Parametricity: A Story in Trivializing 15-150

Hype for Types

November 4, 2024

Motivation

Identity

Recall from last week the function $f : \forall X.X \to X$. A natural question to as is "how many such functions are there?"

Identity

Recall from last week the function $f : \forall X.X \to X$. A natural question to as is "how many such functions are there?"

One. Because...

Identity

Recall from last week the function $f : \forall X.X \to X$. A natural question to as is "how many such functions are there?"

One. Because... you get an $x : \alpha$ and...

Identity

Recall from last week the function $f : \forall X.X \to X$. A natural question to as is "how many such functions are there?"

One. Because... you get an $x : \alpha$ and... what else can you do with it besides return it?

Identity

Recall from last week the function $f : \forall X.X \to X$. A natural question to as is "how many such functions are there?"

One. Because... you get an $x : \alpha$ and... what else can you do with it besides return it? Or something...

Identity

Recall from last week the function $f : \forall X.X \rightarrow X$. A natural question to as is "how many such functions are there?"

One. Because... you get an $x : \alpha$ and... what else can you do with it besides return it? Or something...

This is not very satisfying. So, we would like an equational theory for polymorphic functions to *prove* that there is only one such function.

More Generally...

If I give you a function $f : \forall X.\text{List}(X) \to \text{List}(X)$, what function do you expect it to be?

[1] Pretend this is total

More Generally...

If I give you a function $f : \forall X.\mathsf{List}(X) \to \mathsf{List}(X)$, what function do you expect it to be?

You probably said **reverse** or **duplicate-every-element** or **take-the-first-two-elements-and-copy-them-five-times-and-then-append-the-third-element-to-the-end**[1] $: \forall X.\mathsf{List}(X) \to \mathsf{List}(X)$

[1] Pretend this is total

More Generally...

If I give you a function $f : \forall X.\mathrm{List}(X) \to \mathrm{List}(X)$, what function do you expect it to be?

You probably said **reverse** or **duplicate-every-element** or **take-the-first-two-elements-and-copy-them-five-times-and-then-append-the-third-element-to-the-end**[1] $: \forall X.\mathrm{List}(X) \to \mathrm{List}(X)$

The point is that any function you described is returning some permutation/duplication/removal of the elements which *does not refer to the values themselves.*

[1] Pretend this is total

Mapping Over These

Take your function f from before, and now take your favorite function $g : A \to B$. Consider the following equation:

$$(\textbf{map } g) \circ f[A] = f[B] \circ (\textbf{map } g)$$

[2] We at Hype for Types would like to make it known that this is a joke, we love induction

Mapping Over These

Take your function f from before, and now take your favorite function $g : A \to B$. Consider the following equation:

$$(\textbf{map } g) \circ f[A] = f[B] \circ (\textbf{map } g)$$

It turns out this is true. The intuition is that since f cannot refer to the elements themselves, mapping a function g and then permuting the list should be the same as permuting the list then mapping a function g.

[2] We at Hype for Types would like to make it known that this is a joke, we love induction

Mapping Over These

Take your function f from before, and now take your favorite function $g : A \to B$. Consider the following equation:

$$(\textbf{map } g) \circ f[A] = f[B] \circ (\textbf{map } g)$$

It turns out this is true. The intuition is that since f cannot refer to the elements themselves, mapping a function g and then permuting the list should be the same as permuting the list then mapping a function g.

You probably proved something like this in 15-150

$$\text{For all } f : A \to B, (\textbf{map } f) \circ \textbf{reverse} = \textbf{reverse} \circ (\textbf{map } f)$$

by induction on the input list.

[2] We at Hype for Types would like to make it known that this is a joke, we love induction

Mapping Over These

Take your function f from before, and now take your favorite function $g : A \to B$. Consider the following equation:

$$(\textbf{map } g) \circ f[A] = f[B] \circ (\textbf{map } g)$$

It turns out this is true. The intuition is that since f cannot refer to the elements themselves, mapping a function g and then permuting the list should be the same as permuting the list then mapping a function g.

