Toward a Cubical Type Theory Univalent by Definition

Hugo Moeneclaey,
ENS Paris-Saclay

joint work with:

Hugo Herbelin,
INRIA

HoTT 2019

Summary

Summary

Introduction: Cubical Type Theory and Parametricity

Sketching our theory

Computing with univalence

Features of Cubical Type Theory [Cohen, Coquand, Huber, Mörtberg 2016]

Apart from an abstract interval, it has:

- Connections allowing to degenerate a path to a square.
- Reversal allowing to go through a path backward.
- Kan compositions generalizing the concatenation of paths.
- Glue types, necessary to prove univalence.

Theorem [Huber 2018]

Cubical Type Theory enjoys canonicity.

In this talk

We present an ongoing attempt to build a variant of Cubical Type Theory where we have *univalence by definition*:

$$(A =_{\mathcal{U}} B) \;\equiv\; (A \simeq B)$$

We mainly use ideas from parametricity.

Parametricity

Intuition

Terms built in type theory depend nicely on their type inputs.

Formally: terms send related inputs to related outputs [Reynolds 83].

Applications: Theorems for free! [Wadler 89]

Deduce a result on a polymorphic term from its type.

An example of parametricity

Assume given $X_0, X_1 : \mathcal{U}$ and $X_* : X_0 \to X_1 \to \mathcal{U}$.

Definition

For any simple type A built from X we extend X_* to:

$$A_* : A[X/X_0] \to A[X/X_1] \to \mathcal{U}$$

by:

$$\begin{aligned} (A \times B)_*((a, a'), (b, b')) &\equiv A_*(a, a') \times B_*(b, b') \\ (A \to B)_*(f, g) &\equiv (x_0 : A_0) \to (x_1 : A_1) \\ &\quad \to A_*(x_0, x_1) \to B_*(f(x_0), g(x_1)) \end{aligned}$$

Parametricity result

For any simple type A built from X and a such that:

$$\vdash a : A$$

there exists a_* such that:

$$\vdash a_* : A_*(a[X/X_0], a[X/X_1])$$

Can be extended to PTS and inductive types [Bernardy, Jansson, Paterson 2010], the crucial point being:

$$\mathcal{U}_*(A, B) \;\equiv\; A \to B \to \mathcal{U}$$

Internal parametricity

Parametricity is external, but it can be internalized.

Parametric Type Theory [Bernardy, Coquand, Moulin 2015]

Strikingly similar to Cubical Type Theory.
We denote by $x \sim_A y$ the analogue to path types. We have the relativity axiom, in this case:

$$(A \sim_{\mathcal{U}} B) \ \cong \ (A \to B \to \mathcal{U})$$

where $\_ \cong \_$ stands for definitional isomorphism.

They use predicates rather than relations.

Parametricity and higher dimensional type theory

Ideas flow both ways:

Examples

- [Cavalo, Harper 2018] presents a type theory both Parametric and Higher-dimensional. Relativity is formulated as:

$$(A \sim_{\mathcal{U}} B) \simeq (A \to B \to \mathcal{U})$$

- [Altenkirch, Kaposi 2017] presents ideas toward a higher dimensional type theory without interval, inspired by parametricity.
- [Tabareau, Tanter, Sozeau 2017] implements ideas from parametricity in order to mechanize the transfer of some libraries along equivalences in Coq.

Examples with extensionality

- In Observational Type Theory [Altenkirch, McBride, Swierstra 2007] identity types are defined by induction on a a closed universe.
- XTT [Angiuli, Gratzer, Sterling 2019] uses cubical techniques, but two paths with the same endpoints are definitionally equal.

Summary

Introduction: Cubical Type Theory and Parametricity

Sketching our theory

A core type theory

We start with all the rules for a type theory with:

- Σ and Π with η-rules.
- A hierarchy of universes, all denoted $\mathcal{U}$.

Heterogeneous path types

We denote $\_ =_{\lambda i.A} \_$ by $\_ =_A \_$ when i does not occur in A.

Definition

We add heterogeneous path types:

$$\frac{\Gamma \vdash \epsilon : X =_{\mathcal{U}} Y}{\Gamma \vdash \_ =_{\epsilon} \_ : X \to Y \to \mathcal{U}}$$

$$\frac{\Gamma, i \vdash t : A}{\Gamma \vdash \lambda i.t : t[i/0] =_{\lambda i.A} t[i/1]}$$

$$\frac{\Gamma, i, \Gamma' \vdash p : s =_{\epsilon} t}{\Gamma, i, \Gamma' \vdash p(i) : \epsilon(i)}$$

For $p : a_0 =_{\epsilon} a_1$, we define $(p(i))[i/u]$ as $a_u[i/u]$ where $u \in \{0, 1\}$.

