Univalent Relational Parametricity

A way to automaticaly transport proof

Slides : Thomas Lamiaux, Ens Paris-Saclay
Work : Thomas Lamiaux, Ens Paris-Saclay
Nicolas Tabareau, Inria, Galinette team

April 6, 2022

Introduction

Part 1

The History of Parametricity

- Reynolds, 1983 : The Abstraction Theorem
- Wadler, 1989 : The introduction to Parametricity
- 1990 - 2020 : A bunch of things
- Bernardy et al, 2012 : Internalising Parametricity

Part 2

Usual attempts to transport proofs

- Parametricity
- The Hetereogenous Parametricity
- The Univalence Axiom

Part 3

Univalent Relational Parametricity

- Univalent Parametricity : Definitions, interests and limits
- Univalent Relational Parametricity
- Proving the Abstraction Theorem for CC_ω
- The Abstraction Theorem for Parametric Inductive Types

The History of Parametricity

The History of Parametricity

Reynolds, 1983 : The Abstraction Theorem

Reynolds original approach

"Type structure is a syntatic discipline for enforcing levels of abstraction"

Reynolds original approach

"Type structure is a syntatic discipline for enforcing levels of abstraction"

"Types are not limited to computation. Thus they should be explicable without invoking constructs, such as Scott's domains, that are peculiar to the theory of computation"

Reynolds original approach

"Type structure is a syntatic discipline for enforcing levels of abstraction"

"Types are not limited to computation. Thus they should be explicable without invoking constructs, such as Scott's domains, that are peculiar to the theory of computation"

- Introduces the system T with boolean, type variable and product in a set setting

Reynolds original approach

"Type structure is a syntatic discipline for enforcing levels of abstraction"

"Types are not limited to computation. Thus they should be explicable without invoking constructs, such as Scott's domains, that are peculiar to the theory of computation"

- Introduces the system T with boolean, type variable and product in a set setting
- Define a set model of the system T and traduces the functions by themselves.

Reynolds original approach

"Type structure is a syntatic discipline for enforcing levels of abstraction"

"Types are not limited to computation. Thus they should be explicable without invoking constructs, such as Scott's domains, that are peculiar to the theory of computation"

- Introduces the system T with boolean, type variable and product in a set setting
- Define a set model of the system T and traduces the functions by themselves.
- On this defines a relation semantics that follow the previous one.
- Abstraction theorem : if two sets asignements are related, then the interpretation are

The History of Parametricity

Wadler, 1989 : The introduction to Parametricity

The System F

Types :

$$T ::= X \mid A \to B \mid \forall X.A$$

Terms :

$$t ::= x \mid \lambda(x : X).\ t \mid uv \mid \Lambda X.\ t \mid t_U$$

Reductions :

$$\beta_t \mid \eta_t \mid \beta_T \mid \eta_T$$

The System F

Types :

$$T ::= X \mid A \to B \mid \forall X.A$$

Terms :

$$t ::= x \mid \lambda(x : X).\ t \mid uv \mid \Lambda X.\ t \mid t_U$$

Reductions :

$$\beta_t \mid \eta_t \mid \beta_T \mid \eta_T$$

Properties : Confluence, Strongly Normalising, $F \simeq HA_2$

Parametricity for the system F

We write $\mathcal{A} : \mathrm{A} \Leftrightarrow \mathrm{A}'$ relation between A and A'. We'll asumme $\mathcal{A} : A \Leftrightarrow A'$ and $\mathcal{B} : B \Leftrightarrow B'$.

Parametricity for the system F

We write $\mathcal{A} : \mathrm{A} \Leftrightarrow \mathrm{A}'$ relation between A and A'. We'll asumme $\mathcal{A} : A \Leftrightarrow A'$ and $\mathcal{B} : B \Leftrightarrow B'$.

- For *Bin* and $\mathbb{N}$, the relation is the equality.

Parametricity for the system F

We write $\mathcal{A} : \mathrm{A} \Leftrightarrow \mathrm{A}'$ relation between A and A'. We'll asumme $\mathcal{A} : A \Leftrightarrow A'$ and $\mathcal{B} : B \Leftrightarrow B'$.

- For *Bin* and $\mathbb{N}$, the relation is the equality.
- Given positive inductive types, the parametricity is pontwise on the constructors.
  - $((x, y),\ (x', y')) \in \mathcal{A} \times \mathcal{B}$ iff $(x, x') \in \mathcal{A} \wedge (y, y') \in \mathcal{B}$
  - $([], []) \in \mathcal{A}^*$ and $((a :: l), (a' :: l)) \in \mathcal{A}^*$ iif $(a, a) \in \mathcal{A} \wedge (l, l') \in \mathcal{A}^*$

Parametricity for the system F

We write $\mathcal{A} : \mathrm{A} \Leftrightarrow \mathrm{A}'$ relation between A and A'. We'll asumme $\mathcal{A} : A \Leftrightarrow A'$ and $\mathcal{B} : B \Leftrightarrow B'$.

- For *Bin* and $\mathbb{N}$, the relation is the equality.
- Given positive inductive types, the parametricity is pontwise on the constructors.
  - $((x, y),\ (x', y')) \in \mathcal{A} \times \mathcal{B}$ iff $(x, x') \in \mathcal{A} \wedge (y, y') \in \mathcal{B}$
  - $([], []) \in \mathcal{A}^*$ and $((a :: l), (a' :: l)) \in \mathcal{A}^*$ iif $(a, a) \in \mathcal{A} \wedge (l, l') \in \mathcal{A}^*$
- $(f, f') \in \mathcal{A} \to \mathcal{B}$ iff $\forall (a, a') \in \mathcal{A},\ (f\ a,\ f'\ a') \in \mathcal{B}$

The special case of Polymorphism

For the moment there is only one relation the "pointwise equality" and it is homogenous !

The special case of Polymorphism

For the moment there is only one relation the "pointwise equality" and it is homogenous !
The trick is in polymorphism function $\forall X.F(X)$:

$$(g, g') \in \forall \mathcal{X}.F(\mathcal{X}) \text{ iff } \forall (\mathcal{A} : A \Leftrightarrow A'),\ (g_A, g'_{A'}) \in \mathcal{F}(\mathcal{A})$$

The special case of Polymorphism

For the moment there is only one relation the "pointwise equality" and it is homogenous !
The trick is in polymorphism function $\forall X.F(X)$:

$$(g, g') \in \forall \mathcal{X}.F(\mathcal{X}) \text{ iff } \forall(\mathcal{A} : A \Leftrightarrow A'),\ (g_A, g'_{A'}) \in \mathcal{F}(\mathcal{A})$$

Yet $a : A \to A'$ is a special case of relation between $a : \mathcal{A} \Leftrightarrow \mathcal{A}'$.
The relation generated are much broader than equality

Reynold's Abstraction Theorem

Theorem (Abstraction Theorem)

For all closed term t of types T then $(t, t) \in \mathcal{T}$

Proof.

Models !