You probably proved something like this in 15-150

$$\text{For all } f : A \to B, (\textbf{map } f) \circ \textbf{reverse} = \textbf{reverse} \circ (\textbf{map } f)$$

by induction on the input list. We hate induction[2], let's do better.

[2] We at Hype for Types would like to make it known that this is a joke, we love induction

What the Hype is a Type

Let's ask a fundamental question: how do you think about types?

[3] Types are not technically sets, but let's not get into that right now

What the Hype is a Type

Let's ask a fundamental question: how do you think about types?
You probably view types as sets[3]:

- $[\![\text{Bool}]\!] = \{0, 1\}$
- $[\![\text{Int}]\!] = \mathbb{Z}$
- $[\![A \times B]\!] = [\![A]\!] \times [\![B]\!]$
- $[\![A \to B]\!] = [\![B]\!]^{[\![A]\!]}$
- $[\![\text{List}(A)]\!] = [\![A]\!]^*$

This is generally fine, but today we will view types as relations.

[3] Types are not technically sets, but let's not get into that right now

Some Notation and Ideas

In order to set the scene, we need to introduce some notation:

[4] This is a surprise tool that will help us later

Some Notation and Ideas

In order to set the scene, we need to introduce some notation:

- $\mathcal{A} : A \Leftrightarrow A'$ means $\mathcal{A}$ is a relation between A and A', i.e. $\mathcal{A} \subseteq A \times A'$

[4] This is a surprise tool that will help us later

Some Notation and Ideas

In order to set the scene, we need to introduce some notation:

- $\mathcal{A} : A \Leftrightarrow A'$ means $\mathcal{A}$ is a relation between A and A', i.e. $\mathcal{A} \subseteq A \times A'$
- If $x \in A$ and $x' \in A'$, we write $(x, x') \in \mathcal{A}$ to mean x and x' are related under $\mathcal{A}$

[4]This is a surprise tool that will help us later

Some Notation and Ideas

In order to set the scene, we need to introduce some notation:

- $\mathcal{A} : A \Leftrightarrow A'$ means $\mathcal{A}$ is a relation between A and A', i.e. $\mathcal{A} \subseteq A \times A'$
- If $x \in A$ and $x' \in A'$, we write $(x, x') \in \mathcal{A}$ to mean x and x' are related under $\mathcal{A}$
- I_A is the identity relation on A, i.e. for all $x \in A$, $(x, x) \in I_A$

[4] This is a surprise tool that will help us later

Some Notation and Ideas

In order to set the scene, we need to introduce some notation:

- $\mathcal{A} : A \Leftrightarrow A'$ means $\mathcal{A}$ is a relation between A and A', i.e. $\mathcal{A} \subseteq A \times A'$
- If $x \in A$ and $x' \in A'$, we write $(x, x') \in \mathcal{A}$ to mean x and x' are related under $\mathcal{A}$
- I_A is the identity relation on A, i.e. for all $x \in A$, $(x, x) \in I_A$
- We may view any function $f : A \to B$ as a relation $\mathcal{R}_f : A \Leftrightarrow B$ via $\{(a, f\ a) \mid a \in A\}$

[4] This is a surprise tool that will help us later

Some Notation and Ideas

In order to set the scene, we need to introduce some notation:

- $\mathcal{A} : A \Leftrightarrow A'$ means $\mathcal{A}$ is a relation between A and A', i.e. $\mathcal{A} \subseteq A \times A'$
- If $x \in A$ and $x' \in A'$, we write $(x, x') \in \mathcal{A}$ to mean x and x' are related under $\mathcal{A}$
- I_A is the identity relation on A, i.e. for all $x \in A$, $(x, x) \in I_A$
- We may view any function $f : A \to B$ as a relation $\mathcal{R}_f : A \Leftrightarrow B$ via $\{(a, f\ a) \mid a \in A\}$
  - We can expand this to a relation on lists $\text{List}(\mathcal{R}_f) : \text{List}(A) \to \text{List}(B)$, where $\{(a, \textbf{map}\ f\ a) \mid a \in \text{List}(A)\}$[4]

[4] This is a surprise tool that will help us later

Some Notation and Ideas

In order to set the scene, we need to introduce some notation:

- $\mathcal{A} : A \Leftrightarrow A'$ means $\mathcal{A}$ is a relation between A and A', i.e. $\mathcal{A} \subseteq A \times A'$
- If $x \in A$ and $x' \in A'$, we write $(x, x') \in \mathcal{A}$ to mean x and x' are related under $\mathcal{A}$
- I_A is the identity relation on A, i.e. for all $x \in A$, $(x, x) \in I_A$
- We may view any function $f : A \to B$ as a relation $\mathcal{R}_f : A \Leftrightarrow B$ via $\{(a, f\ a) \mid a \in A\}$
  - We can expand this to a relation on lists $\text{List}(\mathcal{R}_f) : \text{List}(A) \to \text{List}(B)$, where $\{(a, \textbf{map}\ f\ a) \mid a \in \text{List}(A)\}$[4]

Using these ideas, we give an interpretation of types as relations instead of sets.

[4]This is a surprise tool that will help us later

Types as Relations

We may interpret some basic types as relations in the following manner:

- $[\![\text{Bool}]\!] = I_{\text{Bool}}$
- $[\![\text{Int}]\!] = I_{\text{Int}}$
- $[\![A \times B]\!] = \{((x, y), (x', y')) \mid (x, x') \in A \text{ and } (y, y') \in B\}$

Types as Relations

We may interpret some basic types as relations in the following manner:

- $[\![\text{Bool}]\!] = I_{\text{Bool}}$
- $[\![\text{Int}]\!] = I_{\text{Int}}$
- $[\![A \times B]\!] = \{((x, y), (x', y')) \mid (x, x') \in A \text{ and } (y, y') \in B\}$

For more complicated types:

Types as Relations

We may interpret some basic types as relations in the following manner:

- $[\![\text{Bool}]\!] = I_{\text{Bool}}$
- $[\![\text{Int}]\!] = I_{\text{Int}}$
- $[\![A \times B]\!] = \{((x, y), (x', y')) \mid (x, x') \in A \text{ and } (y, y') \in B\}$

For more complicated types:

- For a relation $\mathcal{A} : A \Leftrightarrow A'$, we say that $(l, l') \in \text{List}(\mathcal{A})$ if l and l' have the same length and each of their elements are pairwise related by $\mathcal{A}$

Types as Relations

We may interpret some basic types as relations in the following manner:

- $[\![\text{Bool}]\!] = I_{\text{Bool}}$
- $[\![\text{Int}]\!] = I_{\text{Int}}$
- $[\![A \times B]\!] = \{((x, y), (x', y')) \mid (x, x') \in A \text{ and } (y, y') \in B\}$

For more complicated types:

- For a relation $\mathcal{A} : A \Leftrightarrow A'$, we say that $(l, l') \in \text{List}(\mathcal{A})$ if l and l' have the same length and each of their elements are pairwise related by $\mathcal{A}$
- For two relations $\mathcal{A} : A \Leftrightarrow A'$ and $\mathcal{B} : B \Leftrightarrow B'$, we say that $(f, g) \in \mathcal{A} \to \mathcal{B}$ if, given inputs $(x, x') \in \mathcal{A}$, we have $(f\ x, g\ x') \in \mathcal{B}$

Types as Relations

We may interpret some basic types as relations in the following manner:

- $[\![\text{Bool}]\!] = I_{\text{Bool}}$
- $[\![\text{Int}]\!] = I_{\text{Int}}$
- $[\![A \times B]\!] = \{((x, y), (x', y')) \mid (x, x') \in A \text{ and } (y, y') \in B\}$

For more complicated types:

- For a relation $\mathcal{A} : A \Leftrightarrow A'$, we say that $(l, l') \in \text{List}(\mathcal{A})$ if l and l' have the same length and each of their elements are pairwise related by $\mathcal{A}$
- For two relations $\mathcal{A} : A \Leftrightarrow A'$ and $\mathcal{B} : B \Leftrightarrow B'$, we say that $(f, g) \in \mathcal{A} \rightarrow \mathcal{B}$ if, given inputs $(x, x') \in \mathcal{A}$, we have $(f\ \ x, g\ \ x') \in \mathcal{B}$
- Polymorphic functions are related if they take related typed to related outputs

