Free Theorems — The Basics

Janis Voigtländer

Technische Universität Dresden

January 6, 2006

Outline

Haskell Example:

$$
\begin{array}{l}
\mathit{filter} :: \forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha] \\
\mathit{filter}\ p\ []\qquad\quad = [] \\
\mathit{filter}\ p\ (x : \mathit{xs}) = \mathit{if}\ \ p\ x\ \mathit{then}\ x : \mathit{filter}\ \ p\ \mathit{xs} \\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{else}\ \ \mathit{filter}\ \ p\ \mathit{xs}
\end{array}
$$

Haskell Example:

$$\begin{array}{l} \mathit{filter} :: \forall\alpha.\ (\alpha \rightarrow \mathit{Bool}) \rightarrow [\alpha] \rightarrow [\alpha] \\ \mathit{filter}\ p\ []\qquad\quad = [] \\ \mathit{filter}\ p\ (x : xs) = \mathit{if}\ \ p\ x\ \ \mathit{then}\ x : \mathit{filter}\ \ p\ xs \\ \qquad\qquad\qquad\qquad\qquad\ \ \mathit{else}\ \ \mathit{filter}\ \ p\ xs \end{array}$$

Claim:

$$\boxed{\mathit{filter}\ \ p\ (\mathit{map}\ \ h\ \ l) = \mathit{map}\ \ h\ (\mathit{filter}\ (p \circ h)\ \ l) \qquad (1)}$$

Can be proved by induction on l, using the definition of *filter*.

Haskell Example: Theorems for free! [Wadler 1989]

$$\boxed{\textit{filter} :: \forall\alpha.\ (\alpha \rightarrow \textit{Bool}) \rightarrow [\alpha] \rightarrow [\alpha]}$$

Claim:

$$\boxed{\textit{filter}\ p\ (\textit{map}\ h\ l) = \textit{map}\ h\ (\textit{filter}\ (p \circ h)\ l) \qquad (1)}$$

Can be derived from the parametric polymorphic type of *filter*!

Haskell Example: Theorems for free! [Wadler 1989]

$$\boxed{\textit{filter} :: \forall\alpha.\ (\alpha \to \textit{Bool}) \to [\alpha] \to [\alpha]}$$

Claim:

$$\boxed{\textit{filter}\ p\ (\textit{map}\ h\ l) = \textit{map}\ h\ (\textit{filter}\ (p \circ h)\ l) \qquad (1)}$$

Can be derived from the parametric polymorphic type of *filter*!

Where is the magic? Where is the induction?

Parametric Polymorphism, Intuitively

- $\mathit{filter} :: \forall \alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$ must work uniformly for every instantiation of α.

Parametric Polymorphism, Intuitively

- $\mathit{filter} :: \forall \alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$ must work uniformly for every instantiation of α.
- The output list can only contain elements from the input list l.

Parametric Polymorphism, Intuitively

- $filter :: \forall\alpha.\ (\alpha \rightarrow Bool) \rightarrow [\alpha] \rightarrow [\alpha]$ must work uniformly for every instantiation of α.
- The output list can only contain elements from the input list l.
- Which, and in which order/multiplicity, can only be decided based on l and the input predicate p.

Parametric Polymorphism, Intuitively

- $\mathit{filter} :: \forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$ must work uniformly for every instantiation of α.
- The output list can only contain elements from the input list l.
- Which, and in which order/multiplicity, can only be decided based on l and the input predicate p.
- The only means for this decision are to inspect the length of l and to check the outcome of p on its elements.

Parametric Polymorphism, Intuitively

- $\mathit{filter} :: \forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$ must work uniformly for every instantiation of α.
- The output list can only contain elements from the input list l.
- Which, and in which order/multiplicity, can only be decided based on l and the input predicate p.
- The only means for this decision are to inspect the length of l and to check the outcome of p on its elements.
- The lists $(\mathit{map}\ h\ l)$ and l always have equal length.

Parametric Polymorphism, Intuitively

- $\mathit{filter} :: \forall \alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$ must work uniformly for every instantiation of α.
- The output list can only contain elements from the input list l.
- Which, and in which order/multiplicity, can only be decided based on l and the input predicate p.
- The only means for this decision are to inspect the length of l and to check the outcome of p on its elements.
- The lists $(\mathit{map}\ h\ l)$ and l always have equal length.
- Applying p to an element of $(\mathit{map}\ h\ l)$ always has the same outcome as applying $(p \circ h)$ to the corresponding element of l.

Parametric Polymorphism, Intuitively

- $\mathit{filter} :: \forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$ must work uniformly for every instantiation of α.
- The output list can only contain elements from the input list l.
- Which, and in which order/multiplicity, can only be decided based on l and the input predicate p.
- The only means for this decision are to inspect the length of l and to check the outcome of p on its elements.
- The lists $(\mathit{map}\ h\ l)$ and l always have equal length.
- Applying p to an element of $(\mathit{map}\ h\ l)$ always has the same outcome as applying $(p \circ h)$ to the corresponding element of l.
- filter with p always chooses "the same" elements from $(\mathit{map}\ h\ l)$ for output as does filter with $(p \circ h)$ from l,

Parametric Polymorphism, Intuitively

- $\mathit{filter} :: \forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$ must work uniformly for every instantiation of α.
- The output list can only contain elements from the input list l.
- Which, and in which order/multiplicity, can only be decided based on l and the input predicate p.
- The only means for this decision are to inspect the length of l and to check the outcome of p on its elements.
- The lists $(\mathit{map}\ h\ l)$ and l always have equal length.
- Applying p to an element of $(\mathit{map}\ h\ l)$ always has the same outcome as applying $(p \circ h)$ to the corresponding element of l.
- *filter* with p always chooses "the same" elements from $(\mathit{map}\ h\ l)$ for output as does *filter* with $(p \circ h)$ from l, except that it outputs their images under h.

Parametric Polymorphism, Intuitively

- $\mathit{filter} :: \forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$ must work uniformly for every instantiation of α.
- The output list can only contain elements from the input list l.
- Which, and in which order/multiplicity, can only be decided based on l and the input predicate p.
- The only means for this decision are to inspect the length of l and to check the outcome of p on its elements.
- The lists $(\mathit{map}\ h\ l)$ and l always have equal length.
- Applying p to an element of $(\mathit{map}\ h\ l)$ always has the same outcome as applying $(p \circ h)$ to the corresponding element of l.
- *filter* with p always chooses "the same" elements from $(\mathit{map}\ h\ l)$ for output as does *filter* with $(p \circ h)$ from l, except that it outputs their images under h.
- $(\mathit{filter}\ p\ (\mathit{map}\ h\ l))$ is equivalent to $(\mathit{map}\ h\ (\mathit{filter}\ (p \circ h)\ l))$.