$\square$

Free Theorem : The identity

There is only one function of type $\forall X.X \rightarrow X$:

Free Theorem : The identity

There is only one function of type $\forall X.X \to X$:
Let $r : \forall X.X \to X$ then

$$(r, r) \in \forall \mathcal{X}.\, \mathcal{X} \to \mathcal{X}$$

$$\forall (\mathcal{A} : A \Leftrightarrow A'),\ (r_A, r_{A'}) \in \mathcal{A} \to \mathcal{A}$$

$$\forall (\mathcal{A} : A \Leftrightarrow A'),\ \forall (a, a') \in \mathcal{A}^*,\ (r_A\ a,\ r_{A'}\ a') \in \mathcal{A}$$

Free Theorem : The identity

There is only one function of type $\forall X.X \rightarrow X$:
Let $r : \forall X.X \rightarrow X$ then

$$(r, r) \in \forall \mathcal{X}.\, \mathcal{X} \rightarrow \mathcal{X}$$
$$\forall (\mathcal{A} : A \Leftrightarrow A'),\ (r_A, r_{A'}) \in \mathcal{A} \rightarrow \mathcal{A}$$
$$\forall (\mathcal{A} : A \Leftrightarrow A'),\ \forall (a, a') \in \mathcal{A}^*,\ (r_A\ a,\ r_{A'}\ a') \in \mathcal{A}$$

So in the special case of $\mathcal{A} = \{(x, y),\ x = a\}$ with $a : A$:

$$\forall (a : A).\ a' = a \Leftrightarrow (r_A\ a') = a$$

Free Theorem : The identity

There is only one function of type $\forall X.X \to X$:
Let $r : \forall X.X \to X$ then

$$(r, r) \in \forall \mathcal{X}.\ \mathcal{X} \to \mathcal{X}$$
$$\forall(\mathcal{A} : A \Leftrightarrow A'),\ (r_A, r_{A'}) \in \mathcal{A} \to \mathcal{A}$$
$$\forall(\mathcal{A} : A \Leftrightarrow A'),\ \forall(a, a') \in \mathcal{A}^*,\ (r_A\ a,\ r_{A'}\ a') \in \mathcal{A}$$

So in the special case of $\mathcal{A} = \{(x, y),\ x = a\}$ with $a : A$:

$$\forall(a : A).\ a' = a \Leftrightarrow (r_A\ a') = a$$

So in $a : A$:

$$r_A\ a = a \text{ ie } r_A = id_A \text{ ie } r = id$$

Free Theorem : Invariant under rearrengement

Let $r : \forall X.X^* \to X^*$ then

$$(r, r) \in \forall \mathcal{X}.\ \mathcal{X}^* \to \mathcal{X}^*$$
$$\forall (\mathcal{A} : A \Leftrightarrow A'),\ (r_A, r_{A'}) \in \mathcal{A}^* \to \mathcal{A}^*$$
$$\forall (\mathcal{A} : A \Leftrightarrow A'),\ \forall (l, l') \in \mathcal{A}^*,\ (r_A\ l,\ r_{A'}\ l') \in \mathcal{A}$$

Free Theorem : Invariant under rearrengement

Let $r : \forall X. X^* \to X^*$ then

$$(r, r) \in \forall \mathcal{X}.\ \mathcal{X}^* \to \mathcal{X}^*$$
$$\forall (\mathcal{A} : A \Leftrightarrow A'),\ (r_A, r_{A'}) \in \mathcal{A}^* \to \mathcal{A}^*$$
$$\forall (\mathcal{A} : A \Leftrightarrow A'),\ \forall (l, l') \in \mathcal{A}^*,\ (r_A\ l,\ r_{A'}\ l') \in \mathcal{A}$$

So in the special case of $\mathcal{A} = a : \mathcal{A} \to \mathcal{A}'$:

$$a^*\ (r_A\ l) = r_{A'}\ (a^*\ l)$$

Free Theorem : Other compositional results

$$filter : \forall X.(X \to Bool) \to X^* \to X^*$$

$$a^* \circ (filter_A\ p) = (sort\ p') \circ a^*$$

$$sort : \forall X.(X \to X \to Bool) \to X^* \to X^*$$

$$\forall (x, y : A).$$

$$(x < y) = (a\ x <' a\ y) \Rightarrow a^* \circ (sort_A\ \ <) = (sort\ \ <') \circ a^*$$

$$K : \forall X.\forall Y.X \to Y \to Y$$

$$a \circ (K_{AB}\ x\ y) = K_{A'B'}\ (a\ x)\ (b\ y)$$

Free Theorem : Some interresting results

The different results can be "sorted" in three kinds :

- Unicity results :
  - $\forall X.\ X$ is empty
  - $\forall X.\ X \to X$ unique inhabitant is *Id*
  - $\forall X.\forall Y.(X \to Y) \to (X^* \to Y^*)$, every function is the composition of rearrengement and lifting
- Composition results : previous slides
- $A \simeq \forall X.(A \to X) \to X$

The History of Parametricity

1990 - 2020 : A bunch of things

Extend Parametricity to other type system

- Takeuti, 2004 (unpublished) : Try to extend it to CC
- Vytiniotis and Weirich, 2010 : Extension to $F\omega$
- Krishnaswami and Dreyer, 2013 : Parametricity and extensional type theory
- Bernardy et al, 2015 : Adding the parametricity internaly to MLLT and giving a pre-sheaf model
- Cavaballo and Harper, 2020 : Mixing cubical type theory and parametricity
- ...

Syntax and Semantics

In the classical setting models are used, after 2012 it is formulas that represent the program.

- Mairson, 1991 : introduces the concepts
- Plotkin and Abadi 1993 : Define a logic to express parametricity for the system F
- Wadler, 2007 : The abstraction theorem can be seen as projection of $HA2$ in the system F
- ...

Parametricity for data refinement problem

- Magaud and Bertot, 2000 : Transporting proof for a librairy to another using isomorphism, break on type dependancy
- Cohen, Dénès and Mörtberg : CoqEAL uses parametricty to transport proof dealing with isomorphism and quotient. Break on type dependancy.
- ...

The History of Parametricity

Bernardy et al, 2012 : Internalising Parametricity

$CC\omega$

CC_ω is the PTS that is behind Coq :

- A hierarchy of universes $\mathcal{U}_\rangle$ with a type of proposition $\star$
- The following typing rules $\star : \mathcal{U}_0$ and $\mathcal{U}_i : \mathcal{U}_{i+1}$
- $\star$ is impredicative otherwise the max rule applies

Parametricity for $CC\omega$

$$\llbracket \mathcal{U}_i \rrbracket_p := \prod_{A,B:\mathcal{U}_i} A \to B \to \mathcal{U}_{i+1}$$

$$\left\llbracket \prod_{a:A} B\ a \right\rrbracket_p := \prod_{\substack{f:\prod_{a:A} B\ a \\ g:\prod_{a':A'} B'\ a}} \prod_{\substack{a:A \\ a':A' \\ a^\epsilon:\llbracket A \rrbracket_p\ a\ a'}} \llbracket B\ a \rrbracket_p\ f(a)\ g(a)$$

$$\llbracket x \rrbracket_p := x^\epsilon$$

$$\llbracket \lambda\ (x:A).\ t \rrbracket_p := \lambda\ (x:A)(x':A')(x^\epsilon : \llbracket A \rrbracket\ a\ a').\ \llbracket t \rrbracket$$

$$\llbracket uv \rrbracket_p := \llbracket t \rrbracket\ u\ u'\ \llbracket u \rrbracket$$

$$\llbracket \Gamma,\ x:A \rrbracket := \llbracket \Gamma \rrbracket_p,\ x:A,\ x':A',\ x^\epsilon : \llbracket A \rrbracket\ x\ x')$$

The Abstraction Theorem

Theorem (The Abstraction Theorem)

If $\Gamma \vdash a : A$ *then* $[\![\Gamma]\!] \vdash [\![a]\!] : [\![A]\!]\ a\ a'$

Proof.

Simple induction $\square$

The special cases of universes typing rules :

$$\mathcal{U}_i : \mathcal{U}_{i+1} \implies [\![\mathcal{U}_i]\!] : [\![\mathcal{U}_{i+1}]\!]\ \ \mathcal{U}_i\ \ \mathcal{U}_i$$

$$\prod_{a:A} B\ a : \mathcal{U}_i \implies \left[\!\!\left[\prod_{a:A} B\ a\right]\!\!\right] : [\![\mathcal{U}_i]\!]\ \ \prod_{a:A} B\ a\ \ \prod_{a':A'} B'\ a'$$

Usual attempts to transport proofs

Usual attempts to transport proofs

Parametricity

The Anticipation Problem

The parametricity is reflective and homogenous so it not possible to directly relate $+$ and $+_{bin}$.
So one need to find an abstratction P that can be instantiated with $\mathbb{N}$ or Bin.

$$P : \sum_{A:\mathcal{U}_i} \sum_{0_A:A} \sum_{S_A:A \to A} (\prod_{C:\mathcal{U}_i} C \to (B \to C \to C) \to B \to C)$$

such that $(\mathbb{N}, 0, S, Rec_{\mathbb{N}}) : P$ and $(Bin, 0_{Bin}, S_{Bin}, NRec_{Bin}) : P$

$$+_A : P.1 \to P.1 \to P.1$$

$$+_A : P.4\ P.1\ (\lambda x \to x)\ (\lambda\ x\ g \to P.3\ (g\ x))$$

Then use the parametricity and instantiate it with the type and equivalence

The Computation Problem

Not every term can be express using a general framework :

$$P : \mathbb{N} \to \mathcal{U}_0$$
$$P := Rec_{\mathbb{N}}\ (\lambda\ \_ \to \mathcal{U}_0)\ (0 = 0)\ (\lambda\ \_\ \_ \to \bot)$$
$$Diff\ \ (n : \mathbb{N})\ \ (p : 0 = S\ n)\ \ : \bot$$
$$Diff\ := Rec_{Eq}\ (\mathbb{N}\ 0\ (\lambda\ n\ \_\ \to P\ n)\ refl)\ \ (S\ n)\ \ p$$

This type check only because $P\ (S\ n) \equiv \bot$.

