}_{\alpha, \rho} = \mathcal{C}(\rho(\alpha)),\ \rho(\alpha) := R \in \mathsf{Rel},\ \Delta^{¢}_{\mathsf{Nat}, \rho} = \mathit{id}_{\mathbb{N} \times \mathbb{Z}})$$

$$\forall R \in \mathsf{Rel}, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R).\ \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x} = \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}$$

How to choose R to bring $g :: \tau_1 \to \tau_2$ into play?

Now, What about our Example $\texttt{f} :: \alpha \to \mathsf{Nat}$?

$$\forall \rho.\ (\llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}, \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha \to \mathsf{Nat}, \rho}$$

$$\Rightarrow\ (\Delta^{¢}_{\tau_1 \to \tau_2, \rho} = \{(\mathbf{f}, \mathbf{g}) \mid \mathit{cost}(\mathbf{f}) = \mathit{cost}(\mathbf{g}) \wedge \forall (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\tau_1, \rho}.\ (\mathbf{f}\ ¢\ \mathbf{x}, \mathbf{g}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\tau_2, \rho}\})$$

$$\forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\alpha, \rho}.\ (\llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x}, \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\mathsf{Nat}, \rho}$$

$$\Rightarrow\ (\Delta^{¢}_{\alpha, \rho} = \mathcal{C}(\rho(\alpha)),\ \rho(\alpha) := R \in \mathsf{Rel},\ \Delta^{¢}_{\mathsf{Nat}, \rho} = \mathit{id}_{\mathbb{N} \times \mathbb{Z}})$$

$$\forall R \in \mathsf{Rel}, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R).\ \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x} = \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}$$

How to choose R to bring $g :: \tau_1 \to \tau_2$ into play?

Problem: $\{(\mathbf{x}, \mathbf{y}) \mid \llbracket g \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x} = \mathbf{y}\}$ not $\mathcal{C}(R)$ for any R

Now, What about our Example $\mathtt{f} :: \alpha \to \mathsf{Nat}$?

Problem: $\{(\mathbf{x}, \mathbf{y}) \mid [\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x} = \mathbf{y}\}$ not $\mathcal{C}(R)$ for any R

Solution: but

$$\{(c \rhd appCost(g, \mathbf{x}) \rhd \mathbf{x}, c \rhd \mathbf{y}) \mid [\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x} = \mathbf{y}, c \in \mathbb{Z}\} = \mathcal{C}(R^{g})$$

for

$$R^{g} = \{(val(\mathbf{x}), val([\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x})) \mid \mathbf{x} \in [\![\tau_1]\!]^{¢}_{\emptyset}\}$$

where $appCost(g, \mathbf{x}) = cost([\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x}) - cost(\mathbf{x})$

Now, What about our Example $\mathtt{f} :: \alpha \rightarrow \mathsf{Nat}$?

Problem: $\{(\mathbf{x}, \mathbf{y}) \mid [\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x} = \mathbf{y}\}$ not $\mathcal{C}(R)$ for any R

Solution: but

$$\{(c \rhd appCost(g, \mathbf{x}) \rhd \mathbf{x}, c \rhd \mathbf{y}) \mid [\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x} = \mathbf{y}, c \in \mathbb{Z}\} = \mathcal{C}(R^g)$$

for

$$R^g = \{(val(\mathbf{x}), val([\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x})) \mid \mathbf{x} \in [\![\tau_1]\!]^{¢}_{\emptyset}\}$$

where $appCost(g, \mathbf{x}) = cost([\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x}) - cost(\mathbf{x})$

Now:

$$\forall R \in \mathsf{Rel}, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R). \; [\![\mathtt{f}]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x} = [\![\mathtt{f}]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{y}$$

$$\Rightarrow \; \forall \mathbf{x} \in [\![\tau_1]\!]^{¢}_{\emptyset}. \; [\![\mathtt{f}]\!]^{¢}_{\emptyset} \; ¢ \; (appCost(g, \mathbf{x}) \rhd \mathbf{x}) = [\![\mathtt{f}]\!]^{¢}_{\emptyset} \; ¢ \; ([\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x})$$

Now, What about our Example $\texttt{f} :: \alpha \to \mathsf{Nat}$?