The Big Theorem

Now, we can state the theorem that will allow us to prove the statements we made previously: **The Parametricity Theorem**

The Big Theorem

Now, we can state the theorem that will allow us to prove the statements we made previously: **The Parametricity Theorem**

$$\text{If } t : T, \text{then } (t, t) \in \mathcal{T}$$

The Big Theorem

Now, we can state the theorem that will allow us to prove the statements we made previously: **The Parametricity Theorem**

$$\text{If } t : T, \text{then } (t, t) \in \mathcal{T}$$

That's... kinda underwhelming.

Why Should You Care

Hang on hang on, before you leave, let's look back at our example from earlier. Recall, we wanted to prove

$$\text{For all } g : A \to B \text{ and } f : \forall X.\text{List}(X) \to \text{List}(X),$$

$$(\textbf{map } g) \circ f[A] = f[B] \circ (\textbf{map } g)$$

Why Should You Care

Hang on hang on, before you leave, let's look back at our example from earlier. Recall, we wanted to prove

$$\text{For all } g : A \to B \text{ and } f : \forall X.\text{List}(X) \to \text{List}(X),$$

$$(\textbf{map } g) \circ f[A] = f[B] \circ (\textbf{map } g)$$

Maybe our new parametricity theorem can help?

A Parametrically Polymorphic Proof

1. Parametricity tells us $(f, f) \in \forall \mathcal{X}.\mathsf{List}(\mathcal{X}) \to \mathsf{List}(\mathcal{X})$

A Parametrically Polymorphic Proof

1. Parametricity tells us $(f, f) \in \forall \mathcal{X}.\mathsf{List}(\mathcal{X}) \to \mathsf{List}(\mathcal{X})$
2. We can expand this to see that for all relations $\mathcal{A} : A \Leftrightarrow A'$, $(f[A], f[A']) \in \mathsf{List}(\mathcal{A}) \to \mathsf{List}(\mathcal{A})$

A Parametrically Polymorphic Proof

1. Parametricity tells us $(f, f) \in \forall \mathcal{X}.\text{List}(\mathcal{X}) \to \text{List}(\mathcal{X})$
2. We can expand this to see that for all relations $\mathcal{A} : A \Leftrightarrow A'$, $(f[A], f[A']) \in \text{List}(\mathcal{A}) \to \text{List}(\mathcal{A})$
3. We can then expand this to see that for all $(xs, xs') \in \text{List}(\mathcal{A})$, $(f[A](xs), f[A'](xs')) \in \text{List}(\mathcal{A})$

A Parametrically Polymorphic Proof

1. Parametricity tells us $(f, f) \in \forall \mathcal{X}.\text{List}(\mathcal{X}) \to \text{List}(\mathcal{X})$
2. We can expand this to see that for all relations $\mathcal{A} : A \Leftrightarrow A'$, $(f[A], f[A']) \in \text{List}(\mathcal{A}) \to \text{List}(\mathcal{A})$
3. We can then expand this to see that for all $(xs, xs') \in \text{List}(\mathcal{A})$, $(f[A](xs), f[A'](xs')) \in \text{List}(\mathcal{A})$

This seems to be getting us somewhere, but this is too general to be useful. Let's focus on when $\mathcal{A}$ is the relation $\mathcal{R}_g$ for a function $g : A \to A'$, as defined before. Then, we have:

$$\text{If } xs \in \text{List}(A), \text{then } (xs, \mathbf{map}\ g\ xs) \in \text{List}(\mathcal{R}_g)$$

A Parametrically Polymorphic Proof

4. From the previous slide, we deduced that for all $(xs, xs') \in \mathsf{List}(\mathcal{A})$, $(f[A](xs), f[A'](xs')) \in \mathsf{List}(\mathcal{A})$

A Parametrically Polymorphic Proof

4. From the previous slide, we deduced that for all $(xs, xs') \in \text{List}(\mathcal{A})$, $(f[A](xs), f[A'](xs')) \in \text{List}(\mathcal{A})$
5. If we set $\mathcal{A} = \mathcal{R}_g$, we get that for all $g : A \to A'$, if $(xs, \textbf{map}\ g\ xs) \in \text{List}(\mathcal{R}_g)$, then $(f[A](xs), f[A'](\textbf{map}\ g\ xs)) \in \text{List}(\mathcal{R}_g)$

