Yet Another Cartesian Cubical Type Theory

Anders Mörtberg

Carnegie Mellon University and University of Gothenburg

Types, Homotopy Type theory, and Verification
HIM Bonn
June 6, 2018

Cubes, cubes, cubes...

I will talk about my attempts to understand the new and exciting developments on **Cartesian** cubical type theories:

Computational Higher Type Theory III:
Univalent Universes and Exact Equality
(Angiuli, Favonia, Harper - AFH)

Cartesian Cubical Type Theory
(Angiuli, Brunerie, Coquand, Favonia, Harper, Licata - ABCFHL)

These provide us with new constructive models of Univalent Foundations and higher inductive types

yacctt

Slogan: the best way to understand type theory is to implement it!

Together with Carlo Angiuli I have adapted the code-base of cubicaltt to implement a Cartesian cubical type theory:

`yacctt`: *yet another cartesian cubical type theory*[1]
`https://github.com/mortberg/yacctt/`

Inspired by *Cubical Type Theory: a constructive interpretation of the univalence axiom* (Cohen, Coquand, Huber, M. - CCHM)

[1] `https://en.wikipedia.org/wiki/Yacc`

yacctt

In this talk I will present this syntactically, however everything I say can be done internally in a topos extended with suitable axioms following:

Axioms for Modelling Cubical Type Theory in a Topos
(Orton, Pitts)

Internal Universes in Models of Homotopy Type Theory
(Licata, Orton, Pitts, Spitters)

This is the approach taken in ABCFHL

yacctt

My main motivation for implementing another cubical type theory is to explore the following:

1. How convenient is it to formalize mathematics in this new system with its new primitives?
2. Does this compute more efficiently than cubicaltt? (Can we compute the Brunerie number?)

yacctt

`yacctt` extends dependent type theory (with eta for Π and Σ) with:

- Path types based on a Cartesian interval
- Diagonal context restrictions (generating cofibrations)
- Generalized Kan operations (transport of structures[2])
- V-types (special case of Glue-types that suffices for univalence)
- Fibrant universes
- Some higher inductive types

[2] cf. Bourbaki: Theory of sets, 1968

Cartesian interval

Formal representation of the interval, $\mathbb{I}$:

$$r, s \quad ::= \quad 0 \mid 1 \mid i$$

$i, j, k\ldots$ formal symbols/names representing directions/dimensions

Contexts can contain variables in the interval:

$$\frac{\Gamma \vdash}{\Gamma, i : \mathbb{I} \vdash}$$

Cartesian interval

$i : \mathbb{I} \vdash A$ corresponds to a line:

$$A(0/i) \xrightarrow{\quad A \quad} A(1/i)$$

$i : \mathbb{I}, j : \mathbb{I} \vdash A$ corresponds to a square:

$$\begin{tikzcd}
A(0/i)(1/j) \arrow[r, "A(1/j)"] & A(1/i)(1/j) \\
A(0/i)(0/j) \arrow[u, "A(0/i)"] \arrow[r, "A(0/j)"'] \arrow[ur, "A(j/i)"] & A(1/i)(0/j) \arrow[u, "A(1/i)"']
\end{tikzcd}
\qquad
\begin{tikzcd}
j \arrow[u] \\
\cdot \arrow[r] \arrow[u, "j"] & i
\end{tikzcd}$$

Diagonal substitutions are allowed (no linearity constraint as in BCH)

Path types

The Path types are modelled as:

$$Path(A) := A^{\mathbb{I}}$$

$$Path_A(a, b) := \{p \in Path(A) \mid p\,0 = a \wedge p\,1 = b\}$$

In the syntax we write Path $A\,a\,b$ for the Path types and $\langle i \rangle\; u$ for Path abstraction

These types are defined by the same rules as in CCHM and provide a convenient syntax for directly reasoning about (higher) equality types

We can directly prove that these satisfy function extensionality (CCHM)

Composition operations

We want to be able to compose paths:

$$a \xrightarrow{p} b \qquad\qquad b \xrightarrow{q} c$$

We do this by computing the dashed line in:

$$\begin{tikzcd}
a \arrow[r, dashed] & c \\
a \arrow[u, "a"] \arrow[r, "p"'] & b \arrow[u, "q"']
\end{tikzcd}$$

In general this corresponds to computing the missing sides of n-dimensional cubes

Composition operations

Box principle: any open box has a lid

Cubical version of the Kan condition for simplicial sets:

"Any horn can be filled"

First formulated by Daniel Kan in *"Abstract Homotopy I"* (1955) for cubical complexes