Now:

$$\begin{aligned} & \forall R \in \mathsf{Rel}, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R).\ [\![\texttt{f}]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{x} = [\![\texttt{f}]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{y} \\ \Rightarrow\ & \forall \mathbf{x} \in [\![\tau_1]\!]^{¢}_{\emptyset}.\ [\![\texttt{f}]\!]^{¢}_{\emptyset}\ ¢\ (\mathit{appCost}(g, \mathbf{x}) \triangleright \mathbf{x}) \\ & \qquad = [\![\texttt{f}]\!]^{¢}_{\emptyset}\ ¢\ ([\![g]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{x}) \end{aligned}$$

Now, What about our Example $\texttt{f} :: \alpha \to \mathsf{Nat}$?

Now:

$$
\begin{aligned}
& \forall R \in \mathsf{Rel}, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R).\ [\![\texttt{f}]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{x} = [\![\texttt{f}]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{y} \\
\Rightarrow\ & \forall \mathbf{x} \in [\![\tau_1]\!]^{¢}_{\emptyset}.\ [\![\texttt{f}]\!]^{¢}_{\emptyset}\ ¢\ (\mathit{appCost}(g, \mathbf{x}) \rhd \mathbf{x}) \\
& \qquad = [\![\texttt{f}]\!]^{¢}_{\emptyset}\ ¢\ ([\![g]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{x}) \\
\Rightarrow\ & \forall x :: \tau_1.\ \mathit{appCost}(g, [\![x]\!]^{¢}_{\emptyset}) \rhd ([\![\texttt{f}]\!]^{¢}_{\emptyset}\ ¢\ [\![x]\!]^{¢}_{\emptyset}) \\
& \qquad = [\![\texttt{f}]\!]^{¢}_{\emptyset}\ ¢\ ([\![g]\!]^{¢}_{\emptyset}\ ¢\ [\![x]\!]^{¢}_{\emptyset})
\end{aligned}
$$

Now, What about our Example $\mathtt{f} :: \alpha \to \mathsf{Nat}$?

Now:

$$\forall R \in \mathsf{Rel}, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R).\ [\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{x} = [\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{y}$$

$$\Rightarrow \forall \mathbf{x} \in [\![\tau_1]\!]^{¢}_{\emptyset}.\ [\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ (\mathit{appCost}(g, \mathbf{x}) \triangleright \mathbf{x}) = [\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ ([\![g]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{x})$$

$$\Rightarrow \forall x :: \tau_1.\ \mathit{appCost}(g, [\![x]\!]^{¢}_{\emptyset}) \triangleright ([\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ [\![x]\!]^{¢}_{\emptyset}) = [\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ ([\![g]\!]^{¢}_{\emptyset}\ ¢\ [\![x]\!]^{¢}_{\emptyset})$$

$$\Rightarrow \forall x :: \tau_1.\ \mathit{appCost}(g, [\![x]\!]^{¢}_{\emptyset}) \triangleright [\![\mathtt{f}\ x]\!]^{¢}_{\emptyset} = [\![\mathtt{f}\ (g\ x)]\!]^{¢}_{\emptyset}$$

Now, What about our Example $\mathtt{f} :: \alpha \to \mathsf{Nat}$?

Now:

$$\forall R \in \mathsf{Rel}, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R).\ [\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{x} = [\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{y}$$

$$\Rightarrow\ \forall \mathbf{x} \in [\![\tau_1]\!]^{¢}_{\emptyset}.\ [\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ (appCost(g, \mathbf{x}) \triangleright \mathbf{x}) = [\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ ([\![g]\!]^{¢}_{\emptyset}\ ¢\ \mathbf{x})$$

$$\Rightarrow\ \forall x :: \tau_1.\ appCost(g, [\![x]\!]^{¢}_{\emptyset}) \triangleright ([\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ [\![x]\!]^{¢}_{\emptyset}) = [\![\mathtt{f}]\!]^{¢}_{\emptyset}\ ¢\ ([\![g]\!]^{¢}_{\emptyset}\ ¢\ [\![x]\!]^{¢}_{\emptyset})$$

$$\Rightarrow\ \forall x :: \tau_1.\ appCost(g, [\![x]\!]^{¢}_{\emptyset}) \triangleright [\![\mathtt{f}\ x]\!]^{¢}_{\emptyset} = [\![\mathtt{f}\ (g\ x)]\!]^{¢}_{\emptyset}$$