A Parametrically Polymorphic Proof

4. From the previous slide, we deduced that for all $(xs, xs') \in \text{List}(\mathcal{A})$, $(f[A](xs), f[A'](xs')) \in \text{List}(\mathcal{A})$
5. If we set $\mathcal{A} = \mathcal{R}_g$, we get that for all $g : A \to A'$, if $(xs, \textbf{map}\ g\ xs) \in \text{List}(\mathcal{R}_g)$, then $(f[A](xs), f[A'](\textbf{map}\ g\ xs)) \in \text{List}(\mathcal{R}_g)$
6. Recall that the relation for functions relates inputs to their outputs. Since we have $(f[A](xs), f[A'](\textbf{map}\ g\ xs)) \in \text{List}(\mathcal{R}_g)$, this must mean that
$$\textbf{map}\ g\ (f[A](xs)) = f[A'](\textbf{map}\ g\ xs)$$

A Parametrically Polymorphic Proof

4. From the previous slide, we deduced that for all $(xs, xs') \in \text{List}(\mathcal{A})$, $(f[A](xs), f[A'](xs')) \in \text{List}(\mathcal{A})$
5. If we set $\mathcal{A} = \mathcal{R}_g$, we get that for all $g : A \to A'$, if $(xs, \textbf{map}\ g\ xs) \in \text{List}(\mathcal{R}_g)$, then $(f[A](xs), f[A'](\textbf{map}\ g\ xs)) \in \text{List}(\mathcal{R}_g)$
6. Recall that the relation for functions relates inputs to their outputs. Since we have $(f[A](xs), f[A'](\textbf{map}\ g\ xs)) \in \text{List}(\mathcal{R}_g)$, this must mean that

$$\textbf{map}\ g\ (f[A](xs)) = f[A'](\textbf{map}\ g\ xs)$$

In other words, we have

$$(\textbf{map}\ g) \circ f[A] = f[A'] \circ (\textbf{map}\ g)$$

as desired!

15-150? More Like... Parametricity Theorem

We did it! Not only did we prove that

$$(\textbf{map } f) \circ \textbf{reverse} = \textbf{reverse} \circ (\textbf{map } f)$$

we managed to prove something way more general!

The Original Goal

We claim that if $f : \forall X.X \to X$, then $f = id$[5]. You know this intuitively, but we can use parametricity to prove this!

[5] That is to say that its behavior is equivalent to the identity

The Original Goal

We claim that if $f : \forall X.X \to X$, then $f = id$[5]. You know this intuitively, but we can use parametricity to prove this!

1. We start with $(f, f) \in \forall \mathcal{X}.\mathcal{X} \to \mathcal{X}$ by the Parametricity Theorem

[5] That is to say that its behavior is equivalent to the identity

The Original Goal

We claim that if $f : \forall X.X \to X$, then $f = id$[5]. You know this intuitively, but we can use parametricity to prove this!

1. We start with $(f, f) \in \forall \mathcal{X}.\mathcal{X} \to \mathcal{X}$ by the Parametricity Theorem
2. We then have that $(f[A], f[A']) \in \mathcal{A} \to \mathcal{A}$ for $\mathcal{A} : A \Leftrightarrow A'$

[5] That is to say that its behavior is equivalent to the identity

The Original Goal

We claim that if $f : \forall X.X \to X$, then $f = id$[5]. You know this intuitively, but we can use parametricity to prove this!

1. We start with $(f, f) \in \forall \mathcal{X}.\mathcal{X} \to \mathcal{X}$ by the Parametricity Theorem
2. We then have that $(f[A], f[A']) \in \mathcal{A} \to \mathcal{A}$ for $\mathcal{A} : A \Leftrightarrow A'$
3. This then means that, for $(x, x') \in \mathcal{A}$, we have $(f[A](x), f[A'](x')) \in \mathcal{A}$

[5] That is to say that its behavior is equivalent to the identity

The Original Goal

We claim that if $f : \forall X.X \to X$, then $f = id$[5]. You know this intuitively, but we can use parametricity to prove this!