The Computation Problem

Not every term can be express using a general framework :

$$P : \mathbb{N} \to \mathcal{U}_0$$
$$P := Rec_{\mathbb{N}}\ (\lambda\ \_ \to \mathcal{U}_0)\ (0 = 0)\ (\lambda\ \_\ \_ \to \bot)$$
$$Diff\ \ (n : \mathbb{N})\ \ (p : 0 = S\ n)\ \ : \bot$$
$$Diff\ := Rec_{Eq}\ (\mathbb{N}\ 0\ (\lambda\ n\ \_\ \ \to P\ n)\ refl)\ \ (S\ n)\ \ p$$

This type check only because $P\ (S\ n) \equiv \bot$.

This wouldn't be the case for $NRec_{Bin}$. And so it not possible to define it for a abstraction of the types P.

Usual attempts to transport proofs

The Hetereogenous Parametricity

Adding a context

The issue with the previous framework is that it doesn't allow to relate 0 and 0_{Bin}. However, it is possible to do by adding a global context defined such that :

$$\Xi_1 = (x^\circ : A^\circ,\ x^\bullet : A^\bullet,\ x^\otimes : [\![A^\circ]\!]\ x^\circ\ x^\bullet)$$
$$\Xi_{n+1} = \Xi_n,\ (x^\circ_{n+1} : A^\circ_{n+1},\ x^\bullet_{n+1} : A^\bullet_{n+1},\ x^\otimes_{n+1} : [\![A^\circ_{n+1}]\!]^{\Xi_n}\ x^\circ_{n+1}\ x^\bullet_{n+1})$$

Where $[\![\_]\!]^{\Xi}$ is the classic parametricity with A° replaced by $A^\bullet$ and $[\![x]\!]$ by $x^\otimes$.

White Box Translation

The first and second projections can be seen as contexts $|\Xi|_\circ$ and $|\Xi|_\bullet$.
In which it possible to define the White Box Translation $\uparrow_\Box$ to be the identity function yet replacing x° by $x^\bullet$.

Theorem (The White Box Fundamental Property)

If $|\Xi|_\circ \vdash a : A$ *then :*

- $|\Xi|_\bullet \vdash (\uparrow_\Box a) : (\uparrow_\Box A)$
- $|\Xi| : [\![A]\!]_p^\Xi \; a \; (\uparrow \Box a)$

Over $\mathbb{N}$ and *Bin* 1/3

One point of transporting proofs is to be able to switch from representation, for instance the natural to the binary.
Given an equivalence $e : \mathbb{N} \simeq Bin$, we denote $\uparrow_{\mathbb{N}}: Bin \to \mathbb{N}$.
Then defining $R := \prod_{n:\mathbb{N}} \prod_{b:Bin} n = \uparrow_{\mathbb{N}} b$, we have $(\mathbb{N}, Bin, R)$.

$$0^{\otimes} : [\![\mathbb{N}]\!] \; 0 \; 0_{Bin}$$

$$S^{\otimes} : [\![\mathbb{N} \to \mathbb{N}]\!] \; S \; S_{Bin}$$

Over $\mathbb{N}$ and *Bin* 2/3

One point of transporting proofs is to be able to switch from representation, for instance the natural to the binary.
Given an equivalence $e : \mathbb{N} \simeq Bin$, we denote $\uparrow_{\mathbb{N}}: Bin \to \mathbb{N}$.
Then defining $R := \prod_{n:\mathbb{N}} \prod_{b:Bin} n =\uparrow_{\mathbb{N}} b$, we have $(\mathbb{N}, Bin, R)$.

$$0^{\otimes} : 0 =\uparrow_{\mathbb{N}} 0_{Bin}$$

$$S^{\otimes} : \prod_{\substack{n:\mathbb{N} \\ b:Bin}} n =\uparrow_{\mathbb{N}} b \to S\ n =\uparrow_{\mathbb{N}} (S_{Bin}\ b)$$

Over $\mathbb{N}$ and *Bin* 3/3

One point of transporting proofs is to be able to switch from representation, for instance the natural to the binary.
Given an equivalence $e : \mathbb{N} \simeq Bin$, we denote $\uparrow_{\mathbb{N}} : Bin \to \mathbb{N}$.
Then defining $R := \prod_{n:\mathbb{N}} \prod_{b:Bin} n =\uparrow_{\mathbb{N}} b$, we have $(\mathbb{N}, Bin, R)$.

$$0^{\otimes} : 0 =\uparrow_{\mathbb{N}} 0_{Bin}$$
$$S^{\otimes} : \prod_{b:Bin} S\ (\uparrow_{\mathbb{N}} b) =\uparrow_{\mathbb{N}} (S_{Bin}\ b)$$

Over $\mathbb{N}$ and *Bin* 3/3

One point of transporting proofs is to be able to switch from representation, for instance the natural to the binary.
Given an equivalence $e : \mathbb{N} \simeq Bin$, we denote $\uparrow_{\mathbb{N}}: Bin \to \mathbb{N}$.
Then defining $R := \prod_{n:\mathbb{N}} \prod_{b:Bin} n =\uparrow_{\mathbb{N}} b$, we have $(\mathbb{N}, Bin, R)$.

$$0^{\otimes} : 0 =\uparrow_{\mathbb{N}} 0_{Bin}$$
$$S^{\otimes} : \prod_{b:Bin} S\ (\uparrow_{\mathbb{N}} b) =\uparrow_{\mathbb{N}} (S_{Bin}\ b)$$

Then in $((\mathbb{N}, Bin, R),\ (0, 0_{Bin}, 0^{\times}),\ (S, S_{Bin}, S^{\otimes}))$ it is possible to relate $Rec_{\mathbb{N}}$ and $\mathbb{N} - Rec_{Bin}$

Translating the addition 1/2

Using the WB translation, it is possible to transport operators :

$$\uparrow_{\Box} + : Bin \rightarrow Bin \rightarrow Bin$$

Translating the addition 1/2

Using the WB translation, it is possible to transport operators :

$$\uparrow_{\square} + : Bin \to Bin \to Bin$$

And the parametricity gives us :

$$[\![+]\!] : [\![\mathbb{N} \to \mathbb{N} \to N]\!] \; + \; +_{Bin}$$
$$[\![+]\!] : \prod_{\substack{n:\mathbb{N} \\ b:Bin}} n =\uparrow_{\mathbb{N}} b \to \prod_{\substack{n':\mathbb{N} \\ b':Bin}} n' =\uparrow_{\mathbb{N}} \to n + m =\uparrow_{\mathbb{N}} (b +_{Bin} b')$$

Translating the addition 2/2

Using the WB translation, it is possible to transport operators :

$$\uparrow_{\square} + : Bin \to Bin \to Bin$$

And the parametricity gives us :

$$[\![+]\!] : [\![\mathbb{N} \to \mathbb{N} \to N]\!] \ + \ +_{Bin}$$
$$[\![+]\!] : \prod_{b:Bin} \prod_{b':Bin} (\uparrow_{\mathbb{N}} b) + (\uparrow \mathbb{N} b') = \uparrow_{\mathbb{N}} (b +_{Bin} b')$$

Transporting Proofs

How to transport proofs and not just operators like :

$$\prod_{n,m:\mathbb{N}} n + m = m + n$$

Transporting Proofs

How to transport proofs and not just operators like :

$$\prod_{n,m:\mathbb{N}} n + m = m + n$$

Need to add the support of basic types like 0, 1, $\mathbb{N}$, *Bin* but also some type constructors like $=$, *list*, *vec*...