Parametric Polymorphism, Intuitively

- $\mathit{filter} :: \forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$ must work uniformly for every instantiation of α.
- The output list can only contain elements from the input list l.
- Which, and in which order/multiplicity, can only be decided based on l and the input predicate p.
- The only means for this decision are to inspect the length of l and to check the outcome of p on its elements.
- The lists $(\mathit{map}\ h\ l)$ and l always have equal length.
- Applying p to an element of $(\mathit{map}\ h\ l)$ always has the same outcome as applying $(p \circ h)$ to the corresponding element of l.
- *filter* with p always chooses "the same" elements from $(\mathit{map}\ h\ l)$ for output as does *filter* with $(p \circ h)$ from l, except that it outputs their images under h.
- $(\mathit{filter}\ p\ (\mathit{map}\ h\ l))$ is equivalent to $(\mathit{map}\ h\ (\mathit{filter}\ (p \circ h)\ l))$.
- That is what we wanted to prove!

Parametric Polymorphism, More Formally

Question: What functions are in $\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$?

Approach: Give denotations of types as sets.

Parametric Polymorphism, More Formally

Question: What functions are in $\forall \alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$?

Approach: Give denotations of types as sets.

$$
\begin{array}{lcll}
[\![\mathit{Bool}]\!]_\theta & = & \{\mathit{True}, \mathit{False}\} & = \mathbb{B} \\
[\![\mathit{Nat}]\!]_\theta & = & \{0, 1, 2, \dots\} & = \mathbb{N}
\end{array}
$$

Parametric Polymorphism, More Formally

Question: What functions are in $\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$?

Approach: Give denotations of types as sets.

$$
\begin{array}{lcll}
[\![\mathit{Bool}]\!]_\theta & = & \{\mathit{True}, \mathit{False}\} & = \ \mathbb{B} \\
[\![\mathit{Nat}]\!]_\theta & = & \{0, 1, 2, \dots\} & = \ \mathbb{N} \\
[\![(\tau_1, \tau_2)]\!]_\theta & = & [\![\tau_1]\!]_\theta \times [\![\tau_2]\!]_\theta & \\
[\![[\tau]]\!]_\theta & = & \{[x_1, \dots, x_n] \mid n \geq 0, x_i \in [\![\tau]\!]_\theta\} &
\end{array}
$$

Parametric Polymorphism, More Formally

Question: What functions are in $\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$?

Approach: Give denotations of types as sets.

$$
\begin{array}{lcll}
[\![\mathit{Bool}]\!]_\theta & = & \{\mathit{True}, \mathit{False}\} & = \ \mathbb{B} \\
[\![\mathit{Nat}]\!]_\theta & = & \{0, 1, 2, \dots\} & = \ \mathbb{N} \\
[\![(\tau_1, \tau_2)]\!]_\theta & = & [\![\tau_1]\!]_\theta \times [\![\tau_2]\!]_\theta & \\
[\![[\tau]]\!]_\theta & = & \{[x_1, \dots, x_n] \mid n \geq 0, x_i \in [\![\tau]\!]_\theta\} & \\
[\![\tau_1 \to \tau_2]\!]_\theta & = & \{f : [\![\tau_1]\!]_\theta \to [\![\tau_2]\!]_\theta\} &
\end{array}
$$

Parametric Polymorphism, More Formally

Question: What functions are in $\forall\alpha.\ (\alpha \rightarrow \mathit{Bool}) \rightarrow [\alpha] \rightarrow [\alpha]$?

Approach: Give denotations of types as sets.

$$
\begin{array}{lcll}
[\![\mathit{Bool}]\!]_\theta & = & \{\mathit{True}, \mathit{False}\} & = \ \mathbb{B} \\
[\![\mathit{Nat}]\!]_\theta & = & \{0, 1, 2, \dots\} & = \ \mathbb{N} \\
[\![(\tau_1, \tau_2)]\!]_\theta & = & [\![\tau_1]\!]_\theta \times [\![\tau_2]\!]_\theta & \\
[\![[\tau]]\!]_\theta & = & \{[x_1, \dots, x_n] \mid n \geq 0, x_i \in [\![\tau]\!]_\theta\} & \\
[\![\tau_1 \rightarrow \tau_2]\!]_\theta & = & \{f : [\![\tau_1]\!]_\theta \rightarrow [\![\tau_2]\!]_\theta\} & \\
[\![\forall\alpha.\ \tau]\!]_\theta & = & ? &
\end{array}
$$

Parametric Polymorphism, More Formally

Question: What functions are in $\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$?

Approach: Give denotations of types as sets.

$$
\begin{array}{lcll}
[\![\mathit{Bool}]\!]_\theta & = & \{\mathit{True}, \mathit{False}\} & = \mathbb{B} \\
[\![\mathit{Nat}]\!]_\theta & = & \{0, 1, 2, \ldots\} & = \mathbb{N} \\
[\![(\tau_1, \tau_2)]\!]_\theta & = & [\![\tau_1]\!]_\theta \times [\![\tau_2]\!]_\theta & \\
[\![[\tau]]\!]_\theta & = & \{[x_1, \ldots, x_n] \mid n \geq 0, x_i \in [\![\tau]\!]_\theta\} & \\
[\![\tau_1 \to \tau_2]\!]_\theta & = & \{f : [\![\tau_1]\!]_\theta \to [\![\tau_2]\!]_\theta\} & \\
[\![\forall\alpha.\ \tau]\!]_\theta & = & ? &
\end{array}
$$

- $g \in [\![\forall\alpha.\ \tau]\!]_\theta$ should be a "collection" of values: for every type τ', there is an instance of type $\tau[\tau'/\alpha]$.

Parametric Polymorphism, More Formally

Question: What functions are in $\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$?

Approach: Give denotations of types as sets.

$$
\begin{array}{lcll}
[\![\mathit{Bool}]\!]_\theta & = & \{\mathit{True}, \mathit{False}\} & = \ \mathbb{B} \\
[\![\mathit{Nat}]\!]_\theta & = & \{0, 1, 2, \ldots\} & = \ \mathbb{N} \\
[\![(\tau_1, \tau_2)]\!]_\theta & = & [\![\tau_1]\!]_\theta \times [\![\tau_2]\!]_\theta & \\
[\![[\tau]]\!]_\theta & = & \{[x_1, \ldots, x_n] \mid n \geq 0, x_i \in [\![\tau]\!]_\theta\} & \\
[\![\tau_1 \to \tau_2]\!]_\theta & = & \{f : [\![\tau_1]\!]_\theta \to [\![\tau_2]\!]_\theta\} & \\
[\![\forall\alpha.\ \tau]\!]_\theta & = & ? &
\end{array}
$$

- $g \in [\![\forall\alpha.\ \tau]\!]_\theta$ should be a "collection" of values: for every type τ', there is an instance of type $\tau[\tau'/\alpha]$.
- $[\![\forall\alpha.\ \tau]\!]_\theta = \{g : \mathit{Set} \to \mathit{Value} \mid \forall S \in \mathit{Set}.\ (g\ S) \in [\![\tau]\!]_{\theta[\alpha \mapsto S]}\}$ is maybe a good start, together with $[\![\alpha]\!]_\theta = \theta(\alpha)$.

