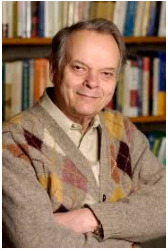
Parametricity via Cohesion

C.B. Aberlé (she/her)

June 20, 2024

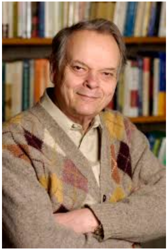
On Parametricity

- Reynolds (1983) introduced the concept of *parametricity* as a formal specification of the idea that "**type structure is a syntactic discipline for enforcing levels of abstraction.**"



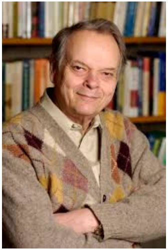
On Parametricity

- Reynolds (1983) introduced the concept of *parametricity* as a formal specification of the idea that "**type structure is a syntactic discipline for enforcing levels of abstraction.**"
- *Idea:* in polymorphic λ-calculus/System F, a polymorphic function of type (e.g.) $\forall X.X \to X$ cannot inspect the types over which it is defined and so must behave essentially the same for all types at which it is instantiated.



On Parametricity

- Reynolds (1983) introduced the concept of *parametricity* as a formal specification of the idea that "**type structure is a syntactic discipline for enforcing levels of abstraction.**"
- *Idea:* in polymorphic λ-calculus/System F, a polymorphic function of type (e.g.) $\forall X.X \to X$ cannot inspect the types over which it is defined and so must behave essentially the same for all types at which it is instantiated.
- Using Reynolds' technique, can show e.g. that all closed terms of type $\forall X.X \to X$ in System F are equivalent to the polymorphic identity function.



On Parametricity

- Reynolds' original analysis of parametricity was wholly *external* to the internal logic of System F.

On Parametricity

- Reynolds' original analysis of parametricity was wholly *external* to the internal logic of System F.
- The emergence of *dependent type theory* raises the possibility of an axiomatic basis for parametricity *internal* to type theory.

On Parametricity

- Reynolds' original analysis of parametricity was wholly *external* to the internal logic of System F.
- The emergence of *dependent type theory* raises the possibility of an axiomatic basis for parametricity *internal* to type theory.
- In recent years, several systems of dependent type theory have emerged, employing various methods to internalize such reasoning via parametricity. Some of these have moreover been applied to significant problems in *homotopy type theory*, arising from the complex higher-categorical structure thereof.

On Parametricity

- Reynolds' original analysis of parametricity was wholly *external* to the internal logic of System F.
- The emergence of *dependent type theory* raises the possibility of an axiomatic basis for parametricity *internal* to type theory.
- In recent years, several systems of dependent type theory have emerged, employing various methods to internalize such reasoning via parametricity. Some of these have moreover been applied to significant problems in *homotopy type theory*, arising from the complex higher-categorical structure thereof.
- As yet no unifying axiomatic framework for such approaches to internal parametricity.

On Parametricity

- Reynolds' original analysis of parametricity was wholly *external* to the internal logic of System F.
- The emergence of *dependent type theory* raises the possibility of an axiomatic basis for parametricity *internal* to type theory.
- In recent years, several systems of dependent type theory have emerged, employing various methods to internalize such reasoning via parametricity. Some of these have moreover been applied to significant problems in *homotopy type theory*, arising from the complex higher-categorical structure thereof.
- As yet no unifying axiomatic framework for such approaches to internal parametricity.
- This work constitutes a first step toward such a unifying framework, based on the category-theoretic concept of *cohesion*.

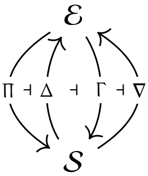
Axiomatic Cohesion

Lawvere: cohesion as an abstract characterization of when one category behaves like a category of spaces defined over another:

Axiomatic Cohesion

Lawvere: cohesion as an abstract characterization of when one category behaves like a category of spaces defined over another:

- A topos $\mathcal{E}$ is *cohesive* over another topos $\mathcal{S}$ if there is a string of four adjoint functors between them as follows:

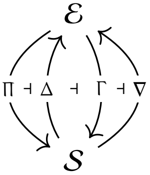


  where Δ, ∇ are fully faithful and Π preserves finite products.