1. We start with $(f, f) \in \forall \mathcal{X}.\mathcal{X} \to \mathcal{X}$ by the Parametricity Theorem
2. We then have that $(f[A], f[A']) \in \mathcal{A} \to \mathcal{A}$ for $\mathcal{A} : A \Leftrightarrow A'$
3. This then means that, for $(x, x') \in \mathcal{A}$, we have $(f[A](x), f[A'](x')) \in \mathcal{A}$

As with the other proof, we get to a point where we need to make a specific choice for $\mathcal{A}$. Here, we will choose $\mathcal{R}_g$, which states that, for all $g : A \to A'$, $(x, g\ x) \in \mathcal{R}_g$ for $x \in A$.

[5] That is to say that its behavior is equivalent to the identity

The Original Goal

4. For all $g : A \to A'$, if $(x, g\ x) \in \mathcal{R}_g$, then $(f[A](x), f[A'](g\ x)) \in \mathcal{R}_g$

The Original Goal

4. For all $g : A \to A'$, if $(x, g\ x) \in \mathcal{R}_g$, then $(f[A](x), f[A'](g\ x)) \in \mathcal{R}_g$
5. Since we're relating inputs to outputs, it must be the case that

$$g(f[A](x)) = f[A'](g\ x)$$

for all $x \in A$

The Original Goal

4. For all $g : A \to A'$, if $(x, g\ x) \in \mathcal{R}_g$, then $(f[A](x), f[A'](g\ x)) \in \mathcal{R}_g$
5. Since we're relating inputs to outputs, it must be the case that

$$g(f[A](x)) = f[A'](g\ x)$$

   for all $x \in A$

To show what we ultimately want to show (that f is the identity), we need one more trick.

The Original Goal

4. For all $g : A \to A'$, if $(x, g\ x) \in \mathcal{R}_g$, then $(f[A](x), f[A'](g\ x)) \in \mathcal{R}_g$
5. Since we're relating inputs to outputs, it must be the case that

$$g(f[A](x)) = f[A'](g\ x)$$

   for all $x \in A$

To show what we ultimately want to show (that f is the identity), we need one more trick.

All we need to do is to choose g to be $\lambda\_ : A.\ x$, i.e. a function that returns the input $x : A$.

The Original Goal

4. For all $g : A \to A'$, if $(x, g\ x) \in \mathcal{R}_g$, then $(f[A](x), f[A'](g\ x)) \in \mathcal{R}_g$
5. Since we're relating inputs to outputs, it must be the case that

$$g(f[A](x)) = f[A'](g\ x)$$

for all $x \in A$

To show what we ultimately want to show (that f is the identity), we need one more trick.

All we need to do is to choose g to be $\lambda\_ : A.\ x$, i.e. a function that returns the input $x : A$. We then have

$$g(f[A](x)) = x \text{ and } f[A](g\ x) = f[A](x)$$

i.e.

$$f[A](x) = x$$

Free Theorems

- Theorems of this form are called “free theorems” named after Phillip Wadler’s paper called, unsurprisingly, “Theorems for Free”[6].

[6] https://dl.acm.org/doi/pdf/10.1145/99370.99404

Free Theorems

- Theorems of this form are called “free theorems” named after Phillip Wadler’s paper called, unsurprisingly, “Theorems for Free”[6].
- Such theorems are direct consequences of the Parametricity Theorem and allow you to prove basically any 15-150 style equality... for free!

[6] https://dl.acm.org/doi/pdf/10.1145/99370.99404

Free Theorems

- Theorems of this form are called "free theorems" named after Phillip Wadler's paper called, unsurprisingly, "Theorems for Free"[6].
- Such theorems are direct consequences of the Parametricity Theorem and allow you to prove basically any 15-150 style equality... for free!
- The website https://free-theorems.nomeata.de/ allows you to generate these free theorems for a given polymorphic type.

[6] https://dl.acm.org/doi/pdf/10.1145/99370.99404