Transporting Proofs

How to transport proofs and not just operators like :

$$\prod_{n,m:\mathbb{N}} n + m = m + n$$

Need to add the support of basic types like 0, 1, $\mathbb{N}$, *Bin* but also some type constructors like $=$, *list*, *vec*...

Possible without to much trouble, however the translation of proofs still fails on dependent types.

Usual attempts to transport proofs

The Univalence Axiom

Using Equivalences

Just using the equivalence properties doesn't suffice :

$$(\uparrow_{\mathbb{N}} b) +_{\mathbb{N}} (\uparrow_{\mathbb{N}} b') = (\uparrow_{\mathbb{N}} b') +_{\mathbb{N}} (\uparrow_{\mathbb{N}} b)$$
$$\uparrow_{Bin} ((\uparrow_{\mathbb{N}} b) +_{\mathbb{N}} (\uparrow_{\mathbb{N}} b')) = \uparrow_{Bin} ((\uparrow_{\mathbb{N}} b') +_{\mathbb{N}} (\uparrow_{\mathbb{N}} b))$$
$$(\uparrow_{Bin}\uparrow_{\mathbb{N}} b) +_{Bin} (\uparrow_{Bin}\uparrow_{\mathbb{N}} b') = (\uparrow_{Bin}\uparrow_{\mathbb{N}} b') +_{Bin} (\uparrow_{Bin}\uparrow_{\mathbb{N}} b)$$
$$b +_{Bin} b' = b' +_{Bin} b$$

Using Equivalences

Just using the equivalence properties doesn't suffice :

$$(\uparrow_{\mathbb{N}} b) +_{\mathbb{N}} (\uparrow_{\mathbb{N}} b') = (\uparrow_{\mathbb{N}} b') +_{\mathbb{N}} (\uparrow_{\mathbb{N}} b)$$

$$\uparrow_{Bin} ((\uparrow_{\mathbb{N}} b) +_{\mathbb{N}} (\uparrow_{\mathbb{N}} b')) = \uparrow_{Bin} ((\uparrow_{\mathbb{N}} b') +_{\mathbb{N}} (\uparrow_{\mathbb{N}} b))$$

$$(\uparrow_{Bin}\uparrow_{\mathbb{N}} b) +_{Bin} (\uparrow_{Bin}\uparrow_{\mathbb{N}} b') = (\uparrow_{Bin}\uparrow_{\mathbb{N}} b') +_{Bin} (\uparrow_{Bin}\uparrow_{\mathbb{N}} b)$$

$$b +_{Bin} b' = b' +_{Bin} b$$

Some issue with dependancy, for instance $P(g(f\ a))$ is not the same type as $P\ a$.

Using the univalence to translate proof

Theorem (Lifting an equivalence)

Given two types A, B and P a family of types, then
$A \simeq B \to P\ A \simeq P\ B$

Proof.

$A \simeq B \to A = B \to P\ A = P\ B \to P\ A \simeq P\ B$ □

How to prove that "$+ = +_{Bin}$" ?

Using the univalence to translate proof

Theorem (Lifting an equivalence)

Given two types A, B and P a family of types, then
$A \simeq B \to P\ A \simeq P\ B$

Proof.

$A \simeq B \to A = B \to P\ A = P\ B \to P\ A \simeq P\ B$ □

How to prove that "$+ = +_{Bin}$" ?

Actually possible to prove the equality between $(\mathbb{N}, +_{\mathbb{N}})$ and $(Bin, +\_Bin)$ in $\Sigma_{A:\mathcal{U}_0} A \to A \to A$

Transport

Then given $p : (\mathbb{N}, +_{\mathbb{N}}) = (Bin, +_{Bin})$ when can do do the following :

$$P\_comm = \lambda\, X\ \_.\ \prod_{x,y:A} X.2\ x\ y = X.2\ y\ x$$

$$+_{Bin}\_comm = \mathrm{transport}^{P\_comm}\ (p, +_{\mathbb{N}}\_comm)$$

In the cubical Setting

In the cubical setting there is heterogenous paths :

$$\mathit{addp} : \mathrm{PathP}$$
$$(\lambda i \to \mathbb{N} \equiv \mathit{Bin}\ i \to \mathbb{N} \equiv \mathit{Bin}\ i \to \mathbb{N} \equiv \mathit{Bin}\ i)$$
$$+_{\mathbb{N}}\ \ +_{\mathit{Bin}}$$

Then it possible to transport along this path :

$$\mathrm{transport}$$
$$(\lambda i \to (x, y : \mathbb{N} \equiv \mathit{Bin}\ i) \to x\ (\mathit{addp}\ i)\ y \equiv y\ (\mathit{addp}\ i)\ x)$$
$$+_{\mathbb{N}}\_\mathit{comm}$$

Univalent Relational Parametricity

Univalent Relational Parametricity

Univalent Parametricity : Definitions, interests and limits

What is the goal of Univalent Parametricity ?

The goal is to stength parametricity over types such as two types are parametric iff they are related and equivalent.

What is the goal of Univalent Parametricity ?

The goal is to stength parametricity over types such as two types are parametric iff they are related and equivalent.

- Define $[\![\mathcal{U}_i]\!] := \prod_{A:\mathcal{U}_i} \prod_{B:\mathcal{U}_i} A \simeq B$.
  But then the typing rules is no long verified :

$$[\![\mathcal{U}_i]\!] : [\![\mathcal{U}_{i+1}]\!]\ \mathcal{U}_i\,\mathcal{U}_i$$

$$\prod_{A,B:\mathcal{U}_i} A \simeq B : \mathcal{U}_i \simeq \mathcal{U}_i$$

What is the goal of Univalent Parametricity ?

The goal is to stength parametricity over types such as two types are parametric iff they are related and equivalent.

- Define $[\![\mathcal{U}_i]\!] := \prod_{A:\mathcal{U}_i} \prod_{B:\mathcal{U}_i} A \simeq B$.
  But then the typing rules is no long verified :

$$[\![\mathcal{U}_i]\!] : [\![\mathcal{U}_{i+1}]\!] \; \mathcal{U}_i \; \mathcal{U}_i$$

$$\prod_{A,B:\mathcal{U}_i} A \simeq B : \mathcal{U}_i \simeq \mathcal{U}_i$$

- Ask for a relation and an equivalence between A and A'
  No connections between the two $\implies$ won't rise to CIC because of equality

Solution ?

Therefore we need a coherence condition between $R : A \to B \to \mathcal{U}_i$ and $e : A \simeq B$

$$ecoh := \prod_{\substack{a:A \\ b:B}} R\ a\ b \simeq (a =\uparrow_e b)$$

Solution ?

Therefore we need a coherence condition between $R : A \to B \to \mathcal{U}_i$ and $e : A \simeq B$

$$ecoh := \prod_{\substack{a:A \\ b:B}} R\ a\ b \simeq (a =\uparrow_e b)$$

Then the parametricity of $\mathcal{U}_i$ is :

$$[\![\mathcal{U}_i]\!] := \prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \sum_{e:A\simeq B} \prod_{\substack{a:A \\ b:B}} R\ a\ b \simeq a =\uparrow_e b$$

Solution ?