Axiomatic Cohesion

Lawvere: cohesion as an abstract characterization of when one category behaves like a category of spaces defined over another:

- A topos $\mathcal{E}$ is *cohesive* over another topos $\mathcal{S}$ if there is a string of four adjoint functors between them as follows:



  where Δ, ∇ are fully faithful and Π preserves finite products.

- Induces a string of adjoint endofunctors on $\mathcal{E}$:

$$\int \dashv \flat \dashv \sharp$$

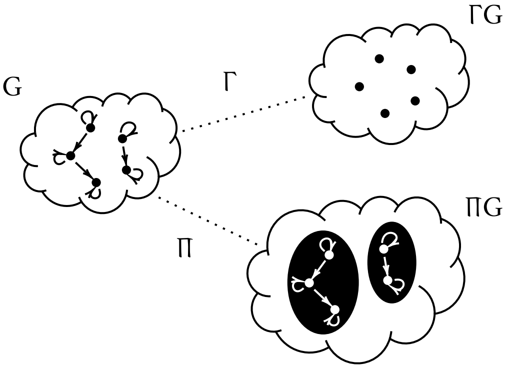
  with $\int, \sharp$ idempotent monads, and $\flat$ an idempotent comonad.

Example: Reflexive Graphs

The category of reflexive graphs **RGph** is cohesive over the category of sets **Set**.

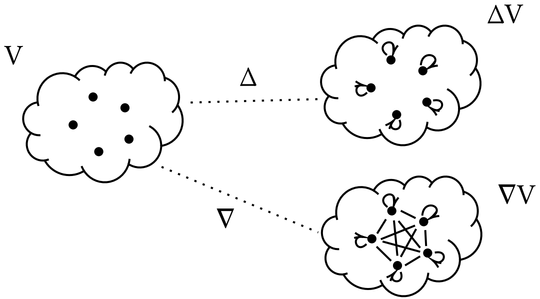
Example: Reflexive Graphs

Γ maps a reflexive graph G to its set of vertices, Π maps G to its set of *weakly connected components*.



Example: Reflexive Graphs

Δ maps a set V to the *discrete* graph with vertex set V, and ∇ maps V to the *codiscrete* (i.e. complete) graph on V.



Example: Reflexive Graphs

- Many classical models of parametricity are based upon semantic interpretations of type structure in terms of reflexive graphs. This is no accident, due to the *cohesive* structure of reflexive graphs.

Example: Reflexive Graphs

- Many classical models of parametricity are based upon semantic interpretations of type structure in terms of reflexive graphs. This is no accident, due to the *cohesive* structure of reflexive graphs.
- More generally, for any base topos $\mathcal{S}$, the corresponding topos $\mathbf{RGph}(\mathcal{S})$ of internal reflexive graphs in $\mathcal{S}$ is cohesive over $\mathcal{S}$. Hence the internal language of reflexive graphs can be used to derive parametricity results for any topos.

Example: Reflexive Graphs

- Many classical models of parametricity are based upon semantic interpretations of type structure in terms of reflexive graphs. This is no accident, due to the *cohesive* structure of reflexive graphs.
- More generally, for any base topos $\mathcal{S}$, the corresponding topos $\mathbf{RGph}(\mathcal{S})$ of internal reflexive graphs in $\mathcal{S}$ is cohesive over $\mathcal{S}$. Hence the internal language of reflexive graphs can be used to derive parametricity results for any topos.
- In fact, this same setup of cohesion is interpretable, *mutatis mutandis*, in the case where $\mathcal{E}, \mathcal{S}$ are not (1-)topoi, but rather ∞-topoi, i.e. models of homotopy type theory (HoTT).