Parametric Polymorphism, More Formally

Question: What functions are in $\forall \alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$?

Approach: Give denotations of types as sets.

$$
\begin{array}{lcll}
[\![\mathit{Bool}]\!]_\theta & = & \{\mathit{True}, \mathit{False}\} & = \mathbb{B} \\
[\![\mathit{Nat}]\!]_\theta & = & \{0, 1, 2, \dots\} & = \mathbb{N} \\
[\![(\tau_1, \tau_2)]\!]_\theta & = & [\![\tau_1]\!]_\theta \times [\![\tau_2]\!]_\theta & \\
[\![[\tau]]\!]_\theta & = & \{[x_1, \dots, x_n] \mid n \geq 0, x_i \in [\![\tau]\!]_\theta\} & \\
[\![\tau_1 \to \tau_2]\!]_\theta & = & \{f : [\![\tau_1]\!]_\theta \to [\![\tau_2]\!]_\theta\} & \\
[\![\forall \alpha.\ \tau]\!]_\theta & = & ? &
\end{array}
$$

- $g \in [\![\forall \alpha.\ \tau]\!]_\theta$ should be a "collection" of values: for every type τ', there is an instance of type $\tau[\tau'/\alpha]$.
- $[\![\forall \alpha.\ \tau]\!]_\theta = \{g : \mathit{Set} \to \mathit{Value} \mid \forall S \in \mathit{Set}.\ (g\ S) \in [\![\tau]\!]_{\theta[\alpha \mapsto S]}\}$ is maybe a good start, together with $[\![\alpha]\!]_\theta = \theta(\alpha)$.
- But this may contain "ad-hoc" polymorphic functions!

Unwanted Ad-Hoc Polymorphism: Example

- With the proposed definition,
  $[\![\forall\alpha.\ (\alpha,\alpha) \to \alpha]\!]_\emptyset = \{g \mid \forall S \in Set.\ (g\ S) : S \times S \to S\}$.

Unwanted Ad-Hoc Polymorphism: Example

- With the proposed definition,
  $[\![\forall\alpha.\ (\alpha,\alpha) \to \alpha]\!]_\emptyset = \{g \mid \forall S \in Set.\ (g\ S) : S \times S \to S\}$.
- But this also allows

$$\begin{array}{rcl} g\ \mathbb{B}\ (x,y) & = & not\ x \\ g\ \mathbb{N}\ (x,y) & = & y+1\,, \end{array}$$

  which is not possible in Haskell at type $\forall\alpha.\ (\alpha,\alpha) \to \alpha$.

Unwanted Ad-Hoc Polymorphism: Example

- With the proposed definition, $[\![\forall\alpha.\ (\alpha,\alpha)\to\alpha]\!]_\emptyset = \{g \mid \forall S \in \mathit{Set}.\ (g\ S) : S \times S \to S\}$.
- But this also allows

$$\begin{aligned} g\ \mathbb{B}\ (x,y) &= \mathit{not}\ x \\ g\ \mathbb{N}\ (x,y) &= y+1\,, \end{aligned}$$

  which is not possible in Haskell at type $\forall\alpha.\ (\alpha,\alpha)\to\alpha$.
- To prevent this, compare/relate

$$\begin{aligned} (g\ \mathbb{B}) &: \mathbb{B}\times\mathbb{B}\to\mathbb{B} \quad \text{and} \\ (g\ \mathbb{N}) &: \mathbb{N}\times\mathbb{N}\to\mathbb{N}\,, \end{aligned}$$

  ensuring that they “behave identically”.
  But how?

Key Idea [Reynolds 1983]

Use relations to tie instances together.

Key Idea [Reynolds 1983]

Use relations to tie instances together.

In the example:

- Choose an $\mathcal{R} \subseteq \mathbb{B} \times \mathbb{N}$.

Key Idea [Reynolds 1983]

Use relations to tie instances together.

In the example:

- Choose an $\mathcal{R} \subseteq \mathbb{B} \times \mathbb{N}$.
- Say that $(x_1, y_1) \in \mathbb{B} \times \mathbb{B}$ and $(x_2, y_2) \in \mathbb{N} \times \mathbb{N}$ are related if $(x_1, x_2) \in \mathcal{R}$ and $(y_1, y_2) \in \mathcal{R}$.

Key Idea [Reynolds 1983]

Use relations to tie instances together.

In the example:

- Choose an $\mathcal{R} \subseteq \mathbb{B} \times \mathbb{N}$.
- Say that $(x_1, y_1) \in \mathbb{B} \times \mathbb{B}$ and $(x_2, y_2) \in \mathbb{N} \times \mathbb{N}$ are related if $(x_1, x_2) \in \mathcal{R}$ and $(y_1, y_2) \in \mathcal{R}$.
- Say that $f_1 : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ and $f_2 : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ are related if they map related arguments to related results.

Key Idea [Reynolds 1983]

Use relations to tie instances together.

In the example:

- Choose an $\mathcal{R} \subseteq \mathbb{B} \times \mathbb{N}$.
- Say that $(x_1, y_1) \in \mathbb{B} \times \mathbb{B}$ and $(x_2, y_2) \in \mathbb{N} \times \mathbb{N}$ are related if $(x_1, x_2) \in \mathcal{R}$ and $(y_1, y_2) \in \mathcal{R}$.
- Say that $f_1 : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ and $f_2 : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ are related if they map related arguments to related results.
- Then $(g\ \mathbb{B})$ and $(g\ \mathbb{N})$ with

$$\begin{aligned} g\ \mathbb{B}\ (x, y) &= \mathit{not}\ x \\ g\ \mathbb{N}\ (x, y) &= y + 1 \end{aligned}$$

  are not related if we choose, e.g., $\mathcal{R} = \{(\mathit{True}, 1)\}$.

Key Idea [Reynolds 1983]

Use relations to tie instances together.

In the example:

- Choose an $\mathcal{R} \subseteq \mathbb{B} \times \mathbb{N}$.
- Say that $(x_1, y_1) \in \mathbb{B} \times \mathbb{B}$ and $(x_2, y_2) \in \mathbb{N} \times \mathbb{N}$ are related if $(x_1, x_2) \in \mathcal{R}$ and $(y_1, y_2) \in \mathcal{R}$.
- Say that $f_1 : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ and $f_2 : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ are related if they map related arguments to related results.
- Then $(g\ \mathbb{B})$ and $(g\ \mathbb{N})$ with

$$\begin{array}{rcl} g\ \mathbb{B}\ (x, y) & = & \mathit{not}\ x \\ g\ \mathbb{N}\ (x, y) & = & y + 1 \end{array}$$

  are not related if we choose, e.g., $\mathcal{R} = \{(\mathit{True}, 1)\}$.