Context restrictions

To formulate this we need syntax for representing partially specified n-dimensional cubes

We add context **restrictions** Γ, φ where φ is a "face formula" representing a subset of the faces of a cube

$$\varphi, \psi \quad ::= \quad 0_{\mathbb{F}} \mid 1_{\mathbb{F}} \mid (i = 0) \mid (i = 1) \mid (i = j) \mid \varphi \wedge \psi \mid \varphi \vee \psi$$

Key new idea is to allow $(i = j)$ as context restrictions! (AFH)

Partial elements

Any judgment valid in a context Γ is also valid in a restriction Γ, φ

$$\frac{\Gamma \vdash A}{\Gamma, \varphi \vdash A}$$

If $\Gamma \vdash A$ and $\Gamma, \varphi \vdash a : A$ then a is a **partial element** of A of extent φ

We write $\Gamma \vdash b : A[\varphi \mapsto a]$ for

$$\Gamma \vdash b : A \qquad \Gamma, \varphi \vdash a : A \qquad \Gamma, \varphi \vdash a = b : A$$

Box principle in CCHM

In CCHM we formulated the box principle as:

$$\frac{\Gamma, i : \mathbb{I} \vdash A \qquad \Gamma, \varphi, i : \mathbb{I} \vdash u : A \qquad \Gamma \vdash u_0 : A(0/i)[\varphi \mapsto u(0/i)]}{\Gamma \vdash \mathsf{comp}^i\ A\ [\varphi \mapsto u]\ u_0 : A(1/i)[\varphi \mapsto u(1/i)]}$$

- u_0 is the bottom
- u is the sides
- $\mathsf{comp}^i\ A\ [\varphi \mapsto u]\ u_0$ is the lid

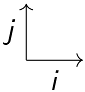
Semantically this is a **structure** (and not a property) of a type. A type is called **fibrant** if it can be equipped with this structure

Composition operations: example

With composition we can justify transitivity of path types:

$$\frac{\Gamma \vdash p : \mathsf{Path}\ A\ a\ b \qquad \Gamma \vdash q : \mathsf{Path}\ A\ b\ c}{\Gamma \vdash \langle i \rangle\ \mathsf{comp}^j\ A\ [(i=0) \mapsto a, (i=1) \mapsto q\ j]\ (p\ i) : \mathsf{Path}\ A\ a\ c}$$

$$\begin{tikzcd}
a \arrow[r, dashed] & c \\
a \arrow[u, "a"] \arrow[r, "p\ i"'] & b \arrow[u, "q\ j"']
\end{tikzcd}$$



Transport as composition in CCHM

Composition for $\varphi = 0_{\mathbb{F}}$ corresponds to transport:

$$\frac{\Gamma, i : \mathbb{I} \vdash A \qquad \Gamma \vdash u_0 : A(i/0)}{\Gamma \vdash \mathsf{transport}^i\ A\ u_0 = \mathsf{comp}^i\ A\ []\ u_0 : A(i/1)}$$

$$u_0 \bullet \qquad\qquad\qquad \bullet\ \mathsf{transport}^i\ A\ u_0$$

$$A(0/i) \xrightarrow[i]{A} A(1/i)$$

Kan filling from composition

A key observation in CCHM is that we can compute the filler of a cube using composition and connections

In yacctt we don't have any connections... What can we do?

Kan filling from composition

A key observation in CCHM is that we can compute the filler of a cube using composition and connections

In yacctt we don't have any connections... What can we do?

Solution: strengthen the composition operation!

Strengthened composition

Compose from r to s:

$$\frac{\Gamma \vdash \varphi : \mathbb{F} \qquad \begin{array}{c}\Gamma, i : \mathbb{I} \vdash A \qquad \Gamma \vdash r : \mathbb{I} \qquad \Gamma \vdash s : \mathbb{I} \\ \Gamma, \varphi, i : \mathbb{I} \vdash u : A \qquad \Gamma \vdash u_0 : A(r/i)[\varphi \mapsto u(r/i)]\end{array}}{\Gamma \vdash \mathsf{com}_i^{r \to s}\, A\, [\varphi \mapsto u]\, u_0 : A(s/i)[\varphi \mapsto u(s/i), r = s \mapsto u_0]}$$