Example: Reflexive Graphs

- Many classical models of parametricity are based upon semantic interpretations of type structure in terms of reflexive graphs. This is no accident, due to the *cohesive* structure of reflexive graphs.
- More generally, for any base topos $\mathcal{S}$, the corresponding topos $\mathbf{RGph}(\mathcal{S})$ of internal reflexive graphs in $\mathcal{S}$ is cohesive over $\mathcal{S}$. Hence the internal language of reflexive graphs can be used to derive parametricity results for any topos.
- In fact, this same setup of cohesion is interpretable, *mutatis mutandis*, in the case where $\mathcal{E}, \mathcal{S}$ are not (1-)topoi, but rather ∞-topoi, i.e. models of homotopy type theory (HoTT).
- We can thus use the language of HoTT – suitably extended with cohesive modalities – to work *synthetically* with the structure of such a cohesive ∞-topos.

Type-Theoretic Cohesion

Following Shulman's (2018) formulation of cohesive HoTT:

Type-Theoretic Cohesion

Following Shulman's (2018) formulation of cohesive HoTT:

Problem: the $\flat$ modality is not well-defined in arbitrary contexts, but only in those consisting entirely of *discrete* variables.

Type-Theoretic Cohesion

Following Shulman's (2018) formulation of cohesive HoTT:

Problem: the $\flat$ modality is not well-defined in arbitrary contexts, but only in those consisting entirely of *discrete* variables.

Solution: modify the structure of contexts to keep track of which variables are *discrete*.

Type-Theoretic Cohesion

Contexts now of the form $\Delta \mid \Xi$ where Δ consists of *discrete* variables, while Ξ consists of ordinary variables. The type of an ordinary variable may depend on both ordinary and discrete variables, but the type of a discrete variable can only depend upon other discrete variables.

$$\frac{\Delta \mid \Xi \ \mathsf{Ctx} \qquad \Delta \mid \Xi \vdash \mathsf{S}\ \mathsf{Type}}{\Delta \mid \Xi, x : \mathsf{S}\ \mathsf{Ctx}} \qquad \frac{\Delta \mid \Xi \ \mathsf{Ctx} \qquad \Delta \mid - \vdash \mathsf{S}\ \mathsf{Type}}{\Delta, x : \mathsf{S} \mid \Xi \ \mathsf{Ctx}}$$

Type-Theoretic Cohesion

Rules for $\flat$ are then essentially those of a Pfenning-Davies-style modal necessity operator:

$$\frac{\Delta \mid - \vdash \mathsf{S}\ \mathsf{Type}}{\Delta \mid \Xi \vdash \flat\mathsf{S}\ \mathsf{Type}} \qquad \frac{\Delta \mid - \vdash s : \mathsf{S}}{\Delta \mid \Xi \vdash s^{\flat} : \flat\mathsf{S}}$$

$$\frac{\Delta \mid \Xi \vdash s : \flat\mathsf{S} \qquad \Delta \mid \Xi, z : \flat\mathsf{S} \vdash \mathrm{R}\ \mathsf{Type} \qquad \Delta, x : \mathsf{S} \mid \Xi \vdash r : \mathrm{R}[x^{\flat}/z]}{\Delta \mid \Xi \vdash \mathsf{let}\ x^{\flat} = s\ \mathsf{in}\ r : \mathrm{R}[s/z]}$$

$$\mathsf{let}\ x^{\flat} = s^{\flat}\ \mathsf{in}\ r \equiv r[s/x]$$

Type-Theoretic Cohesion

Rules for $\flat$ are then essentially those of a Pfenning-Davies-style modal necessity operator:

$$\frac{\Delta \mid - \vdash \mathsf{S}\ \mathsf{Type}}{\Delta \mid \Xi \vdash \flat\mathsf{S}\ \mathsf{Type}} \qquad \frac{\Delta \mid - \vdash s : \mathsf{S}}{\Delta \mid \Xi \vdash s^{\flat} : \flat\mathsf{S}}$$