Hence:

$$\mathtt{f}\ x \sqsubset \mathtt{f}\ (g\ x)$$

Let's Try another Example, $\mathtt{f} :: \alpha \to \alpha$

$$\forall \rho.\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha \to \alpha, \rho}$$

Let's Try another Example, $\mathtt{f} :: \alpha \to \alpha$

$$\begin{array}{ll} & \forall \rho.\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha \to \alpha, \rho} \\ \Rightarrow & \forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\alpha, \rho}.\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\alpha, \rho} \end{array}$$

Let's Try another Example, $\texttt{f} :: \alpha \to \alpha$

$$
\begin{aligned}
& \forall \rho.\ (\llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}, \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha\to\alpha,\rho} \\
\Rightarrow\ & \forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\alpha,\rho}.\ (\llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x}, \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\alpha,\rho} \\
\Rightarrow\ & \forall g :: \tau_1 \to \tau_2, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R^g). \\
& \quad (\llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x}, \llbracket \texttt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}) \in \mathcal{C}(R^g)
\end{aligned}
$$

Let's Try another Example, $\mathtt{f} :: \alpha \to \alpha$

$$
\begin{aligned}
& \forall \rho.\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha\to\alpha,\rho} \\
\Rightarrow\ & \forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\alpha,\rho}.\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\alpha,\rho} \\
\Rightarrow\ & \forall g :: \tau_1 \to \tau_2, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R^g). \\
& \quad (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}) \in \mathcal{C}(R^g) \\
\Rightarrow\ & \forall g :: \tau_1 \to \tau_2, \mathbf{x} \in \llbracket \tau_1 \rrbracket^{¢}_{\emptyset}. \\
& \quad (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ (appCost(g, \mathbf{x}) \rhd \mathbf{x}), \\
& \quad\ \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ (\llbracket g \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x})) \in \mathcal{C}(R^g)
\end{aligned}
$$

Let's Try another Example, $\mathtt{f} :: \alpha \to \alpha$

$$
\begin{aligned}
& \forall \rho.\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}) \in \Delta^{¢}_{\alpha\to\alpha,\rho} \\
\Rightarrow\ & \forall \rho, (\mathbf{x}, \mathbf{y}) \in \Delta^{¢}_{\alpha,\rho}.\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}) \in \Delta^{¢}_{\alpha,\rho} \\
\Rightarrow\ & \forall g :: \tau_1 \to \tau_2, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R^g). \\
& \quad (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x}, \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{y}) \in \mathcal{C}(R^g) \\
\Rightarrow\ & \forall g :: \tau_1 \to \tau_2, \mathbf{x} \in \llbracket \tau_1 \rrbracket^{¢}_{\emptyset}. \\
& \quad (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ (\mathit{appCost}(g, \mathbf{x}) \rhd \mathbf{x}), \\
& \quad\ \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ (\llbracket g \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x})) \in \mathcal{C}(R^g)
\end{aligned}
$$

Does this imply

$$\llbracket g \rrbracket^{¢}_{\emptyset}\ ¢\ (\llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x}) = \llbracket \mathtt{f} \rrbracket^{¢}_{\emptyset}\ ¢\ (\llbracket g \rrbracket^{¢}_{\emptyset}\ ¢\ \mathbf{x})\ ?$$