We recover comp when $r = 0$ and $s = 1$

We get the filler when $r = 0$ and s is a dimension variable $j : \mathbb{I}$

Strengthened composition

We can now define com by cases on the type A just like in CCHM, however in order to also be able to support HITs we first decompose the operation into **homogeneous composition** and **coercion**:

$$\frac{\begin{array}{c}\Gamma \vdash A \qquad \Gamma \vdash r : \mathbb{I} \qquad \Gamma \vdash s : \mathbb{I} \\ \Gamma \vdash \varphi : \mathbb{F} \qquad \Gamma, \varphi, i : \mathbb{I} \vdash u : A \qquad \Gamma \vdash u_0 : A[\varphi \mapsto u(r/i)]\end{array}}{\Gamma \vdash \mathsf{hcom}_i^{r \to s}\, A\, [\varphi \mapsto u]\, u_0 : A[\varphi \mapsto u(s/i), r = s \mapsto u_0]}$$

$$\frac{\Gamma, i : \mathbb{I} \vdash A \qquad \Gamma \vdash r : \mathbb{I} \qquad \Gamma \vdash s : \mathbb{I} \qquad \Gamma \vdash u : A(r/i)}{\Gamma \vdash \mathsf{coe}_i^{r \to s}\, A\, u : A(s/i)[r = s \mapsto u]}$$

Coercion examples

Given $i : \mathbb{I} \vdash u : A$ we get:

$$\begin{tikzcd}
 & u(1/i) \\
u(0/i) \arrow[ur, "u"] & \\
A(0/i) \arrow[r, "A"'] & A(1/i)
\end{tikzcd}$$

Coercion examples

Given $i : \mathbb{I} \vdash u : A$ we get:

$$
\begin{tikzcd}
 & u(1/i) \\
u(0/i) \arrow[ur, "u"] & \mathsf{coe}_i^{0 \to 1}\, A\, u(0/i) \\
A(0/i) \arrow[r, "A"'] & A(1/i)
\end{tikzcd}
$$

Coercion examples

Given $i : \mathbb{I} \vdash u : A$ we get:

$$
\begin{tikzcd}
 & & u(1/i) \\
u(0/i) \arrow[urr, "u"] \arrow[rr, "\mathsf{coe}_i^{0 \to i}\, A\, u(0/i)"'] & & \mathsf{coe}_i^{0 \to 1}\, A\, u(0/i)
\end{tikzcd}
$$

$$
A(0/i) \xrightarrow[A]{} A(1/i)
$$

Coercion examples

Given $i : \mathbb{I} \vdash u : A$ we get:

$$
\begin{tikzcd}
 & & u(1/i) \\
u(0/i) \arrow[urr, "u"] \arrow[rr, "\mathsf{coe}_i^{0 \to i}\, A\, u(0/i)"'] & & \mathsf{coe}_i^{0 \to 1}\, A\, u(0/i) \arrow[u, "\mathsf{coe}_i^{i \to 1}\, A\, u"']
\end{tikzcd}
$$

$$
\begin{tikzcd}
A(0/i) \arrow[rr, "A"'] & & A(1/i)
\end{tikzcd}
$$

Reversals

Given $p : \mathrm{Path}\, A\, a\, b$ we can define $p^{-1} : \mathrm{Path}\, A\, b\, a$ as

$$p^{-1} := \langle i \rangle \mathrm{hcom}_j^{0 \to 1}\, A\, [(i = 0) \mapsto p\, j, (i = 1) \mapsto a]\, a$$

This corresponds to the dashed line in:

$$\begin{tikzcd}
b \arrow[r, dashed, "p^{-1}"] & a \\
a \arrow[u, "p"] \arrow[r, "a"'] & a \arrow[u, "a"']
\end{tikzcd}$$

Connections

Given p : Path $A\,a\,b$ we can define:

$$
\begin{aligned}
\langle i\ j\rangle \mathsf{hcom}_k^{0\to 1}\,A\,[&(i=0) \mapsto \mathsf{hcom}_l^{1\to 0}\,A\,[(k=0)\mapsto a,(k=1)\mapsto p\,l]\,(p\,k) \\
&,(i=1) \mapsto \mathsf{hcom}_l^{1\to j}\,A\,[(k=0)\mapsto a,(k=1)\mapsto p\,l]\,(p\,k) \\
&,(j=0) \mapsto \mathsf{hcom}_l^{1\to 0}\,A\,[(k=0)\mapsto a,(k=1)\mapsto p\,l]\,(p\,k) \\
&,(j=1) \mapsto \mathsf{hcom}_l^{1\to i}\,A\,[(k=0)\mapsto a,(k=1)\mapsto p\,l]\,(p\,k)]\;a
\end{aligned}
$$