Therefore we need a coherence condition between $R : A \to B \to \mathcal{U}_i$ and $e : A \simeq B$

$$ecoh := \prod_{\substack{a:A \\ b:B}} R\ a\ b \simeq (a =\uparrow_e b)$$

Then the parametricity of $\mathcal{U}_i$ is :

$$[\![\mathcal{U}_i]\!] := \prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \sum_{e:A\simeq B} \prod_{\substack{a:A \\ b:B}} R\ a\ b \simeq a =\uparrow_e b$$

Yet we want to be able to prove :

$$[\![\mathcal{U}_i]\!] : [\![\mathcal{U}_{i+1}]\!]\ \mathcal{U}_i\ \mathcal{U}_i$$

Need to modify the translation

$$[\mathcal{U}_i] := (\prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \sum_{e:A\simeq B} \prod_{\substack{a:A\\ b:B}} R\ a\ b \simeq a =\uparrow_e b,\ id_{\mathcal{U}_i},\ univ_{\mathcal{U}_i})$$

$$[\prod_{a:A} B\ a] := (\prod_{\substack{f:\prod_{a:A} B\ a\\ g:\prod_{a':A'} B'\ a}} \prod_{\substack{a:A\\ a':A'\\ a^\epsilon:[\![A]\!]\ a\ a'}} [\![B\ a]\!]\ f(a)\ g(a),\ equiv_\Pi,\ univ_\Pi)$$

$$[x] = x^\epsilon$$

$$[\lambda x.t] = \lambda(x : A)(x' : A)(x^\epsilon : [\![A]\!]\ x\ x').[t]$$

$$[uv] = [u]\ v\ v'\ [v]$$

$$[\![A]\!] = fst\ [A]$$

Reynolds Absraction Theorem

Theorem (The Abstaction Theorem)

If $\Gamma \vdash a : A$ *then* $[\![A]\!] \vdash [a] : [\![A]\!]\ a\ a'$

Proof.

Very Very Hard !

□

Reynolds Absraction Theorem

Theorem (The Abstaction Theorem)

If $\Gamma \vdash a : A$ *then* $[\![A]\!] \vdash [a] : [\![A]\!]\ a\ a'$

Proof.

Very Very Hard !

□

First, let see why the unvialent parametricity enables automatic transport of proofs !

The White Box and Black Box translations

Theorem (The White Box Fundamental Property)

If $|\Xi|_\circ \vdash a : A$ *then* $|\Xi|_\bullet \vdash (\uparrow_\Box a) : (\uparrow_\Box A)$ *and* $|\Xi| : [\![A]\!]_p^\Xi \; a \; (\uparrow \Box a)$

The White Box and Black Box translations

Theorem (The White Box Fundamental Property)

If $|\Xi|_\circ \vdash a : A$ *then* $|\Xi|_\bullet \vdash (\uparrow_\square a) : (\uparrow_\square A)$ *and* $|\Xi| : [\![A]\!]_p^\Xi \ a \ (\uparrow \square a)$

Theorem (The Black Box Fundamental Property)

For $A, B : \mathcal{U}_i$, *if* $A \approx B$ *then there is* $\uparrow_\blacksquare : A \to B$

The White Box and Black Box translations

Theorem (The White Box Fundamental Property)

If $|\Xi|_\circ \vdash a : A$ *then* $|\Xi|_\bullet \vdash (\uparrow_\Box a) : (\uparrow_\Box A)$ *and* $|\Xi| : [\![A]\!]^{\Xi}_{p}\ a\ (\uparrow \Box a)$

Theorem (The Black Box Fundamental Property)

For $A, B : \mathcal{U}_i$*, if* $A \approx B$ *then there is* $\uparrow_\blacksquare : A \to B$

We can use the White Box Fundamental Property to parametricaly translate operators and proof types such as

$$\prod_{n:\mathbb{N}} (0 = S\ n) \to \bot \approx \prod_{b:Bin} (0_{Bin} = S_{Bin}\ b) \to \bot$$

The White Box and Black Box translations

Theorem (The White Box Fundamental Property)

If $|\Xi|_\circ \vdash a : A$ *then* $|\Xi|_\bullet \vdash (\uparrow_\Box a) : (\uparrow_\Box A)$ *and* $|\Xi| : [\![A]\!]_p^\Xi\ a\ (\uparrow \Box a)$

Theorem (The Black Box Fundamental Property)

For $A, B : \mathcal{U}_i$, *if* $A \approx B$ *then there is* $\uparrow_\blacksquare : A \to B$

We can use the White Box Fundamental Property to parametricaly translate operators and proof types such as

$$\prod_{n:\mathbb{N}} (0 = S\ n) \to \bot \approx \prod_{b:Bin} (0_{Bin} = S_{Bin}\ b) \to \bot$$

Then automatically transport the proof by the Black Box Fundamental Property !

What goes wrong in the proof ?

Where does this proof goes wrong ? Veryfing the Universes typing rules !

What goes wrong in the proof ?

Where does this proof goes wrong ? Veryfing the Universes typing rules !

For the rule $\mathcal{U}_i : \mathcal{U}_{i+1}$ we need a term such that :

$$\prod_{A,B:\mathcal{U}_i} ([\![\mathcal{U}_i]\!] \; A \; B \simeq A = B)$$

What goes wrong in the proof ?

Where does this proof goes wrong ? Veryfing the Universes typing rules !

For the rule $\mathcal{U}_i : \mathcal{U}_{i+1}$ we need a term such that :

$$\prod_{A,B:\mathcal{U}_i} ([\![\mathcal{U}_i]\!]\ A\ B \simeq A = B)$$

Given $A, B : \mathcal{U}_i$ this unfolds to

$$(\sum_{R:A\to B\to\mathcal{U}_i}\ \sum_{e:A\simeq B}\ \prod_{\substack{a:A\\ b:B}} R\ a\ b \simeq a =\uparrow_e b) \simeq (A \simeq B)$$

A short proof

$$\sum_{R:A\to B\to\mathcal{U}_i}\ \sum_{e:A\simeq B}\ \prod_{\substack{a:A\\b:B}} R\ a\ b \simeq a =\!\!\uparrow_e b$$

$$\sum_{e:A\simeq B}\ \sum_{R:A\to B\to\mathcal{U}_i}\ \prod_{\substack{a:A\\b:B}} R\ a\ b \simeq a =\!\!\uparrow_e b$$

$$\sum_{e:A\simeq B}\ \sum_{R:A\to B\to\mathcal{U}_i}\ \prod_{\substack{a:A\\b:B}} R\ a\ b = (a =\!\!\uparrow_e b)$$

$$\simeq \sum_{e:A\simeq B}\ \sum_{R:A\to B\to\mathcal{U}_i} R = \lambda(a:A).\lambda(b:B).\ a =\!\!\uparrow_e b$$

$$\simeq A \simeq B$$

For CC_ω

Here the equivalence is the identity !

For CC_ω

Here the equivalence is the identity !

Not hard to define one for product. But proving the coherence condition gets really hard. More than 500 lignes of Coq.

For CC_ω

Here the equivalence is the identity !

Not hard to define one for product. But proving the coherence condition gets really hard. More than 500 lignes of Coq.

"This part is quite involved. In essence, this is where we prove that transporting in many hard- to-predict places is equivalent to transporting only at the top level. This is done by repeated use of commutativity lemmas of transport of equality over functions."

For CC_ω

Here the equivalence is the identity !

Not hard to define one for product. But proving the coherence condition gets really hard. More than 500 lignes of Coq.

"This part is quite involved. In essence, this is where we prove that transporting in many hard- to-predict places is equivalent to transporting only at the top level. This is done by repeated use of commutativity lemmas of transport of equality over functions."

Doesn't rise to CIC automatically.

Where does the definition goes wrong ?

$$[\![\mathcal{U}_i]\!] := \prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \sum_{e:A\simeq B} \prod_{\substack{a:A \\ b:B}} R\ a\ b \simeq a =\uparrow_e b$$

Where does the definition goes wrong ?

$$[\![\mathcal{U}_i]\!] := \prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \sum_{e:A\simeq B} \prod_{\substack{a:A \\ b:B}} R\ a\ b \simeq a =\uparrow_e b$$

The issues with the Univalent Parametricity are :

- The information is spread between R and e

Where does the definition goes wrong ?

$$[\![\mathcal{U}_i]\!] := \prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \sum_{e:A\simeq B} \prod_{\substack{a:A \\ b:B}} R\ a\ b \simeq a =\uparrow_e b$$

The issues with the Univalent Parametricity are :

- The information is spread between R and e
- The coherence condition is on terms !