Let's Try another Example, $\mathtt{f} :: \alpha \to \alpha$

$$\forall g :: \tau_1 \to \tau_2, \mathbf{x} \in [\![\tau_1]\!]^{¢}_{\emptyset}.$$
$$(appCost(g, \mathbf{x}) \rhd ([\![\mathtt{f}]\!]^{¢}_{\emptyset} \, ¢ \, \mathbf{x}),$$
$$[\![\mathtt{f}]\!]^{¢}_{\emptyset} \, ¢ \, ([\![g]\!]^{¢}_{\emptyset} \, ¢ \, \mathbf{x})) \in \mathcal{C}(R^g)$$

Does this imply

$$[\![g]\!]^{¢}_{\emptyset} \, ¢ \, ([\![\mathtt{f}]\!]^{¢}_{\emptyset} \, ¢ \, \mathbf{x}) = [\![\mathtt{f}]\!]^{¢}_{\emptyset} \, ¢ \, ([\![g]\!]^{¢}_{\emptyset} \, ¢ \, \mathbf{x}) \, ?$$

Let's see:

$$\mathcal{C}(R^g) = \{(c \rhd appCost(g, \mathbf{x}) \rhd \mathbf{x}, c \rhd \mathbf{y}) \mid [\![g]\!]^{¢}_{\emptyset} \, ¢ \, \mathbf{x} = \mathbf{y}, c \in \mathbb{Z}\}$$

Let's Try another Example, $\mathtt{f} :: \alpha \to \alpha$

$$\forall g :: \tau_1 \to \tau_2, \mathbf{x} \in [\![\tau_1]\!]^{¢}_{\emptyset}.$$
$$(appCost(g, \mathbf{x}) \rhd ([\![\mathtt{f}]\!]^{¢}_{\emptyset} \mathbin{¢} \mathbf{x}),$$
$$[\![\mathtt{f}]\!]^{¢}_{\emptyset} \mathbin{¢} ([\![g]\!]^{¢}_{\emptyset} \mathbin{¢} \mathbf{x})) \in \mathcal{C}(R^{g})$$

Does this imply

$$[\![g]\!]^{¢}_{\emptyset} \mathbin{¢} ([\![\mathtt{f}]\!]^{¢}_{\emptyset} \mathbin{¢} \mathbf{x}) = [\![\mathtt{f}]\!]^{¢}_{\emptyset} \mathbin{¢} ([\![g]\!]^{¢}_{\emptyset} \mathbin{¢} \mathbf{x})\,?$$

Let's see:

$$\mathcal{C}(R^{g}) = \{(c \rhd appCost(g, \mathbf{x}') \rhd \mathbf{x}', c \rhd \mathbf{y}) \mid [\![g]\!]^{¢}_{\emptyset} \mathbin{¢} \mathbf{x}' = \mathbf{y}, c \in \mathbb{Z}\}$$

Let's Try another Example, $\mathtt{f} :: \alpha \to \alpha$

$$\forall g :: \tau_1 \to \tau_2, \mathbf{x} \in [\![\tau_1]\!]^{¢}_{\emptyset}.$$
$$(appCost(g, \mathbf{x}) \triangleright ([\![\mathtt{f}]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x}),$$
$$[\![\mathtt{f}]\!]^{¢}_{\emptyset} \; ¢ \; ([\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x})) \in \mathcal{C}(R^{g})$$

Let's see:

$$\mathcal{C}(R^{g}) = \{(c \triangleright appCost(g, \mathbf{x}') \triangleright \mathbf{x}', c \triangleright \mathbf{y}) \mid [\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x}' = \mathbf{y}, c \in \mathbb{Z}\}$$

Actually, the above only imply:

$$\forall \mathbf{x}. \exists \mathbf{x}'. \quad appCost(g, \mathbf{x}) \triangleright ([\![g]\!]^{¢}_{\emptyset} \; ¢ \; ([\![\mathtt{f}]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x}))$$
$$= appCost(g, \mathbf{x}') \triangleright ([\![\mathtt{f}]\!]^{¢}_{\emptyset} \; ¢ \; ([\![g]\!]^{¢}_{\emptyset} \; ¢ \; \mathbf{x}))$$