This has boundary:

$$
\begin{tikzcd}
a \arrow[r, "p"] & b \\
a \arrow[u, "a"] \arrow[r, "a"'] & a \arrow[u, "p"']
\end{tikzcd}
$$

Connections

Given $p : \mathsf{Path}\, A\, a\, b$ we can define:

$$
\begin{aligned}
\langle i\ j\rangle \mathsf{hcom}_k^{0\to 1}\, A\, [(i=0) &\mapsto \mathsf{hcom}_l^{1\to 0}\, A\, [(k=0)\mapsto a, (k=1)\mapsto p\, l]\, (p\, k) \\
,(i=1) &\mapsto \mathsf{hcom}_l^{1\to j}\, A\, [(k=0)\mapsto a, (k=1)\mapsto p\, l]\, (p\, k) \\
,(j=0) &\mapsto \mathsf{hcom}_l^{1\to 0}\, A\, [(k=0)\mapsto a, (k=1)\mapsto p\, l]\, (p\, k) \\
,(j=1) &\mapsto \mathsf{hcom}_l^{1\to i}\, A\, [(k=0)\mapsto a, (k=1)\mapsto p\, l]\, (p\, k) \\
,(i=j) &\mapsto \mathsf{hcom}_l^{1\to i}\, A\, [(k=0)\mapsto a, (k=1)\mapsto p\, l]\, (p\, k)]\, a
\end{aligned}
$$

This has boundary:

$$
\begin{tikzcd}
a \arrow[r, "p"] & b \\
a \arrow[u, "a"] \arrow[r, "a"'] \arrow[ur, "p"] & a \arrow[u, "p"']
\end{tikzcd}
$$

Reversals and connections

These definitions of reversals and connections does not satisfy as many judgmental equalities as the corresponding ones in CCHM

How does this affect practical formalization?

Reversals and connections

These definitions of reversals and connections does not satisfy as many judgmental equalities as the corresponding ones in CCHM

How does this affect practical formalization?

For example in CCHM we directly get that $(C^{op})^{op} = C$ using reversals

However, many examples that use reversals and connections in CCHM can be done directly with the generalized Kan operations

Coercion back and forth

Given $i : \mathbb{I} \vdash A$ and $a : A(0/i)$ we have:

$$\begin{array}{c} a \\ A(0/i) \xrightarrow[A]{\hspace{6em}} A(1/i) \end{array}$$

Coercion back and forth

Given $i : \mathbb{I} \vdash A$ and $a : A(0/i)$ we have:

$$
\begin{tikzcd}
a \arrow[r, dashed] & \mathsf{coe}_i^{0 \to 1}\, A\, a \\
A(0/i) \arrow[r, "A"'] & A(1/i)
\end{tikzcd}
$$

Coercion back and forth

Given $i : \mathbb{I} \vdash A$ and $a : A(0/i)$ we have:

$$
\begin{tikzcd}
\mathrm{coe}_i^{1\to 0}\,A\,(\mathrm{coe}_i^{0\to 1}\,A\,a) & \\
a \arrow[r, dashed] & \mathrm{coe}_i^{0\to 1}\,A\,a \arrow[ul, dashed] \\
A(0/i) \arrow[r, "A"'] & A(1/i)
\end{tikzcd}
$$

Coercion back and forth

Given $i : \mathbb{I} \vdash A$ and $a : A(0/i)$ we have:

$$
\begin{tikzcd}
\mathsf{coe}_i^{1\to 0}\, A\, (\mathsf{coe}_i^{0\to 1}\, A\, a) & \\
a \arrow[u, "p?"] \arrow[r, dashed] & \mathsf{coe}_i^{0\to 1}\, A\, a \arrow[ul, dashed] \\
A(0/i) \arrow[r, "A"'] & A(1/i)
\end{tikzcd}
$$

Coercion back and forth

Given $i : \mathbb{I} \vdash A$ and $a : A(0/i)$ we have:

$$
\begin{tikzcd}
\mathsf{coe}_i^{1\to 0}\, A\, (\mathsf{coe}_i^{0\to 1}\, A\, a) & \\
a \arrow[u, "p?"] \arrow[r, dashed] & \mathsf{coe}_i^{0\to 1}\, A\, a \arrow[ul, dashed]
\end{tikzcd}
$$

$$A(0/i) \xrightarrow[A]{\hspace{4em}} A(1/i)$$

We take $p := \langle j \rangle\, \mathsf{coe}_i^{j\to 0}\, A\, (\mathsf{coe}_i^{0\to j}\, A\, a)$