Where does the definition goes wrong ?

$$[\![\mathcal{U}_i]\!] := \prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \sum_{e:A\simeq B} \prod_{\substack{a:A\\ b:B}} R\ a\ b \simeq a =\uparrow_e b$$

The issues with the Univalent Parametricity are :

- The information is spread between R and e
- The coherence condition is on terms !
- The definition is asymmetrical !

Univalent Relational Parametricity

Univalent Relational Parametricity

A new definition of the equivalence

A new definition of the equivalence :

$$A \bowtie B := \sum_{R:A \to B \to \mathcal{U}_i} \left(\prod_{a:A} \mathrm{isContr}\left(\sum_{b:B} R\ a\ b\right)\right) \times \left(\prod_{b:B} \mathrm{isContr}\left(\sum_{a:A} R\ a\ b\right)\right)$$

A new definition of the equivalence

A new definition of the equivalence :

$$A \bowtie B := \sum_{R:A\to B\to\mathcal{U}_i} \left(\prod_{a:A} \mathrm{isContr}\left(\sum_{b:B} R\ a\ b\right)\right) \times \left(\prod_{b:B} \mathrm{isContr}\left(\sum_{a:A} R\ a\ b\right)\right)$$

Then by denoting $\mathrm{isFun}\ R := \prod_{a:A} \mathrm{isContr}(\sum_{b:B} R\ a\ b)$ we get :

$$A \bowtie B := \sum_{R:A\to B\to\mathcal{U}_i} \mathrm{isFun}(R) \times \mathrm{isFun}(R^{op})$$

A new definition of the equivalence

A new definition of the equivalence :

$$A \bowtie B := \sum_{R:A\to B\to\mathcal{U}_i} \left(\prod_{a:A} \mathrm{isContr}\left(\sum_{b:B} R\ a\ b\right)\right) \times \left(\prod_{b:B} \mathrm{isContr}\left(\sum_{a:A} R\ a\ b\right)\right)$$

Then by denoting $\mathrm{isFun}\ R := \prod_{a:A} \mathrm{isContr}(\sum_{b:B} R\ a\ b)$ we get :

$$A \bowtie B := \sum_{R:A\to B\to\mathcal{U}_i} \mathrm{isFun}(R) \times \mathrm{isFun}(R^{op})$$

And the very important theorem :

$$(A \simeq B) \simeq (A \bowtie B)$$

The Univalent Relational Parametricity

With new definition it is possible to replace

$$[\![\mathcal{U}_i]\!] := \prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \sum_{e:A\simeq B} \prod_{\substack{a:A \\ b:B}} R\ a\ b \simeq a =\uparrow_e b$$

The Univalent Relational Parametricity

With new definition it is possible to replace

$$[\![\mathcal{U}_i]\!] := \prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \sum_{e:A\simeq B} \prod_{\substack{a:A \\ b:B}} R\ a\ b \simeq a =\uparrow_e b$$

by the following one :

$$[\![\mathcal{U}_i]\!] := \prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \mathrm{isFun}(R) \times \mathrm{isFun}(R^{op})$$

$$:= \prod_{A,B:\mathcal{U}_i} A \bowtie B$$

Thus if we can prove the abstraction theorem for this definition, then the white box and black box theorems are preserved !

Why is definition better ?

Why is this definition better for proving the abstract theorem ?

$$[\![\mathcal{U}_i]\!] := \prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \mathrm{isFun}(R) \times \mathrm{isFun}(R^{op})$$

$$\mathrm{isFun}(R) := \prod_{a:A} \mathrm{isContr}(\sum_{b:B} R\ a\ b)$$

- The definition is no longer spreading the information
- The cohernece condition on R is no longer on term, it is on spaces.
- The definition is symmetrical

Univalent Relational Parametricity

Proving the Abstraction Theorem for CC_ω

Need to modify the translation

$$[\mathcal{U}_i] := (\prod_{A,B:\mathcal{U}_i} \sum_{R:A\to B\to\mathcal{U}_i} \mathrm{isFun}(R) \times \mathrm{isFun}(R^{op}),\ FP_{\mathcal{U}_i}$$

$$[\prod_{a:A} B\ a] := (\prod_{\substack{f:\prod_{a:A} B\ a \\ g:\prod_{a':A'} B'\ a}} \prod_{\substack{a:A \\ a':A' \\ a^\epsilon:[\![A]\!]\ a\ a'}} [\![B\ a]\!]\ f(a)\ g(a),\ FP_\Pi)$$

$$[x] = x^\epsilon$$

$$[\lambda x.t] = \lambda(x : A)(x' : A)(x^\epsilon : [\![A]\!]\ x\ x').[t]$$

$$[uv] = [u]\ v\ v'\ [v]$$

$$[\![A]\!] = fst\ [A]$$

Proof for $\mathcal{U}_i : \mathcal{U}_{i+1}$

We nedd to prove $[\mathcal{U}_i] : [\![\mathcal{U}_{i+1}]\!]\ \mathcal{U}_i\ \mathcal{U}_i$.

The relation is fixed and is

$$FR_{\mathcal{U}_i} := \prod_{A,B:\mathcal{U}_i} A \bowtie B$$

Proof for $\mathcal{U}_i : \mathcal{U}_{i+1}$

We nedd to prove $[\mathcal{U}_i] : [\![\mathcal{U}_{i+1}]\!]\ \mathcal{U}_i\, \mathcal{U}_i$.

The relation is fixed and is

$$FR_{\mathcal{U}_i} := \prod_{A,B:\mathcal{U}_i} A \bowtie B$$

So it suffices to prove $\mathrm{isFun}(FR_{\mathcal{U}_i})$ and $\mathrm{isFun}(FR_{\mathcal{U}_i}^{op})$

$$\mathrm{isFun}(\prod_{A,B:\mathcal{U}_i} A \bowtie B) = \prod_{A:\mathcal{U}_i} \mathrm{isContr}(\sum_{B:\mathcal{U}_i} A \bowtie B)$$

Yet

$$(\sum_{B:\mathcal{U}_i} A \bowtie B) \simeq (\sum_{B:\mathcal{U}_i} A \simeq B) \simeq (\sum_{B:\mathcal{U}_i} A = B)$$

Proof for $\mathcal{U}_i : \mathcal{U}_{i+1}$

We nedd to prove $[\mathcal{U}_i] : [\![\mathcal{U}_{i+1}]\!]\ \mathcal{U}_i\, \mathcal{U}_i$.

The relation is fixed and is

$$FR_{\mathcal{U}_i} := \prod_{A,B:\mathcal{U}_i} A \bowtie B$$

So it suffices to prove $\mathrm{isFun}(FR_{\mathcal{U}_i})$ and $\mathrm{isFun}(FR_{\mathcal{U}_i}^{op})$

$$\mathrm{isFun}(\prod_{A,B:\mathcal{U}_i} A \bowtie B) = \prod_{A:\mathcal{U}_i} \mathrm{isContr}(\sum_{B:\mathcal{U}_i} A \bowtie B)$$

Yet

$$(\sum_{B:\mathcal{U}_i} A \bowtie B) \simeq (\sum_{B:\mathcal{U}_i} A \simeq B) \simeq (\sum_{B:\mathcal{U}_i} A = B)$$

This proves $\mathrm{isFun}(FR_{\mathcal{U}_i})$. By reversing A and B, we get $\mathrm{isFun}(FR_{\mathcal{U}_i}^{op})$ and so the result.