$$\frac{\Delta \mid \Xi \vdash s : \flat\mathsf{S} \qquad \Delta \mid \Xi, z : \flat\mathsf{S} \vdash \mathrm{R}\ \mathsf{Type} \qquad \Delta, x : \mathsf{S} \mid \Xi \vdash r : \mathrm{R}[x^{\flat}/z]}{\Delta \mid \Xi \vdash \mathsf{let}\ x^{\flat} = s\ \mathsf{in}\ r : \mathrm{R}[s/z]}$$

$$\mathsf{let}\ x^{\flat} = s^{\flat}\ \mathsf{in}\ r \equiv r[s/x]$$

For any type S, we have $\epsilon_{\mathsf{S}} : \flat\mathsf{S} \to \mathsf{S}$ given by

$$\epsilon(s) :\equiv \mathsf{let}\ x^{\flat} = s\ \mathsf{in}\ x$$

S is *discrete* if ϵ_{S} is an equivalence.

Sufficient Cohesion

How is this all related to parametricity?

Sufficient Cohesion

How is this all related to parametricity?

- Cohesion lets us ask what is the *shape* of an abstract relation between elements of a type.

Sufficient Cohesion

How is this all related to parametricity?

- Cohesion lets us ask what is the *shape* of an abstract relation between elements of a type.
- In particular, there should be some type I that *classifies* this shape, in the sense that maps $I \longrightarrow S$ correspond to abstract relations – or, to use a more geometric term, *paths* – between elements of S.

Sufficient Cohesion

How is this all related to parametricity?

- Cohesion lets us ask what is the *shape* of an abstract relation between elements of a type.
- In particular, there should be some type I that *classifies* this shape, in the sense that maps $\mathrm{I} \longrightarrow \mathrm{S}$ correspond to abstract relations – or, to use a more geometric term, *paths* – between elements of S.
- In **RGph**, the role of such a path classifier is played by the *walking edge graph* $\mathrm{I} := \{\mathbf{0} \rightarrow \mathbf{1}\}$

Sufficient Cohesion

How is this all related to parametricity?

- Cohesion lets us ask what is the *shape* of an abstract relation between elements of a type.
- In particular, there should be some type I that *classifies* this shape, in the sense that maps $I \longrightarrow S$ correspond to abstract relations – or, to use a more geometric term, *paths* – between elements of S.
- In **RGph**, the role of such a path classifier is played by the *walking edge graph* $I := \{\mathbf{0} \rightarrow \mathbf{1}\}$
- Two key properties of I:

Sufficient Cohesion

How is this all related to parametricity?

- Cohesion lets us ask what is the *shape* of an abstract relation between elements of a type.
- In particular, there should be some type I that *classifies* this shape, in the sense that maps $\mathrm{I} \longrightarrow \mathrm{S}$ correspond to abstract relations – or, to use a more geometric term, *paths* – between elements of S.
- In **RGph**, the role of such a path classifier is played by the *walking edge graph* $\mathrm{I} := \{\mathbf{0} \longrightarrow \mathbf{1}\}$
- Two key properties of I:
  - It is *strictly bipointed*, i.e. $\mathbf{0} \neq \mathbf{1} \in \mathrm{I}$

Sufficient Cohesion

How is this all related to parametricity?

- Cohesion lets us ask what is the *shape* of an abstract relation between elements of a type.
- In particular, there should be some type I that *classifies* this shape, in the sense that maps $\mathrm{I} \longrightarrow \mathrm{S}$ correspond to abstract relations – or, to use a more geometric term, *paths* – between elements of S.
- In **RGph**, the role of such a path classifier is played by the *walking edge graph* $\mathrm{I} := \{\mathbf{0} \longrightarrow \mathbf{1}\}$
- Two key properties of I:
  - It is *strictly bipointed*, i.e. $\mathbf{0} \neq \mathbf{1} \in \mathrm{I}$
  - It is *connected*, i.e. $\int \mathrm{I} \simeq 1$

Sufficient Cohesion

How is this all related to parametricity?