Let's Try another Example, f $:: \alpha \to \alpha$

Define:

$$R_x^g = \{(val([\![x]\!]_\emptyset^\cent), val([\![g]\!]_\emptyset^\cent \; \cent \; [\![x]\!]_\emptyset^\cent))\}$$

Let's Try another Example, $\texttt{f} :: \alpha \to \alpha$

Define:

$$R_x^g = \{(val(\llbracket x \rrbracket_\emptyset^¢), val(\llbracket g \rrbracket_\emptyset^¢ \ ¢ \ \llbracket x \rrbracket_\emptyset^¢))\}$$

Then:

$$\begin{aligned}\mathcal{C}(R_x^g) = \{(& c \triangleright appCost(g, \llbracket x \rrbracket_\emptyset^¢) \triangleright \llbracket x \rrbracket_\emptyset^¢, \\ & c \triangleright (\llbracket g \rrbracket_\emptyset^¢ \ ¢ \ \llbracket x \rrbracket_\emptyset^¢)) \mid c \in \mathbb{Z}\}\end{aligned}$$

Let's Try another Example, $\texttt{f} :: \alpha \to \alpha$

Define:

$$R_x^g = \{(val(\llbracket x \rrbracket_\emptyset^{¢}), val(\llbracket g \rrbracket_\emptyset^{¢} \; ¢ \; \llbracket x \rrbracket_\emptyset^{¢}))\}$$

Then:

$$\mathcal{C}(R_x^g) = \{(c \triangleright appCost(g, \llbracket x \rrbracket_\emptyset^{¢}) \triangleright \llbracket x \rrbracket_\emptyset^{¢}, \\ c \triangleright (\llbracket g \rrbracket_\emptyset^{¢} \; ¢ \; \llbracket x \rrbracket_\emptyset^{¢})) \mid c \in \mathbb{Z}\}$$

Thus:

$$\ldots$$

$$\Rightarrow \; \forall g :: \tau_1 \to \tau_2, x :: \tau_1, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R_x^g). \\ (\llbracket \texttt{f} \rrbracket_\emptyset^{¢} \; ¢ \; \mathbf{x}, \llbracket \texttt{f} \rrbracket_\emptyset^{¢} \; ¢ \; \mathbf{y}) \in \mathcal{C}(R_x^g)$$

$$\Rightarrow \; \forall g :: \tau_1 \to \tau_2, x :: \tau_1. \\ (\llbracket \texttt{f} \rrbracket_\emptyset^{¢} \; ¢ \; (appCost(g, \llbracket x \rrbracket_\emptyset^{¢}) \triangleright \llbracket x \rrbracket_\emptyset^{¢}), \\ \llbracket \texttt{f} \rrbracket_\emptyset^{¢} \; ¢ \; (\llbracket g \rrbracket_\emptyset^{¢} \; ¢ \; \llbracket x \rrbracket_\emptyset^{¢})) \in \mathcal{C}(R_x^g)$$

Let's Try another Example, $\texttt{f} :: \alpha \to \alpha$

Then:

$$\mathcal{C}(R_x^g) = \{(c \rhd appCost(g, [\![x]\!]^{¢}_{\emptyset}) \rhd [\![x]\!]^{¢}_{\emptyset}, \\ c \rhd ([\![g]\!]^{¢}_{\emptyset} \text{ ¢ } [\![x]\!]^{¢}_{\emptyset})) \mid c \in \mathbb{Z}\}$$

Thus:

$$\ldots$$

$$\Rightarrow \forall g :: \tau_1 \to \tau_2, x :: \tau_1, (\mathbf{x}, \mathbf{y}) \in \mathcal{C}(R_x^g). \\ ([\![\texttt{f}]\!]^{¢}_{\emptyset} \text{ ¢ } \mathbf{x}, [\![\texttt{f}]\!]^{¢}_{\emptyset} \text{ ¢ } \mathbf{y}) \in \mathcal{C}(R_x^g)$$