Reynolds: $g \in [\![\forall \alpha.\ \tau]\!]_\theta$ only if for every $S_1, S_2, \mathcal{R} \subseteq S_1 \times S_2$, $(g\ S_1)$ is related to $(g\ S_2)$ by the "propagation" of $\mathcal{R}$ according to τ.

Polymorphic Lambda Calculus [Girard 1972, Reynolds 1974]

Types: $\tau := \alpha \mid \tau \to \tau \mid \forall\alpha.\ \tau$

Terms: $t := x \mid \lambda x : \tau.\ t \mid t\ t \mid \Lambda\alpha.\ t \mid t\ \tau$

Polymorphic Lambda Calculus [Girard 1972, Reynolds 1974]

Types: $\tau := \alpha \mid \tau \to \tau \mid \forall \alpha.\ \tau$

Terms: $t := x \mid \lambda x : \tau.\ t \mid t\ t \mid \Lambda \alpha.\ t \mid t\ \tau$

$$\Gamma, x : \tau \vdash x : \tau$$

Polymorphic Lambda Calculus [Girard 1972, Reynolds 1974]

Types: $\tau := \alpha \mid \tau \to \tau \mid \forall \alpha.\ \tau$

Terms: $t := x \mid \lambda x : \tau.\ t \mid t\ t \mid \Lambda \alpha.\ t \mid t\ \tau$

$$\Gamma, x : \tau \vdash x : \tau$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1.\ t) : \tau_1 \to \tau_2}$$

Polymorphic Lambda Calculus [Girard 1972, Reynolds 1974]

Types: $\tau := \alpha \mid \tau \to \tau \mid \forall\alpha.\ \tau$

Terms: $t := x \mid \lambda x : \tau.\ t \mid t\ t \mid \Lambda\alpha.\ t \mid t\ \tau$

$$\Gamma, x : \tau \vdash x : \tau$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1.\ t) : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash t : \tau_1 \to \tau_2 \qquad \Gamma \vdash u : \tau_1}{\Gamma \vdash (t\ u) : \tau_2}$$

Polymorphic Lambda Calculus [Girard 1972, Reynolds 1974]

Types: $\tau := \alpha \mid \tau \to \tau \mid \forall\alpha.\ \tau$

Terms: $t := x \mid \lambda x : \tau.\ t \mid t\ t \mid \Lambda\alpha.\ t \mid t\ \tau$

$$\Gamma, x : \tau \vdash x : \tau$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1.\ t) : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash t : \tau_1 \to \tau_2 \qquad \Gamma \vdash u : \tau_1}{\Gamma \vdash (t\ u) : \tau_2}$$

$$\frac{\alpha, \Gamma \vdash t : \tau}{\Gamma \vdash (\Lambda\alpha.\ t) : \forall\alpha.\ \tau}$$

Polymorphic Lambda Calculus [Girard 1972, Reynolds 1974]

Types: $\tau := \alpha \mid \tau \to \tau \mid \forall \alpha.\ \tau$

Terms: $t := x \mid \lambda x : \tau.\ t \mid t\ t \mid \Lambda\alpha.\ t \mid t\ \tau$

$$\Gamma, x : \tau \vdash x : \tau$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1.\ t) : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash t : \tau_1 \to \tau_2 \qquad \Gamma \vdash u : \tau_1}{\Gamma \vdash (t\ u) : \tau_2}$$

$$\frac{\alpha, \Gamma \vdash t : \tau}{\Gamma \vdash (\Lambda\alpha.\ t) : \forall \alpha.\ \tau}$$

$$\frac{\Gamma \vdash t : \forall \alpha.\ \tau}{\Gamma \vdash (t\ \tau') : \tau[\tau'/\alpha]}$$

Polymorphic Lambda Calculus [Girard 1972, Reynolds 1974]

Types: $\tau := \alpha \mid \tau \to \tau \mid \forall\alpha.\ \tau$

Terms: $t := x \mid \lambda x : \tau.\ t \mid t\ t \mid \Lambda\alpha.\ t \mid t\ \tau$

$$\Gamma, x : \tau \vdash x : \tau \qquad [\![x]\!]_{\theta,\sigma} = \sigma(x)$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1.\ t) : \tau_1 \to \tau_2} \qquad [\![\lambda x : \tau_1.\ t]\!]_{\theta,\sigma}\ a = [\![t]\!]_{\theta,\sigma[x \mapsto a]}$$

$$\frac{\Gamma \vdash t : \tau_1 \to \tau_2 \qquad \Gamma \vdash u : \tau_1}{\Gamma \vdash (t\ u) : \tau_2} \qquad [\![t\ u]\!]_{\theta,\sigma} = [\![t]\!]_{\theta,\sigma}\ [\![u]\!]_{\theta,\sigma}$$

$$\frac{\alpha, \Gamma \vdash t : \tau}{\Gamma \vdash (\Lambda\alpha.\ t) : \forall\alpha.\ \tau} \qquad [\![\Lambda\alpha.\ t]\!]_{\theta,\sigma}\ S = [\![t]\!]_{\theta[\alpha \mapsto S],\sigma}$$

$$\frac{\Gamma \vdash t : \forall\alpha.\ \tau}{\Gamma \vdash (t\ \tau') : \tau[\tau'/\alpha]} \qquad [\![t\ \tau']\!]_{\theta,\sigma} = [\![t]\!]_{\theta,\sigma}\ [\![\tau']\!]_{\theta}$$

Parametricity Theorem [Reynolds 1983, Wadler 1989]

Given τ and environments θ_1, θ_2, ρ with $\rho(\alpha) \subseteq \theta_1(\alpha) \times \theta_2(\alpha)$, define $\Delta_{\tau,\rho} \subseteq [\![\tau]\!]_{\theta_1} \times [\![\tau]\!]_{\theta_2}$ as follows:

Parametricity Theorem [Reynolds 1983, Wadler 1989]

Given τ and environments θ_1, θ_2, ρ with $\rho(\alpha) \subseteq \theta_1(\alpha) \times \theta_2(\alpha)$, define $\Delta_{\tau,\rho} \subseteq [\![\tau]\!]_{\theta_1} \times [\![\tau]\!]_{\theta_2}$ as follows:

$$\Delta_{\alpha,\rho} \quad = \quad \rho(\alpha)$$

Parametricity Theorem [Reynolds 1983, Wadler 1989]