- Cohesion lets us ask what is the *shape* of an abstract relation between elements of a type.
- In particular, there should be some type I that *classifies* this shape, in the sense that maps $\mathrm{I} \longrightarrow \mathrm{S}$ correspond to abstract relations – or, to use a more geometric term, *paths* – between elements of S.
- In **RGph**, the role of such a path classifier is played by the *walking edge graph* $\mathrm{I} := \{\mathbf{0} \longrightarrow \mathbf{1}\}$
- Two key properties of I:
    - It is *strictly bipointed*, i.e. $\mathbf{0} \neq \mathbf{1} \in \mathrm{I}$
    - It is *connected*, i.e. $\int \mathrm{I} \simeq 1$
- The existence of an object with these two properties is equivalent to what Lawvere called *sufficient cohesion*.

Sufficient Cohesion

Lemma: in (the internal language of) a sufficiently cohesive topos, all paths in discrete types are constant.

Sufficient Cohesion

Lemma: in (the internal language of) a sufficiently cohesive topos, all paths in discrete types are constant.
Proof: let S be a discrete type. A path in S is a function $f : I \longrightarrow S$. Since S is discrete, f factors as

$$I \xrightarrow{f_\flat} \flat S \xrightarrow{\epsilon_S} S$$

for some $f_\flat : I \longrightarrow \flat S$.

Sufficient Cohesion

Lemma: in (the internal language of) a sufficiently cohesive topos, all paths in discrete types are constant.

Proof: let S be a discrete type. A path in S is a function $f : \mathrm{I} \longrightarrow \mathrm{S}$. Since S is discrete, f factors as

$$\mathrm{I} \xrightarrow{f_\flat} \flat\mathrm{S} \xrightarrow{\epsilon_\mathrm{S}} \mathrm{S}$$

for some $f_\flat : \mathrm{I} \longrightarrow \flat\mathrm{S}$. But then since $\int \dashv \flat$, it follows that there is $f_\int : \int \mathrm{I} \longrightarrow \mathrm{S}$ such that

$$\begin{array}{ccc} \mathrm{I} & \xrightarrow{\eta_\mathrm{I}} & \int \mathrm{I} \\ {\scriptstyle f_\flat}\downarrow & \searrow^{f} & \downarrow{\scriptstyle f_\int} \\ \flat\mathrm{S} & \xrightarrow[\epsilon_\mathrm{S}]{} & \mathrm{S} \end{array}$$

where η is the unit for the monad $\int$. Then since I is connected, $\int \mathrm{I} \simeq 1$ and so f factors through 1, i.e. f is constant.

Path Types

- To represent such paths in type theory, we may borrow some ideas from cubical type theory and simplicial type theory.

Path Types

- To represent such paths in type theory, we may borrow some ideas from cubical type theory and simplicial type theory.
- We postulate an abstract interval type $\mathbb{I}$ with two points $\mathbf{0}, \mathbf{1} : \mathbb{I}$.

Path Types

- To represent such paths in type theory, we may borrow some ideas from cubical type theory and simplicial type theory.
- We postulate an abstract interval type I with two points $\mathbf{0}, \mathbf{1} : \mathrm{I}$.
- Given a family of types $i : \mathrm{I} \vdash \mathrm{S}(i)\ \mathsf{Type}$, a path from $s_0 : \mathrm{S}(\mathbf{0})$ to $s_1 : \mathrm{S}(\mathbf{1})$ is a dependent function

$$f : \prod_{i:\mathrm{I}} \mathrm{S}(i) \quad \text{such that} \quad f\mathbf{0} \equiv s_0 \text{ and } f\mathbf{1} \equiv s_1$$

Path Types

- To represent such paths in type theory, we may borrow some ideas from cubical type theory and simplicial type theory.
- We postulate an abstract interval type I with two points $\mathbf{0}, \mathbf{1} : \mathrm{I}$.
- Given a family of types $i : \mathrm{I} \vdash \mathrm{S}(i)\ \mathsf{Type}$, a path from $s_0 : \mathrm{S}(\mathbf{0})$ to $s_1 : \mathrm{S}(\mathbf{1})$ is a dependent function

$$f : \prod_{i:\mathrm{I}} \mathrm{S}(i) \quad \text{such that} \quad f\,\mathbf{0} \equiv s_0 \text{ and } f\,\mathbf{1} \equiv s_1$$

- Write $\mathsf{Path}_{i.\mathrm{S}(i)}(s_0, s_1)$ for the type of such paths.