$$\Rightarrow \forall g :: \tau_1 \to \tau_2, x :: \tau_1. \\ ([\![\texttt{f}]\!]^{¢}_{\emptyset} \text{ ¢ } (appCost(g, [\![x]\!]^{¢}_{\emptyset}) \rhd [\![x]\!]^{¢}_{\emptyset}), \\ [\![\texttt{f}]\!]^{¢}_{\emptyset} \text{ ¢ } ([\![g]\!]^{¢}_{\emptyset} \text{ ¢ } [\![x]\!]^{¢}_{\emptyset})) \in \mathcal{C}(R_x^g)$$

$$\Rightarrow \forall g :: \tau_1 \to \tau_2, x :: \tau_1. \\ [\![g]\!]^{¢}_{\emptyset} \text{ ¢ } ([\![\texttt{f}]\!]^{¢}_{\emptyset} \text{ ¢ } [\![x]\!]^{¢}_{\emptyset}) = [\![\texttt{f}]\!]^{¢}_{\emptyset} \text{ ¢ } ([\![g]\!]^{¢}_{\emptyset} \text{ ¢ } [\![x]\!]^{¢}_{\emptyset})$$

Other Examples

For $\texttt{f} :: \alpha \to \alpha \to \alpha$,

$$g\ (\texttt{f}\ x\ y) \sqsubseteq \texttt{f}\ (g\ x)\ (g\ y)$$

Other Examples

For $\texttt{f} :: \alpha \to \alpha \to \alpha$,

$$g\ (\texttt{f}\ x\ y) \sqsubseteq \texttt{f}\ (g\ x)\ (g\ y)$$

For $\texttt{f} :: \alpha \to (\alpha, \alpha)$,

$$\texttt{mapPair}\ (g, g)\ (\texttt{f}\ x) \sqsupseteq \texttt{f}\ (g\ x)$$

Other Examples

For $\mathtt{f} :: \alpha \to \alpha \to \alpha$,

$$g\ (\mathtt{f}\ x\ y) \sqsubseteq \mathtt{f}\ (g\ x)\ (g\ y)$$

For $\mathtt{f} :: \alpha \to (\alpha, \alpha)$,

$$\mathtt{mapPair}\ (g, g)\ (\mathtt{f}\ x) \sqsupseteq \mathtt{f}\ (g\ x)$$

For $\mathtt{f} :: [\alpha] \to \mathsf{Nat}$,

$$\mathtt{f}\ l \sqsubseteq \mathtt{f}\ (\mathtt{map}\ g\ l)$$

Other Examples

For $\texttt{f} :: \alpha \to \alpha \to \alpha$,

$$g\ (\texttt{f}\ x\ y) \sqsubseteq \texttt{f}\ (g\ x)\ (g\ y)$$

For $\texttt{f} :: \alpha \to (\alpha, \alpha)$,

$$\texttt{mapPair}\ (g, g)\ (\texttt{f}\ x) \sqsupseteq \texttt{f}\ (g\ x)$$

For $\texttt{f} :: [\alpha] \to \mathsf{Nat}$,

$$\texttt{f}\ l \sqsubseteq \texttt{f}\ (\texttt{map}\ g\ l)$$

For $\texttt{f} :: [\alpha] \to [\alpha]$, get conditional statements about relative efficiency of $\texttt{map}\ g\ (\texttt{f}\ l)$ and $\texttt{f}\ (\texttt{map}\ g\ l)$.

Conclusion

We did:

- develop of “cost-aware” logical relation and its parametricity theorem,

Conclusion

We did:

- develop of “cost-aware” logical relation and its parametricity theorem,
- realize that more effort has to go into the actual deriving of concrete free theorems,

Conclusion

We did:

- develop of "cost-aware" logical relation and its parametricity theorem,
- realize that more effort has to go into the actual deriving of concrete free theorems,
- establish a number of building blocks/heuristics for that.

Conclusion

We did:

- develop of “cost-aware” logical relation and its parametricity theorem,
- realize that more effort has to go into the actual deriving of concrete free theorems,
- establish a number of building blocks/heuristics for that.

Further plans:

- mechanize derivation of cost-aware statements,

Conclusion

We did:

- develop of "cost-aware" logical relation and its parametricity theorem,
- realize that more effort has to go into the actual deriving of concrete free theorems,
- establish a number of building blocks/heuristics for that.