Given τ and environments θ_1, θ_2, ρ with $\rho(\alpha) \subseteq \theta_1(\alpha) \times \theta_2(\alpha)$, define $\Delta_{\tau,\rho} \subseteq [\![\tau]\!]_{\theta_1} \times [\![\tau]\!]_{\theta_2}$ as follows:

$$
\begin{array}{lcl}
\Delta_{\alpha,\rho} & = & \rho(\alpha) \\
\Delta_{\tau_1 \to \tau_2,\rho} & = & \{(f_1, f_2) \mid \forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ (f_1\ a_1, f_2\ a_2) \in \Delta_{\tau_2,\rho}\}
\end{array}
$$

Parametricity Theorem [Reynolds 1983, Wadler 1989]

Given τ and environments θ_1, θ_2, ρ with $\rho(\alpha) \subseteq \theta_1(\alpha) \times \theta_2(\alpha)$, define $\Delta_{\tau,\rho} \subseteq [\![\tau]\!]_{\theta_1} \times [\![\tau]\!]_{\theta_2}$ as follows:

$$
\begin{array}{lcl}
\Delta_{\alpha,\rho} & = & \rho(\alpha) \\
\Delta_{\tau_1 \to \tau_2,\rho} & = & \{(f_1, f_2) \mid \forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ (f_1\ a_1, f_2\ a_2) \in \Delta_{\tau_2,\rho}\} \\
\Delta_{\forall \alpha.\,\tau,\rho} & = & \{(g_1, g_2) \mid \forall \mathcal{R} \subseteq S_1 \times S_2.\ (g_1\ S_1, g_2\ S_2) \in \Delta_{\tau,\rho[\alpha \mapsto \mathcal{R}]}\}
\end{array}
$$

Parametricity Theorem [Reynolds 1983, Wadler 1989]

Given τ and environments θ_1, θ_2, ρ with $\rho(\alpha) \subseteq \theta_1(\alpha) \times \theta_2(\alpha)$, define $\Delta_{\tau,\rho} \subseteq [\![\tau]\!]_{\theta_1} \times [\![\tau]\!]_{\theta_2}$ as follows:

$$\begin{array}{lcl}
\Delta_{\alpha,\rho} & = & \rho(\alpha) \\
\Delta_{\tau_1 \to \tau_2,\rho} & = & \{(f_1, f_2) \mid \forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ (f_1\ a_1, f_2\ a_2) \in \Delta_{\tau_2,\rho}\} \\
\Delta_{\forall\alpha.\,\tau,\rho} & = & \{(g_1, g_2) \mid \forall \mathcal{R} \subseteq S_1 \times S_2.\ (g_1\ S_1, g_2\ S_2) \in \Delta_{\tau,\rho[\alpha \mapsto \mathcal{R}]}\}
\end{array}$$

Then, for every closed term t of closed type τ:

$$([\![t]\!]_{\emptyset,\emptyset}, [\![t]\!]_{\emptyset,\emptyset}) \in \Delta_{\tau,\emptyset}.$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate.

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $([\![t]\!]_{\theta_1,\sigma_1}, [\![t]\!]_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1.\ t) : \tau_1 \to \tau_2}$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{(\llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho}}$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $([\![t]\!]_{\theta_1,\sigma_1}, [\![t]\!]_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ ([\![t]\!]_{\theta_1,\sigma_1[x \mapsto a_1]}, [\![t]\!]_{\theta_2,\sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2,\rho}}{([\![\lambda x : \tau_1.\ t]\!]_{\theta_1,\sigma_1}, [\![\lambda x : \tau_1.\ t]\!]_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho}}$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $([\![t]\!]_{\theta_1,\sigma_1}, [\![t]\!]_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ ([\![t]\!]_{\theta_1,\sigma_1[x\mapsto a_1]}, [\![t]\!]_{\theta_2,\sigma_2[x\mapsto a_2]}) \in \Delta_{\tau_2,\rho}}{([\![\lambda x : \tau_1.\ t]\!]_{\theta_1,\sigma_1}, [\![\lambda x : \tau_1.\ t]\!]_{\theta_2,\sigma_2}) \in \Delta_{\tau_1\to\tau_2,\rho}}$$

$$\frac{\Gamma \vdash t : \tau_1 \to \tau_2 \qquad \Gamma \vdash u : \tau_1}{\Gamma \vdash (t\ u) : \tau_2}$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ (\llbracket t \rrbracket_{\theta_1,\sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2,\sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2,\rho}}{(\llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho}}$$

$$\frac{\Gamma \vdash t : \tau_1 \to \tau_2 \qquad \Gamma \vdash u : \tau_1}{(\llbracket t\ u \rrbracket_{\theta_1,\sigma_1}, \llbracket t\ u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_2,\rho}}$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ (\llbracket t \rrbracket_{\theta_1,\sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2,\sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2,\rho}}{(\llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho}}$$

$$\frac{(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho} \qquad (\llbracket u \rrbracket_{\theta_1,\sigma_1}, \llbracket u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1,\rho}}{(\llbracket t\ u \rrbracket_{\theta_1,\sigma_1}, \llbracket t\ u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_2,\rho}}$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ (\llbracket t \rrbracket_{\theta_1,\sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2,\sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2,\rho}}{(\llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho}}$$

$$\frac{(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho} \qquad (\llbracket u \rrbracket_{\theta_1,\sigma_1}, \llbracket u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1,\rho}}{(\llbracket t\ u \rrbracket_{\theta_1,\sigma_1}, \llbracket t\ u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_2,\rho}}$$

$$\frac{\alpha, \Gamma \vdash t : \tau}{\Gamma \vdash (\Lambda\alpha.\ t) : \forall \alpha.\ \tau}$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ (\llbracket t \rrbracket_{\theta_1,\sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2,\sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2,\rho}}{(\llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho}}$$

$$\frac{(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho} \qquad (\llbracket u \rrbracket_{\theta_1,\sigma_1}, \llbracket u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1,\rho}}{(\llbracket t\ u \rrbracket_{\theta_1,\sigma_1}, \llbracket t\ u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_2,\rho}}$$

$$\frac{\alpha, \Gamma \vdash t : \tau}{(\llbracket \Lambda\alpha.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \Lambda\alpha.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\forall\alpha.\,\tau,\rho}}$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ (\llbracket t \rrbracket_{\theta_1,\sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2,\sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2,\rho}}{(\llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho}}$$

$$\frac{(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho} \qquad (\llbracket u \rrbracket_{\theta_1,\sigma_1}, \llbracket u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1,\rho}}{(\llbracket t\ u \rrbracket_{\theta_1,\sigma_1}, \llbracket t\ u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_2,\rho}}$$