Path Types

- To represent such paths in type theory, we may borrow some ideas from cubical type theory and simplicial type theory.
- We postulate an abstract interval type I with two points $\mathbf{0}, \mathbf{1} : \mathrm{I}$.
- Given a family of types $i : \mathrm{I} \vdash \mathrm{S}(i)\ \mathsf{Type}$, a path from $s_0 : \mathrm{S}(\mathbf{0})$ to $s_1 : \mathrm{S}(\mathbf{1})$ is a dependent function

$$f : \prod_{i:\mathrm{I}} \mathrm{S}(i) \quad \text{such that} \quad f\mathbf{0} \equiv s_0 \text{ and } f\mathbf{1} \equiv s_1$$

- Write $\mathsf{Path}_{i.\mathrm{S}(i)}(s_0, s_1)$ for the type of such paths.
- A type S is *path-discrete* if for all $s_0, s_1 : \mathrm{S}$, the canonical map $s_0 =_{\mathrm{S}} s_1 \rightarrow \mathsf{Path}_{i.\mathrm{S}}(s_0, s_1)$ is an equivalence.

Path Types

- To represent such paths in type theory, we may borrow some ideas from cubical type theory and simplicial type theory.
- We postulate an abstract interval type I with two points $\mathbf{0}, \mathbf{1} : \mathrm{I}$.
- Given a family of types $i : \mathrm{I} \vdash \mathrm{S}(i)\ \mathsf{Type}$, a path from $s_0 : \mathrm{S}(\mathbf{0})$ to $s_1 : \mathrm{S}(\mathbf{1})$ is a dependent function

$$f : \prod_{i:\mathrm{I}} \mathrm{S}(i) \quad \text{such that} \quad f\mathbf{0} \equiv s_0 \text{ and } f\mathbf{1} \equiv s_1$$

- Write $\mathsf{Path}_{i.\mathrm{S}(i)}(s_0, s_1)$ for the type of such paths.
- A type S is *path-discrete* if for all $s_0, s_1 : \mathrm{S}$, the canonical map $s_0 =_{\mathrm{S}} s_1 \to \mathsf{Path}_{i.\mathrm{S}}(s_0, s_1)$ is an equivalence.
- The above lemma says that, if a type is discrete, then it is path-discrete.

Graph Types

To make full use of the structure of sufficient cohesion, we also need some way to make use of the fact that I is strictly bipointed. For this purpose, we introduce *graph types*.

Graph Types

To make full use of the structure of sufficient cohesion, we also need some way to make use of the fact that I is strictly bipointed. For this purpose, we introduce *graph types*.

- Given S Type, a type family $x : \mathrm{S} \vdash \mathrm{T}(x)$ Type, and an element $i : \mathrm{I}$, the *graph type* $\mathsf{Gph1}^{i}_{x:\mathrm{S}}\mathrm{T}(x)$ is the type of dependent pairs whose second element exists only under the assumption that $i \equiv \mathbf{1}$, i.e.

$$(s, t) \quad \text{such that} \quad s : \mathrm{S} \text{ and } i \equiv \mathbf{1} \vdash t : \mathrm{T}(s)$$

Graph Types

To make full use of the structure of sufficient cohesion, we also need some way to make use of the fact that I is strictly bipointed. For this purpose, we introduce *graph types*.