Proof for the function type 1/3

As before the relation is fixed

$$FR_{\Pi} := \prod_{f:\prod_{a:A} B\ a} \prod_{g:\prod_{a':A'} B'\ a'} \prod_{\substack{a:A \\ a':A' \\ a^{\epsilon}:[\![A]\!]\ a\ a'}} [\![B]\!]\ (f\ a)\ (g\ a')$$

First we nedd to prove $\mathrm{isContr}(FR_{\Pi})$ which can be done by the following :

Proof for the function type 2/3

$$\Delta = \sum_{g : \prod_{a' : A'} B'\ a'} \prod_{\substack{a : A \\ a' : A' \\ a^\epsilon : [\![A]\!]\ a\ a'}} [\![B]\!]\ (f\ a)\ (g\ a')$$

$$\simeq \sum_{g : \prod_{a' : A'} B'\ a'} \prod_{a' : A'} \prod_{\substack{a : A \\ a^\epsilon : [\![A]\!]\ a\ a'}} [\![B]\!]\ (f\ a)\ (g\ a')$$

$$\simeq \sum_{g : \prod_{a' : A'} B'\ a'} \prod_{a' : A'} \prod_{z : \sum_{a : A} [\![A]\!]\ a\ a'} [\![B]\!]\ (f\ z.1)\ (g\ a')$$

$$\simeq \sum_{g : \prod_{a' : A'} B'\ a'} \prod_{a' : A'} [\![B]\!]\ (f\ \uparrow a')\ (g\ a')$$

$$\simeq \sum_{g : \prod_{a' : A'} B'\ a'} \prod_{a' : A'} \uparrow (f\ \uparrow a') = (g\ a')$$

$$\simeq \sum_{g : \prod_{a' : A'} B'\ a'} (\lambda(a' : A').\ \uparrow (f\ \uparrow a') = g$$

Proof for the function type 3/3

The definition being symetric we have :

$$(FR_{\Pi}\ RA\ RB)^{op} \simeq FR_{\Pi}\ RA^{op}\ (\lambda a, a', a^{\epsilon} \rightarrow (\llbracket B \rrbracket\ a\ a'\ a^{\epsilon})^{op})$$

Proof for the function type 3/3

The definition being symetric we have :

$$(FR_{\Pi}\ RA\ RB)^{op} \simeq FR_{\Pi}\ RA^{op}\ (\lambda a, a', a^{\epsilon} \to (\llbracket B \rrbracket\ a\ a'\ a^{\epsilon})^{op})$$

Thanks to this result it suffices to prove $\mathrm{isContr}(FR_{\Pi})$ to prove $\mathrm{isContr}(FR_{\Pi}^{op})$.

Indeed, by definition RA^{op} and $(\lambda a, a', a^{\epsilon} \to (\llbracket B \rrbracket\ a\ a'\ a^{\epsilon})^{op})$ have the same properties as RA and RB.

Proof for the function type 3/3

The definition being symetric we have :

$$(FR_\Pi\ RA\ RB)^{op} \simeq FR_\Pi\ RA^{op}\ (\lambda a, a', a^\epsilon \to (\llbracket B \rrbracket\ a\ a'\ a^\epsilon)^{op})$$

Thanks to this result it suffices to prove $\mathrm{isContr}(FR_\Pi)$ to prove $\mathrm{isContr}(FR_\Pi^{op})$.

Indeed, by definition RA^{op} and $(\lambda a, a', a^\epsilon \to (\llbracket B \rrbracket\ a\ a'\ a^\epsilon)^{op})$ have the same properties as RA and RB.

Thus the result.

Univalent Relational Parametricity

The Abstraction Theorem for Parametric Inductive Types

How to add Inductive type : Typing rules

Adding inductive types is adding both type constructors and term constructors. For instance :

$$
\begin{aligned}
&\mathrm{list}\ A : \mathcal{U}_i \\
&\mid [] : \mathrm{list}\ A \\
&\mid\ :: : A \to \mathrm{list}\ A \to \mathrm{list}\ A
\end{aligned}
$$

How to add Inductive type : Typing rules

Adding inductive types is adding both type constructors and term constructors. For instance :

$$\begin{aligned} &\text{list } A : \mathcal{U}_i \\ &\mid [] : \text{list } A \\ &\mid \ :: : A \to \text{list } A \to \text{list } A \end{aligned}$$

Given $A, A' : \mathcal{U}_n$ such that $A \bowtie A'$ then

$$\begin{aligned} & [\text{list } A] : \ [\![A]\!] \ (\text{list } A)\ (\text{list } A') \\ \text{ie} \quad & [\text{list } A] : (\text{list } A) \bowtie (\text{list } A') \\ \text{ie} \quad & [\text{list } A] : \sum_{R : \text{list } A \to \text{list } A' \to \mathcal{U}_i} \text{isFun}(R) \times \text{isFun}(R^{op}) \end{aligned}$$

How to add Inductive type : Constructors

Then we nedd to do so for the constructors :
In the degenerate case, one need to prove

$$[\ []\ ] : [\![\mathrm{list}\ A]\!]\ \ []\ []$$

In the recursive case a, l, a', l' such that $a \approx a'$ and $l \approx l'$ prove

$$[a :: l] : [\![\mathrm{list}\ A]\!]\ \ (a :: l)\ (a' :: l')$$

Find a relation

First we need to define a relation over the list given $RA : A \approx A'$:

$$
\begin{aligned}
& FR_{\mathrm{list}}\ RA\ l\ l' := \mathrm{match}\ l, l'\ \mathrm{with} \\
& \mid [],\ [] \qquad\qquad \Longrightarrow \top \\
& \mid a :: l,\ a' :: l' \Longrightarrow \sum_{Xa : a \approx a'} FR_{\mathrm{list}}\ RA\ l\ l' \\
& \mid \_,\ \_ \qquad\qquad \Longrightarrow \bot
\end{aligned}
$$

Proving $\mathrm{isFun}(FR_{\mathrm{list}})$ 1/2

By the previous remarks it suffices to show $\mathrm{isFun}(FR_{\mathrm{list}})$ to get $\mathrm{isFun}(FR^{op}_{\mathrm{list}})$.

First, one need to do an induction on l then prove :

$$\mathrm{isContr}(\sum_{l' : \mathrm{list}\ A'} FR_{\mathrm{list}}\ []\ l') \qquad \mathrm{isContr}(\sum_{l' : \mathrm{list}\ A'} FR_{\mathrm{list}}\ (a :: l)\ l')$$

Proving $\mathrm{isFun}(FR_{\mathrm{list}})$ 1/2

By the previous remarks it suffices to show $\mathrm{isFun}(FR_{\mathrm{list}})$ to get $\mathrm{isFun}(FR^{op}_{\mathrm{list}})$.

First, one need to do an induction on l then prove :

$$\mathrm{isContr}(\sum_{l':\mathrm{list}\ A'} FR_{\mathrm{list}}\ []\ l') \qquad \mathrm{isContr}(\sum_{l':\mathrm{list}\ A'} FR_{\mathrm{list}}\ (a :: l)\ l')$$

The key is to reason on the entire sum rather than on the inside

$$\sum_{l':\mathrm{list}\ A'} \mathrm{FR\_list}\ RA\ []\ l' \quad \simeq \quad FR_{\mathrm{list}}\ RA\ []\ [] \quad \equiv \quad \top$$

Proving $\mathrm{isFun}(\mathbf{FR\_list})$ 2/2

Then for the induction, one gets :

$$
\begin{aligned}
\Delta &:= \sum_{l':\mathrm{list}\ A'} \mathrm{FR\_list}\ RA\ (a :: l)\ l' \\
&\simeq \sum_{a':A'} \sum_{l':\mathrm{list}\ A'} \mathrm{FR\_list}\ RA\ (a :: l)\ (a' :: l') \\
&\equiv \sum_{a':A'} \sum_{l':\mathrm{list}\ A'} \sum_{Xa:a \approx a'} \mathrm{FR\_list}\ RA\ l\ l' \\
&\simeq \sum_{a':A'} \sum_{Xa:a \approx a'} \sum_{l':\mathrm{list}\ A'} \mathrm{FR\_list}\ RA\ l\ l' \\
&\simeq \sum_{a':A'} a \approx a'
\end{aligned}
$$

Proving the result on the constructors

We need to give :

$$[\ []\ ] : [\![\mathrm{list}\ A]\!]\ \ []\ []$$
$$[\ []\ ] : \top$$

And in the recursive case, given a, l, a', l' such that $a \approx a'$ and $l \approx l'$ give

$$[a :: l] : [\![\mathrm{list}\ A]\!]\ \ (a :: l)\ (a' :: l')$$
$$[a :: l] : \sum_{Xa : a \approx a'} l \approx l'$$