Equivalences

Definition

An equivalence $\epsilon : A \simeq B$ consists of a relation $R : A \to B \to \mathcal{U}$ with contractible fibers. In particular we have:

- Functions $\overrightarrow{\epsilon} : A \to B$ and $\overrightarrow{\overrightarrow{\epsilon}} : (x : A) \to R(x, \overrightarrow{\epsilon}(x))$.
- Functions $\overleftarrow{\epsilon} : B \to A$ and $\overleftarrow{\overleftarrow{\epsilon}} : (y : B) \to R(\overleftarrow{\epsilon}(y), y)$.

We add:

$$(X =_{\mathcal{U}} Y) \equiv (X \simeq Y)$$

We identify $\_ =_{\epsilon} \_$ with the underlying relation of $\epsilon : A =_{\mathcal{U}} B$.

Computing with path types: some examples

For product types we add:

$$
\begin{array}{rcl}
(a, b) =_{\lambda i.A\times B} (a', b') & \equiv & (a =_{\lambda i.A} a') \times (b =_{\lambda i.B} b') \\
\overrightarrow{\lambda i.A \times B}(a, b) & \equiv & \left(\overrightarrow{\lambda i.A}(a), \overrightarrow{\lambda i.B}(b)\right) \\
\overrightarrow{\overrightarrow{\lambda i.A \times B}}(a, b) & \equiv & \left(\overrightarrow{\overrightarrow{\lambda i.A}}(a), \overrightarrow{\overrightarrow{\lambda i.B}}(b)\right) \\
(\lambda i.c).1 & \equiv & \lambda i.(c.1) \\
(p, q)(i) & \equiv & (p(i), q(i))
\end{array}
$$

For function types we add:

$$
\begin{array}{rcl}
f =_{\lambda i.A \to B} g & \equiv & (x_0 : A[i/0]) \to (x_1 : A[i/1]) \\
 & & \to x_0 =_{\lambda i.A} x_1 \to f(x_0) =_{\lambda i.B} g(x_1) \\
\overrightarrow{\lambda i.A \to B}(f) & \equiv & \overrightarrow{\lambda i.B} \circ f \circ \overleftarrow{\lambda i.A} \\
(\lambda i.f)(a_0, a_1, a_*) & \equiv & \lambda i.f(a_*(i)) \\
(\lambda a_0, a_1, a_*.\, t)(i) & \equiv & ?
\end{array}
$$

Computing with path types: regularity

When i does not occur in A, we add:

$$\begin{array}{rcl} \overrightarrow{\lambda i.A} & \equiv & \lambda(x : A).\, x \\ \overrightarrow{\overrightarrow{\lambda i.A}} & \equiv & \lambda(x : A).\, \mathrm{refl}_x \end{array}$$

Warning

This is not known to be consistent with univalence.

Toward full computation

How to add type formers

For any type former T, we need to give computation rules for:

- Components of the equivalence $\lambda i. T(A, B)$, for example:

$$t_1 =_{\lambda i. T(A,B)} t_2 \quad \equiv \quad C(t_1, t_2, \lambda i.A, \lambda i.B)$$

- $\mathbf{elim}_=(\lambda i.t)$ with $\mathbf{elim}_=$ eliminator of C.
- $\mathbf{cons}_=(t)(i)$ with $\mathbf{cons}_=$ constructor of C.

We have all rules for Σ and Π, except for:

$$(\lambda a_0, a_1, a_*. \, t)(i)$$

These rules respect regularity.

A guess for normal forms

We write $\mathrm{Equiv}(\epsilon)$ for the second projection of $\epsilon : A =_{\mathcal{U}} B$.
We write $\langle \_, \cdots, \_ \rangle$ for the constructor of equivalences.

Definition

We define the set neutral terms N and values V by induction:

$$
\begin{aligned}
N \quad &:= \quad x \mid N(i) \mid N.1 \mid N.2 \mid N(V) \mid \\
&\qquad \_ =_{\lambda i.N} \_ \mid \mathrm{Equiv}(\lambda i.N) \mid \langle V, \cdots, V \rangle(i) \\
V \quad &:= \quad N \mid \lambda i.V \mid (V, V) \mid \lambda x.V \mid \\
&\qquad \Sigma(x : V).V \mid \Pi(x : V).V \mid \mathcal{U}
\end{aligned}
$$

Toward interpretation

How to justify this theory?

Iterated parametricity

We hope for a translation similar to parametricity, but with:

$$\mathcal{U}_*(A, B) \quad \equiv \quad A \simeq B$$

Then this translation should be iterated once per dimension name.

Further work

- We need to solve the problem with Π-types.
- We need to give an interpretation. Is regularity consistent?
- What about confluence, normalization, canonicity?
- What about inductive types? And higher inductive types?
- Can we internalize parametricity similarly?
- Can we internalize other principles this way?