- Given S Type, a type family $x : \mathrm{S} \vdash \mathrm{T}(x)$ Type, and an element $i : \mathrm{I}$, the *graph type* $\mathsf{Gph1}^{i}_{x:\mathrm{S}}\mathrm{T}(x)$ is the type of dependent pairs whose second element exists only under the assumption that $i \equiv \mathbf{1}$, i.e.

$$(s, t) \quad \text{such that} \quad s : \mathrm{S} \text{ and } i \equiv \mathbf{1} \vdash t : \mathrm{T}(s)$$

- In the case where $i \equiv \mathbf{0}$, we therefore have $\mathsf{Gph1}^{i}_{x:\mathrm{S}}\mathrm{T}(x) \simeq \mathrm{S}$, and we strengthen this equivalence into the following judgmental equalities:

$$\mathsf{Gph1}^{\mathbf{0}}_{x:\mathrm{S}}\mathrm{T}(x) \equiv \mathrm{S} \qquad \frac{p : \mathsf{Gph1}^{\mathbf{0}}_{x:\mathrm{S}}\mathrm{T}(x)}{\pi_1(p) \equiv p} \qquad \frac{(s,t) : \mathsf{Gph1}^{\mathbf{0}}_{x:\mathrm{S}}\mathrm{T}(x)}{(s,t) \equiv s}$$

The Polymorphic Identity

Lemma: given $\alpha : \prod_{X:\mathsf{Type}} X \to X$, for any *path-discrete* type A together with $x : A \vdash B(x)$ Type and $a : A$ with $b : B(a)$, the type $B(\alpha\ A\ a)$ is inhabited.

Three steps to prove parametricity:

1. Define a function $\mathsf{step1} : \prod_{i:I} \mathsf{Gph1}^{i}_{x:A} B(x)$ such that $\mathsf{step1}(\mathbf{0}) \equiv \alpha\ A\ a$

$$\mathsf{step1} := \lambda i : I.\ \alpha\ (\mathsf{Gph1}^{i}_{x:A} B(x))\ (a, b)$$

The Polymorphic Identity

Lemma: given $\alpha : \prod_{X:\mathsf{Type}} X \to X$, for any *path-discrete* type A together with $x : A \vdash B(x)$ Type and $a : A$ with $b : B(a)$, the type $B(\alpha\ A\ a)$ is inhabited.

Three steps to prove parametricity:

1. Define a function $\mathsf{step1} : \prod_{i:\mathrm{I}} \mathsf{Gph1}^{i}_{x:A} B(x)$ such that $\mathsf{step1}(\mathbf{0}) \equiv \alpha\ A\ a$

$$\mathsf{step1} := \lambda i : \mathrm{I}.\ \alpha\ (\mathsf{Gph1}^{i}_{x:A} B(x))\ (a, b)$$

2. Taking the second projection of $\mathsf{step1}(\mathbf{1})$ gives $\mathsf{step2} : B(\pi_1(\mathsf{step1}(\mathbf{1})))$

The Polymorphic Identity

Lemma: given $\alpha : \prod_{X:\mathsf{Type}} X \to X$, for any *path-discrete* type A together with $x : A \vdash B(x)$ Type and $a : A$ with $b : B(a)$, the type $B(\alpha\ A\ a)$ is inhabited.

Three steps to prove parametricity:

1. Define a function $\mathsf{step1} : \prod_{i:I} \mathsf{Gph1}^{i}_{x:A} B(x)$ such that $\mathsf{step1}(\mathbf{0}) \equiv \alpha\ A\ a$

$$\mathsf{step1} := \lambda i : I.\ \alpha\ (\mathsf{Gph1}^{i}_{x:A} B(x))\ (a, b)$$

2. Taking the second projection of $\mathsf{step1}(\mathbf{1})$ gives $\mathsf{step2} : B(\pi_1(\mathsf{step1}(\mathbf{1})))$
3. Taking the first projection of $\mathsf{step1}(i)$ for $i : I$ gives a path $\mathsf{step3} : \mathsf{Path}_{i.A}(\alpha\ A\ a,\ \pi_1(\mathsf{step1}(\mathbf{1})))$, and since A is path-discrete, this yields an identity $\alpha\ A\ a =_A \pi_1(\mathsf{step1}(\mathbf{1}))$, along which we can transport $\mathsf{step2}$ to obtain an inhabitant of $B(\alpha\ A\ a)$. □

Applications in HoTT

- This same technique can be used to derive induction principles for inductive and *higher* inductive types from their recursors alone. The derivation is very straightforward, following essentially the same **three steps to prove parametricity** as above.