So verifiying the abstract theorem trivial fr constructors

Univalent Relational Parametricity

The Abstraction Theorem For Indexed Inductive Types

The special case of equality 1/2

To prove the results for all indexed inductive types, we are going to treat first the equality

$$\begin{aligned} & \mathit{eq}\ A\ x : A \to \mathcal{U}_n := \\ & \mid \mathit{refl} : \mathit{eq}\ x\ x \end{aligned}$$

The special case of equality 1/2

To prove the results for all indexed inductive types, we are going to treat first the equality

$$\begin{aligned} &eq\ A\ x : A \to \mathcal{U}_n := \\ &\mid refl : eq\ x\ x \end{aligned}$$

Then given $A, A' : \mathcal{U}_n$, $x, y : A$ such that $x', y' : A'$ such that $RA : A \bowtie A'$, $Xx : x \approx x'$, $Xy : y \approx y'$:

$$FR_{eq}\ (p : x = x')\ (q : y = y') :=$$
$$\mathrm{transport}^{\lambda x' \to x \approx x'}\ q\ Xx = \mathrm{transport}^{\lambda x \to x \approx y'}\ p^{-1}\ Xy$$

The special case of equality 2/2

Then we nedd to prove :

$$\mathrm{isContr}(\sum_{q:y=y'} \mathrm{transport}^{\lambda x' \to x \approx x'}\ q\ Xx = \mathrm{transport}^{\lambda x \to x \approx y'}\ p^{-1}\ Xy)$$

By path induction on p then we have the following :

$$\sum_{q:y=y'} \mathrm{transport}^{\lambda x' \to x \approx x'}\ q\ Xx = Xy \quad \simeq \quad (y, Xx) = (y', Xy)$$

Then it suffices to prove $\mathrm{isContr}(\sum_{y:A} RA\ x\ y)$, which is the case by definition of RA.

We also need to give $[refl] : [\![eq]\!]\ refl\ refl$ ie a proof of $Xx = Yy$ which is possible because both are proof $RA\ x\ x$

All the others inductive types 1/2

Let's use the vectors as an example :

$$\begin{aligned} &\mathit{vec}\ A : \mathbb{N} \to \mathcal{U}_n \\ \mid\ &\mathit{nil} : \mathit{vec}\ A\ 0 \\ \mid\ &\mathit{cons} : \prod_{k:\mathbb{N}} A \to \mathit{vec}\ A\ k \to \mathit{vec}\ A\ (k+1) \end{aligned}$$

All the others inductive types 1/2

Let's use the vectors as an example :

$$\begin{aligned} & vec\ A : \mathbb{N} \to \mathcal{U}_n \\ & \mid nil : vec\ A\ 0 \\ & \mid cons : \prod_{k:\mathbb{N}} A \to vec\ A\ k \to vec\ A\ (k+1) \end{aligned}$$

They can be turn in an non-indexed inductive type :

$$\begin{aligned} & vecF\ A\ k : \mathcal{U}_n \\ & \mid nilF : 0 = k \to vecF\ A\ k \\ & \mid consF : \prod_{l:\mathbb{N}} S\ l = n \to A \to vecF\ A\ l \to vecF\ A\ n \end{aligned}$$

All the others inductive types 2/2

Then *vecF* it a regular parametrise inductive types and verifies parametricity.

All the others inductive types 2/2

Then *vecF* it a regular parametrise inductive types and verifies parametricity. Then we can define :

$$FR^F_{vec}\ v\ v' := FR_{vecF}\ (\uparrow_F v)\ (\uparrow_F v')$$

From this it is easily to prove that FR_{vec} is paraemtric
Pus one can prove that $FR_{vec}\ v\ v' \simeq FR^F_{vec}\ v\ v'$ so the constructors are trivial to do.

All the others inductive types 2/2

Then *vecF* it a regular parametrise inductive types and verifies parametricity. Then we can define :

$$FR^{F}_{vec}\ v\ v' := FR_{vecF}\ (\uparrow_F v)\ (\uparrow_F v')$$

From this it is easily to prove that FR_{vec} is paraemtric
Pus one can prove that $FR_{vec}\ v\ v' \simeq FR^{F}_{vec}\ v\ v'$ so the constructors are trivial to do.

Hence we have proved that Parametricity goes trough the vectors and so for general indexed inductive types.

Conlusion

Conclusion 1/2

- An idea of what is parametricity and its history

Conclusion 1/2

- An idea of what is parametricity and its history
- Parametricity (made heterogenous) is not enough for automatic transport, it fails on the computation problem

Conclusion 1/2

- An idea of what is parametricity and its history
- Parametricity (made heterogenous) is not enough for automatic transport, it fails on the computation problem
- Univalece is not enough either, failing on the anticipation problem

Conclusion 1/2

- An idea of what is parametricity and its history
- Parametricity (made heterogenous) is not enough for automatic transport, it fails on the computation problem
- Univalece is not enough either, failing on the anticipation problem
- A cubical setting enables to transport proof but not automatically

Conclusion 1/2

- An idea of what is parametricity and its history
- Parametricity (made heterogenous) is not enough for automatic transport, it fails on the computation problem
- Univalece is not enough either, failing on the anticipation problem
- A cubical setting enables to transport proof but not automatically
- By mixing Parametricity and Univalence one can automatically transport proof !

Conclusion 1/2

- An idea of what is parametricity and its history
- Parametricity (made heterogenous) is not enough for automatic transport, it fails on the computation problem
- Univalece is not enough either, failing on the anticipation problem
- A cubical setting enables to transport proof but not automatically
- By mixing Parametricity and Univalence one can automatically transport proof !
- However, for it to work, they must be made carefully

Conclusion 1/2

- An idea of what is parametricity and its history
- Parametricity (made heterogenous) is not enough for automatic transport, it fails on the computation problem
- Univalece is not enough either, failing on the anticipation problem
- A cubical setting enables to transport proof but not automatically
- By mixing Parametricity and Univalence one can automatically transport proof !
- However, for it to work, they must be made carefully
- Then it is possible to automatically transport proof for CC_ω + parametrise inductive types + indexed inductive types. So basically for Coq !

Conclusion 1/2

- An idea of what is parametricity and its history
- Parametricity (made heterogenous) is not enough for automatic transport, it fails on the computation problem
- Univalece is not enough either, failing on the anticipation problem
- A cubical setting enables to transport proof but not automatically
- By mixing Parametricity and Univalence one can automatically transport proof !
- However, for it to work, they must be made carefully
- Then it is possible to automatically transport proof for CC_ω + parametrise inductive types + indexed inductive types. So basically for Coq !

Conclusion 2/2

Some limitations with this approach :

- For the data refinement problem, one need the data to be equivalent
- This approach doesn't seem to be automatisable for HIT... or even work
- This means no quotient which annoying for the data refinement problem

References

JEAN-PHILIPPE BERNARDY, PATRIK JANSSON, and ROSS PATERSON. "Proofs for free". In: *Journal of Functional Programming* 22.2 (2012), pp. 107–152. DOI: 10.1017/s0956796812000056.

Cyril Cohen, Maxime Dénès, and Anders Mörtberg. "Refinements for Free!" In: *Certified Programs and Proofs - Third International Conference, CPP 2013, Melbourne, VIC, Australia, December 11-13, 2013, Proceedings*. Ed. by Georges Gonthier and Michael Norrish. Vol. 8307. Lecture Notes in Computer Science. Springer, 2013, pp. 147–162. DOI: 10.1007/978-3-319-03545-1_10. URL: https://doi.org/10.1007/978-3-319-03545-1%5C_10.

John C. Reynolds. "Types, Abstraction and Parametric Polymorphism". In: *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19-23, 1983*. Ed. by R. E. A. Mason. North-Holland/IFIP, 1983, pp. 513–523.

Nicolas Tabareau, Éric Tanter, and Matthieu Sozeau. "The marriage of univalence and Parametricity". In: *Journal of the ACM* 68.1 (2021), pp. 1–44. DOI: 10.1145/3429979.

The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013.

Philip Wadler. "Theorems for free!" In: *FPCA '89* (1989). DOI: 10.1145/99370.99404.