$$\frac{\forall \mathcal{R} \subseteq S_1 \times S_2.\ (\llbracket t \rrbracket_{\theta_1[\alpha \mapsto S_1],\sigma_1}, \llbracket t \rrbracket_{\theta_2[\alpha \mapsto S_2],\sigma_2}) \in \Delta_{\tau,\rho[\alpha \mapsto \mathcal{R}]}}{(\llbracket \Lambda\alpha.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \Lambda\alpha.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\forall \alpha.\,\tau,\rho}}$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $([\![t]\!]_{\theta_1,\sigma_1}, [\![t]\!]_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ ([\![t]\!]_{\theta_1,\sigma_1[x\mapsto a_1]}, [\![t]\!]_{\theta_2,\sigma_2[x\mapsto a_2]}) \in \Delta_{\tau_2,\rho}}{([\![\lambda x : \tau_1.\ t]\!]_{\theta_1,\sigma_1}, [\![\lambda x : \tau_1.\ t]\!]_{\theta_2,\sigma_2}) \in \Delta_{\tau_1\to\tau_2,\rho}}$$

$$\frac{([\![t]\!]_{\theta_1,\sigma_1}, [\![t]\!]_{\theta_2,\sigma_2}) \in \Delta_{\tau_1\to\tau_2,\rho} \qquad ([\![u]\!]_{\theta_1,\sigma_1}, [\![u]\!]_{\theta_2,\sigma_2}) \in \Delta_{\tau_1,\rho}}{([\![t\ u]\!]_{\theta_1,\sigma_1}, [\![t\ u]\!]_{\theta_2,\sigma_2}) \in \Delta_{\tau_2,\rho}}$$

$$\frac{\forall \mathcal{R} \subseteq S_1 \times S_2.\ ([\![t]\!]_{\theta_1[\alpha\mapsto S_1],\sigma_1}, [\![t]\!]_{\theta_2[\alpha\mapsto S_2],\sigma_2}) \in \Delta_{\tau,\rho[\alpha\mapsto\mathcal{R}]}}{([\![\Lambda\alpha.\ t]\!]_{\theta_1,\sigma_1}, [\![\Lambda\alpha.\ t]\!]_{\theta_2,\sigma_2}) \in \Delta_{\forall\alpha.\,\tau,\rho}}$$

$$\frac{\Gamma \vdash t : \forall\alpha.\ \tau}{\Gamma \vdash (t\ \tau') : \tau[\tau'/\alpha]}$$

Proof Sketch

Prove the following more general statement:

$$\Gamma \vdash t : \tau \text{ implies } (\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho} \ ,$$
$$\text{provided } (\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho} \text{ for every } x : \tau' \text{ in } \Gamma$$

by induction on the structure of typing derivations.

The base case is immediate. In the step cases:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ (\llbracket t \rrbracket_{\theta_1,\sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2,\sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2,\rho}}{(\llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho}}$$

$$\frac{(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho} \qquad (\llbracket u \rrbracket_{\theta_1,\sigma_1}, \llbracket u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1,\rho}}{(\llbracket t\ u \rrbracket_{\theta_1,\sigma_1}, \llbracket t\ u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_2,\rho}}$$

$$\frac{\forall \mathcal{R} \subseteq S_1 \times S_2.\ (\llbracket t \rrbracket_{\theta_1[\alpha \mapsto S_1],\sigma_1}, \llbracket t \rrbracket_{\theta_2[\alpha \mapsto S_2],\sigma_2}) \in \Delta_{\tau,\rho[\alpha \mapsto \mathcal{R}]}}{(\llbracket \Lambda\alpha.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \Lambda\alpha.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\forall\alpha.\,\tau,\rho}}$$

$$\frac{\Gamma \vdash t : \forall\alpha.\ \tau}{(\llbracket t\ \tau' \rrbracket_{\theta_1,\sigma_1}, \llbracket t\ \tau' \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau[\tau'/\alpha],\rho}}$$

Proof Sketch

Prove the following more general statement:

$\Gamma \vdash t : \tau$ implies $(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau,\rho}$,
provided $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau',\rho}$ for every $x : \tau'$ in Γ

by induction on the structure of typing derivations.
The base case is immediate. In the step cases:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1,\rho}.\ (\llbracket t \rrbracket_{\theta_1,\sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2,\sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2,\rho}}{(\llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \lambda x : \tau_1.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho}}$$

$$\frac{(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1 \to \tau_2,\rho} \qquad (\llbracket u \rrbracket_{\theta_1,\sigma_1}, \llbracket u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_1,\rho}}{(\llbracket t\ u \rrbracket_{\theta_1,\sigma_1}, \llbracket t\ u \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau_2,\rho}}$$

$$\frac{\forall \mathcal{R} \subseteq S_1 \times S_2.\ (\llbracket t \rrbracket_{\theta_1[\alpha \mapsto S_1],\sigma_1}, \llbracket t \rrbracket_{\theta_2[\alpha \mapsto S_2],\sigma_2}) \in \Delta_{\tau,\rho[\alpha \mapsto \mathcal{R}]}}{(\llbracket \Lambda\alpha.\ t \rrbracket_{\theta_1,\sigma_1}, \llbracket \Lambda\alpha.\ t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\forall\alpha.\,\tau,\rho}}$$

$$\frac{(\llbracket t \rrbracket_{\theta_1,\sigma_1}, \llbracket t \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\forall\alpha.\,\tau,\rho}}{(\llbracket t\ \tau' \rrbracket_{\theta_1,\sigma_1}, \llbracket t\ \tau' \rrbracket_{\theta_2,\sigma_2}) \in \Delta_{\tau[\tau'/\alpha],\rho}}$$

Adding Datatypes

$$\text{Types: } \tau := \cdots \mid \mathit{Bool} \mid [\tau]$$
$$\text{Terms: } t := \cdots \mid \mathit{True} \mid \mathit{False} \mid []_\tau \mid t : t \mid \mathit{case}\ t\ \mathit{of}\ \{\cdots\}$$

Adding Datatypes

Types: $\tau := \;\cdots \mid \mathit{Bool} \mid [\tau]$

Terms: $t := \;\cdots \mid \mathit{True} \mid \mathit{False} \mid []_\tau \mid t : t \mid \mathit{case}\ t\ \mathit{of}\ \{\cdots\}$

$$\Gamma \vdash \mathit{True} : \mathit{Bool} \quad , \quad \Gamma \vdash \mathit{False} : \mathit{Bool} \quad , \quad \Gamma \vdash []_\tau : [\tau]$$

$$\frac{\Gamma \vdash t : \tau \qquad \Gamma \vdash u : [\tau]}{\Gamma \vdash (t : u) : [\tau]}$$

$$\frac{\Gamma \vdash t : \mathit{Bool} \qquad \Gamma \vdash u : \tau \qquad \Gamma \vdash v : \tau}{\Gamma \vdash (\mathit{case}\ t\ \mathit{of}\ \{\mathit{True} \to u\,; \mathit{False} \to v\}) : \tau}$$

$$\frac{\Gamma \vdash t : [\tau'] \qquad \Gamma \vdash u : \tau \qquad \Gamma, x_1 : \tau', x_2 : [\tau'] \vdash v : \tau}{\Gamma \vdash (\mathit{case}\ t\ \mathit{of}\ \{[] \to u\,; (x_1 : x_2) \to v\}) : \tau}$$