Applications in HoTT

- This same technique can be used to derive induction principles for inductive and *higher* inductive types from their recursors alone. The derivation is very straightforward, following essentially the same **three steps to prove parametricity** as above.
- Previously, induction principles could be derived from recursors using the Awodey-Frey-Speight strategy of restricting to instances of recursors satisfying certain higher-categorical *coherence conditions*. However, these conditions quickly grow in complexity and become intractable to work with. This is essentially an instance of the *coherence problem* in HoTT.

Applications in HoTT

- This same technique can be used to derive induction principles for inductive and *higher* inductive types from their recursors alone. The derivation is very straightforward, following essentially the same **three steps to prove parametricity** as above.
- Previously, induction principles could be derived from recursors using the Awodey-Frey-Speight strategy of restricting to instances of recursors satisfying certain higher-categorical *coherence conditions*. However, these conditions quickly grow in complexity and become intractable to work with. This is essentially an instance of the *coherence problem* in HoTT.
- The approach to this problem via internal parametricity in cohesive HoTT suffers none of these defects, and easily handles examples such as the circle, for which the analogous Awodey-Frey-Speight encoding is already quite complex.

...and Beyond?

- These results have all been formalized in Agda using the `--cohesion` flag.

...and Beyond?

- These results have all been formalized in Agda using the `--cohesion` flag.
  - anyone who's interested can import the code and start using it to prove parametricity theorems in Agda *today*: https://github.com/cbaberle/Parametricity-via-Cohesion

...and Beyond?

- These results have all been formalized in Agda using the `--cohesion` flag.
  - anyone who's interested can import the code and start using it to prove parametricity theorems in Agda *today*: https://github.com/cbaberle/Parametricity-via-Cohesion
- In this talk we have mainly considered *unary* parametricity, but this approach handles binary and n-ary parametricity just as well.
  - Jason Reed has a nice formalization of K-ary parametricity for any type K with decidable equality: https://github.com/jcreedcmu/aberle-parametricity-exercise/blob/main/ExerciseN.agda

...and Beyond?

- These results have all been formalized in Agda using the `--cohesion` flag.
  - anyone who's interested can import the code and start using it to prove parametricity theorems in Agda *today*: https://github.com/cbaberle/Parametricity-via-Cohesion
- In this talk we have mainly considered *unary* parametricity, but this approach handles binary and n-ary parametricity just as well.
  - Jason Reed has a nice formalization of K-ary parametricity for any type K with decidable equality: https://github.com/jcreedcmu/aberle-parametricity-exercise/blob/main/ExerciseN.agda
- Hope that the account of parametricity via cohesion – or some suitable generalization thereof – can serve as a unifying framework for these and other applications of internal parametricity in dependent type theory.

...and Beyond?

- These results have all been formalized in Agda using the `--cohesion` flag.
  - anyone who's interested can import the code and start using it to prove parametricity theorems in Agda *today*: https://github.com/cbaberle/Parametricity-via-Cohesion
- In this talk we have mainly considered *unary* parametricity, but this approach handles binary and *n*-ary parametricity just as well.
  - Jason Reed has a nice formalization of K-ary parametricity for any type K with decidable equality: https://github.com/jcreedcmu/aberle-parametricity-exercise/blob/main/ExerciseN.agda
- Hope that the account of parametricity via cohesion – or some suitable generalization thereof – can serve as a unifying framework for these and other applications of internal parametricity in dependent type theory.
  - Further work: internal parametricity for linear programs (in some form of linear dependent type theory?)

Thank you!