Further plans:

- mechanize derivation of cost-aware statements,
- apply to automatic program transformations,

Conclusion

We did:

- develop of "cost-aware" logical relation and its parametricity theorem,
- realize that more effort has to go into the actual deriving of concrete free theorems,
- establish a number of building blocks/heuristics for that.

Further plans:

- mechanize derivation of cost-aware statements,
- apply to automatic program transformations,
- extend to full recursion, other language features,

Conclusion

We did:

- develop of “cost-aware” logical relation and its parametricity theorem,
- realize that more effort has to go into the actual deriving of concrete free theorems,
- establish a number of building blocks/heuristics for that.

Further plans:

- mechanize derivation of cost-aware statements,
- apply to automatic program transformations,
- extend to full recursion, other language features,
- investigate call-by-name/call-by-need,

Conclusion

We did:

- develop of “cost-aware” logical relation and its parametricity theorem,
- realize that more effort has to go into the actual deriving of concrete free theorems,
- establish a number of building blocks/heuristics for that.

Further plans:

- mechanize derivation of cost-aware statements,
- apply to automatic program transformations,
- extend to full recursion, other language features,
- investigate call-by-name/call-by-need,
- use more realistic cost measures?

References I

J.-P. Bernardy, P. Jansson, and K. Claessen.
Testing polymorphic properties.
In *European Symposium on Programming, Proceedings*, volume 6012 of *LNCS*, pages 125–144. Springer-Verlag, 2010.

B. Bjerner and S. Holmström.
A compositional approach to time analysis of first order lazy functional programs.
In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 157–165. ACM Press, 1989.

N.A. Day, J. Launchbury, and J. Lewis.
Logical abstractions in Haskell.
In *Haskell Workshop, Proceedings*. Technical Report UU-CS-1999-28, Utrecht University, 1999.

References II

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.

Y. Liu and G. Gómez.
Automatic accurate cost-bound analysis for high-level languages.
IEEE Transactions on Computers, 50(12):1295–1309, 2001.

J.C. Reynolds.
Types, abstraction and parametric polymorphism.
In *Information Processing, Proceedings*, pages 513–523. Elsevier, 1983.

References III

M. Rosendahl.
Automatic complexity analysis.
In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 144–156. ACM Press, 1989.

D. Sands.
A naïve time analysis and its theory of cost equivalence.
Journal of Logic and Computation, 5(4):495–541, 1995.

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

References IV

- A. Takano and E. Meijer.
  Shortcut deforestation in calculational form.
  In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 306–313. ACM Press, 1995.

- J. Voigtländer.
  Much ado about two: A pearl on parallel prefix computation.
  In *Principles of Programming Languages, Proceedings*, pages 29–35. ACM Press, 2008.

- J. Voigtländer.
  Bidirectionalization for free!
  In *Principles of Programming Languages, Proceedings*, pages 165–176. ACM Press, 2009.

References V

P. Wadler.
Strictness analysis aids time analysis.
In *Principles of Programming Languages, Proceedings*, pages 119–132. ACM Press, 1988.

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

A "Real" Example: Fusion [Gill et al., FPCA'93]

Extensional free theorem:

For every $\texttt{f} :: (\tau \to \alpha \to \alpha) \to \alpha \to \alpha$,

$$\texttt{foldr}\ k\ z\ (\texttt{f}\ (:)\ [\,]) = \texttt{f}\ k\ z$$

The whole point of fusion:

We expect,

$$\texttt{foldr}\ k\ z\ (\texttt{f}\ (:)\ [\,]) \sqsupseteq \texttt{f}\ k\ z$$

A counterexample (in call-by-value setting):

$$\texttt{f} :: (\mathsf{Nat} \to \alpha \to \alpha) \to \alpha \to \alpha$$
$$\texttt{f}\ k\ z = \textbf{case}\ [k\ 5\ z]\ \textbf{of}\ \{[\,] \to z; x : xs \to z\}$$