Adding Datatypes

$$\text{Types: } \tau := \;\cdots \mid \mathit{Bool} \mid [\tau]$$
$$\text{Terms: } t := \;\cdots \mid \mathit{True} \mid \mathit{False} \mid []_\tau \mid t : t \mid \mathit{case}\ t\ \mathit{of}\ \{\cdots\}$$

$$\Gamma \vdash \mathit{True} : \mathit{Bool} \quad , \quad \Gamma \vdash \mathit{False} : \mathit{Bool} \quad , \quad \Gamma \vdash []_\tau : [\tau]$$

$$\frac{\Gamma \vdash t : \tau \qquad \Gamma \vdash u : [\tau]}{\Gamma \vdash (t : u) : [\tau]}$$

$$\frac{\Gamma \vdash t : \mathit{Bool} \qquad \Gamma \vdash u : \tau \qquad \Gamma \vdash v : \tau}{\Gamma \vdash (\mathit{case}\ t\ \mathit{of}\ \{\mathit{True} \to u\,; \mathit{False} \to v\}) : \tau}$$

$$\frac{\Gamma \vdash t : [\tau'] \qquad \Gamma \vdash u : \tau \qquad \Gamma, x_1 : \tau', x_2 : [\tau'] \vdash v : \tau}{\Gamma \vdash (\mathit{case}\ t\ \mathit{of}\ \{[] \to u\,; (x_1 : x_2) \to v\}) : \tau}$$

With the straightforward extension of term-semantics and with

$$\begin{aligned} \Delta_{\mathit{Bool},\rho} &= \{(\mathit{True}, \mathit{True}), (\mathit{False}, \mathit{False})\} \\ \Delta_{[\tau],\rho} &= \{([x_1, \ldots, x_n], [y_1, \ldots, y_n]) \mid n \geq 0, (x_i, y_i) \in \Delta_{\tau,\rho}\}\,, \end{aligned}$$

the parametricity theorem still holds.

Adding General Recursion

Terms: $t := \quad \cdots \mid \mathit{fix}\ t$

Adding General Recursion

Terms: $t := \quad \cdots \mid \mathit{fix}\ t$

$$\frac{\Gamma \vdash t : \tau \to \tau}{\Gamma \vdash (\mathit{fix}\ t) : \tau}$$

Adding General Recursion

Terms: $t := \quad \cdots \mid \mathit{fix}\ t$

$$\frac{\Gamma \vdash t : \tau \to \tau}{\Gamma \vdash (\mathit{fix}\ t) : \tau}$$

To provide semantics, types are interpreted as pointed complete partial orders now.

$$[\![\mathit{fix}\ t]\!]_{\theta,\sigma} = \bigsqcup_{i \geq 0} ([\![t]\!]^{i}_{\theta,\sigma}\ \bot).$$

Adding General Recursion

Terms: $t := \ \cdots \mid \mathit{fix}\ t$

$$\frac{\Gamma \vdash t : \tau \to \tau}{\Gamma \vdash (\mathit{fix}\ t) : \tau}$$

To provide semantics, types are interpreted as pointed complete partial orders now.

$$[\![\mathit{fix}\ t]\!]_{\theta,\sigma} = \bigsqcup_{i \geq 0} ([\![t]\!]^{i}_{\theta,\sigma}\ \bot).$$

The parametricity theorem still holds, provided all relations are strict and continuous.

Back to Haskell

The original example

$$\begin{array}{l} \mathit{filter} :: \forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha] \\ \mathit{filter}\ p\ [] \qquad\quad = [] \\ \mathit{filter}\ p\ (x : xs) = \mathit{if}\ \ p\ x\ \ \mathit{then}\ x : \mathit{filter}\ p\ xs \\ \qquad\qquad\qquad\qquad\quad \mathit{else}\ \ \mathit{filter}\ p\ xs \end{array}$$

has a “desugaring” in the extended calculus as follows:

$$\begin{array}{l} \mathit{fix}\ (\lambda f : (\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]). \\ \quad \Lambda\alpha.\ \lambda p : (\alpha \to \mathit{Bool}).\ \lambda l : [\alpha]. \\ \quad \mathit{case}\ \ l\ \ \mathit{of}\ \ \{[] \qquad\quad \to []_\alpha\,; \\ \qquad\qquad\qquad (x : xs) \to \mathit{case}\ p\ x\ \mathit{of} \\ \qquad\qquad\qquad\qquad\qquad \{\mathit{True}\ \to x : (f\ \ \alpha\ \ p\ xs)\,; \\ \qquad\qquad\qquad\qquad\qquad\ \ \mathit{False} \to f\ \ \alpha\ \ p\ xs\}\}) \end{array}$$

The Magic Dissolves

Given g of type $\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$,
by the parametricity theorem:

$$(g, g) \in \Delta_{\forall\alpha.\,(\alpha\to \mathit{Bool})\to[\alpha]\to[\alpha],\emptyset}$$

The Magic Dissolves

Given g of type $\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$,
by the parametricity theorem:

$$
\begin{array}{l}
\quad (g, g) \in \Delta_{\forall\alpha.\,(\alpha\to\mathit{Bool})\to[\alpha]\to[\alpha],\emptyset} \\
\Rightarrow \forall\mathcal{R} \in \mathit{Rel}.\ (g, g) \in \Delta_{(\alpha\to\mathit{Bool})\to[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\quad \text{by definition of } \Delta
\end{array}
$$

The Magic Dissolves

Given g of type $\forall\alpha.\,(\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$,
by the parametricity theorem:

$$
\begin{aligned}
& (g, g) \in \Delta_{\forall\alpha.\,(\alpha\to \mathit{Bool})\to[\alpha]\to[\alpha],\emptyset} \\
\Rightarrow\ & \forall\mathcal{R} \in \mathit{Rel}.\ (g, g) \in \Delta_{(\alpha\to \mathit{Bool})\to[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow\ & \forall\mathcal{R} \in \mathit{Rel}, (a_1, a_2) \in \Delta_{\alpha\to \mathit{Bool},[\alpha\mapsto\mathcal{R}]}.\ (g\ a_1, g\ a_2) \in \Delta_{[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
& \text{by definition of } \Delta
\end{aligned}
$$

The Magic Dissolves

Given g of type $\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$,
by the parametricity theorem:

$$
\begin{array}{l}
\quad (g, g) \in \Delta_{\forall\alpha.\,(\alpha\to \mathit{Bool})\to[\alpha]\to[\alpha],\emptyset} \\
\Rightarrow \forall\mathcal{R} \in \mathit{Rel}.\ (g, g) \in \Delta_{(\alpha\to \mathit{Bool})\to[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow \forall\mathcal{R} \in \mathit{Rel}, (a_1, a_2) \in \Delta_{\alpha\to \mathit{Bool},[\alpha\mapsto\mathcal{R}]}.\ (g\ a_1, g\ a_2) \in \Delta_{[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow \forall\mathcal{R} \in \mathit{Rel}, (a_1, a_2) \in \Delta_{\alpha\to \mathit{Bool},[\alpha\mapsto\mathcal{R}]}, (l_1, l_2) \in \Delta_{[\alpha],[\alpha\mapsto\mathcal{R}]}. \\
\quad\ (g\ a_1\ l_1, g\ a_2\ l_2) \in \Delta_{[\alpha],[\alpha\mapsto\mathcal{R}]}
\end{array}
$$

by definition of Δ

The Magic Dissolves

Given g of type $\forall\alpha.\,(\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$,
by the parametricity theorem:

$$
\begin{array}{l}
\quad (g, g) \in \Delta_{\forall\alpha.\,(\alpha\to\mathit{Bool})\to[\alpha]\to[\alpha],\emptyset} \\
\Rightarrow \forall\mathcal{R} \in \mathit{Rel}.\ (g, g) \in \Delta_{(\alpha\to\mathit{Bool})\to[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow \forall\mathcal{R} \in \mathit{Rel}, (a_1, a_2) \in \Delta_{\alpha\to\mathit{Bool},[\alpha\mapsto\mathcal{R}]}.\ (g\ a_1, g\ a_2) \in \Delta_{[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow \forall\mathcal{R} \in \mathit{Rel}, (a_1, a_2) \in \Delta_{\alpha\to\mathit{Bool},[\alpha\mapsto\mathcal{R}]}, (l_1, l_2) \in \Delta_{[\alpha],[\alpha\mapsto\mathcal{R}]}. \\
\qquad (g\ a_1\ l_1, g\ a_2\ l_2) \in \Delta_{[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow \forall(a_1, a_2) \in \Delta_{\alpha\to\mathit{Bool},[\alpha\mapsto h]}, (l_1, l_2) \in (\mathit{map}\ h). \\
\qquad (g\ a_1\ l_1, g\ a_2\ l_2) \in (\mathit{map}\ h) \\
\quad \text{by instantiating } \mathcal{R} = h \text{ and realizing that } \Delta_{[\alpha],[\alpha\mapsto h]} = \mathit{map}\ h
\end{array}
$$

for every function h

The Magic Dissolves

Given g of type $\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$,
by the parametricity theorem:

$$
\begin{aligned}
& (g, g) \in \Delta_{\forall\alpha.\,(\alpha\to \mathit{Bool})\to[\alpha]\to[\alpha],\emptyset} \\
\Rightarrow\ & \forall \mathcal{R} \in \mathit{Rel}.\ (g, g) \in \Delta_{(\alpha\to \mathit{Bool})\to[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow\ & \forall \mathcal{R} \in \mathit{Rel}, (a_1, a_2) \in \Delta_{\alpha\to \mathit{Bool},[\alpha\mapsto\mathcal{R}]}.\ (g\ a_1, g\ a_2) \in \Delta_{[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow\ & \forall \mathcal{R} \in \mathit{Rel}, (a_1, a_2) \in \Delta_{\alpha\to \mathit{Bool},[\alpha\mapsto\mathcal{R}]}, (l_1, l_2) \in \Delta_{[\alpha],[\alpha\mapsto\mathcal{R}]}. \\
& \quad (g\ a_1\ l_1, g\ a_2\ l_2) \in \Delta_{[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow\ & \forall (a_1, a_2) \in \Delta_{\alpha\to \mathit{Bool},[\alpha\mapsto h]}, (l_1, l_2) \in (\mathit{map}\ h). \\
& \quad (g\ a_1\ l_1, g\ a_2\ l_2) \in (\mathit{map}\ h) \\
\Rightarrow\ & \forall (l_1, l_2) \in (\mathit{map}\ h).\ (g\ (p \circ h)\ l_1, g\ p\ l_2) \in (\mathit{map}\ h) \\
& \text{by instantiating } (a_1, a_2) = (p \circ h, p) \in \Delta_{\alpha\to \mathit{Bool},[\alpha\mapsto h]}
\end{aligned}
$$

for every function h and every p.

The Magic Dissolves

Given g of type $\forall\alpha.\ (\alpha \to \mathit{Bool}) \to [\alpha] \to [\alpha]$, by the parametricity theorem:

$$
\begin{array}{l}
\quad (g, g) \in \Delta_{\forall\alpha.\,(\alpha\to\mathit{Bool})\to[\alpha]\to[\alpha],\emptyset} \\
\Rightarrow \forall\mathcal{R} \in \mathit{Rel}.\ (g, g) \in \Delta_{(\alpha\to\mathit{Bool})\to[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow \forall\mathcal{R} \in \mathit{Rel}, (a_1, a_2) \in \Delta_{\alpha\to\mathit{Bool},[\alpha\mapsto\mathcal{R}]}.\ (g\ a_1, g\ a_2) \in \Delta_{[\alpha]\to[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow \forall\mathcal{R} \in \mathit{Rel}, (a_1, a_2) \in \Delta_{\alpha\to\mathit{Bool},[\alpha\mapsto\mathcal{R}]}, (l_1, l_2) \in \Delta_{[\alpha],[\alpha\mapsto\mathcal{R}]}. \\
\quad (g\ a_1\ l_1, g\ a_2\ l_2) \in \Delta_{[\alpha],[\alpha\mapsto\mathcal{R}]} \\
\Rightarrow \forall(a_1, a_2) \in \Delta_{\alpha\to\mathit{Bool},[\alpha\mapsto h]}, (l_1, l_2) \in (\mathit{map}\ h). \\
\quad (g\ a_1\ l_1, g\ a_2\ l_2) \in (\mathit{map}\ h) \\
\Rightarrow \forall(l_1, l_2) \in (\mathit{map}\ h).\ (g\ (p \circ h)\ l_1, g\ p\ l_2) \in (\mathit{map}\ h)
\end{array}
$$

for every function h and every p.

This is exactly the claim (1) for $g = \mathit{filter}$!

References

J.-Y. Girard.
Interprétation functionelle et élimination des coupures dans l'arithmétique d'ordre supérieure.
PhD thesis, Université Paris VII, 1972.

J.C. Reynolds.
Towards a theory of type structure.
In *Colloque sur la Programmation, Proceedings*, pages 408–423. Springer-Verlag, 1974.

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

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