Coercion back and forth

Given $i : \mathbb{I} \vdash A$ and $a : A(0/i)$ we have:

$$
\begin{tikzcd}
\mathsf{coe}_i^{1\to 0}\, A\, (\mathsf{coe}_i^{0\to 1}\, A\, a) & \\
a \arrow[u, "p?"] \arrow[r, dashed] & \mathsf{coe}_i^{0\to 1}\, A\, a \arrow[ul, dashed]
\end{tikzcd}
$$

$$
A(0/i) \xrightarrow[A]{} A(1/i)
$$

We take $p := \langle j \rangle\, \mathsf{coe}_i^{j\to 0}\, A\, (\mathsf{coe}_i^{0\to j}\, A\, a)$

The corresponding result in CCHM is quite a bit more involved (it uses 3 non-homogeneous compositions)

hcom and coe

We define the judgmental computation rules for hcom and coe by cases of the type A

There are no surprises for Π, Σ, Path and basic datatypes like $\mathbb{N}$

hcom and coe

We define the judgmental computation rules for hcom and coe by cases of the type A

There are no surprises for Π, Σ, Path and basic datatypes like $\mathbb{N}$

The algorithms for hcom and coe are often simpler than the corresponding ones for com, so I would conjecture that this decomposition is also good for efficiency

The decomposition is also very natural for formalization: we often want to compose in a constant type or just coerce without any constraints

Univalence and V-types

In order to be able to prove univalence we need a way to turn equivalences into paths in the universe

We could use CCHM Glue-types (as in ABCFHL), but instead we follow AFH and introduce a special case of Glue-types called "V-types"[3]

These allow us to "glue" an equivalence to one side of a line between types (i.e. to extend an equivalence along an endpoint inclusion)

[3] "V" in honour of Voevodsky

V-types

In the case when r is a dimension variable $i : \mathbb{I}$ the V-type $\mathsf{V}_i(A, B, e)$ can be drawn as the dashed line in:

$$
\begin{tikzcd}
A \arrow[d, "e"', "\wr"] \arrow[dr, dashed, "\mathsf{V}_i(A, B, e)"] & \\
B(0/i) \arrow[r, "B"'] & B(1/i)
\end{tikzcd}
$$

V-types typing rules (slide of death)

$$\frac{\Gamma \vdash r : \mathbb{I} \qquad \Gamma, r = 0 \vdash A \qquad \Gamma \vdash B \qquad \Gamma, r = 0 \vdash e : \mathsf{Equiv}\, A\, B}{\Gamma \vdash \mathsf{V}_r(A, B, e)[r = 0 \mapsto A, r = 1 \mapsto B]}$$

$$\frac{\begin{array}{c}\Gamma \vdash r : \mathbb{I} \qquad \Gamma, r = 0 \vdash u : A \\ \Gamma \vdash v : B[r = 0 \mapsto e\ u] \qquad \Gamma, r = 0 \vdash e : \mathsf{Equiv}\, A\, B\end{array}}{\Gamma \vdash \mathsf{Vin}_r\, u\, v : \mathsf{V}_r(A, B, e)[r = 0 \mapsto u, r = 1 \mapsto v]}$$

$$\frac{\Gamma \vdash r : \mathbb{I} \qquad \Gamma \vdash u : \mathsf{V}_r(A, B, e)}{\Gamma \vdash \mathsf{Vproj}_r\, u\, e : B[r = 0 \mapsto e\ u, r = 1 \mapsto u]}$$

V-types

Semantically V-types correspond to the following special case of Glue-types:

$$
\begin{tikzcd}
A \arrow[drr, "e", "\sim"'] \arrow[ddr, two heads] & & & & \\
& & B \arrow[rr] \arrow[dl, two heads] \arrow[dr, phantom, "\lrcorner", very near start] & & B \arrow[dl, two heads] \\
& \Gamma, (i = 0) \arrow[rr, tail] & & \Gamma &
\end{tikzcd}
$$

V-types

Semantically V-types correspond to the following special case of Glue-types:

$$
\begin{tikzcd}
A \arrow[rrr, dashed] \arrow[drr, "e", "\sim"'] \arrow[ddr, two heads] \arrow[dr, phantom, "\lrcorner", very near start] & & & \mathsf{V}_i(A, B, e) \arrow[drr, dashed, "Vproj_i"] \arrow[ddr, dashed, two heads] & & \\
& & B \arrow[rrr] \arrow[dl, two heads] \arrow[dr, phantom, "\lrcorner", very near start] & & & B \arrow[dl, two heads] \\
& \Gamma, (i = 0) \arrow[rrr, tail] & & & \Gamma &
\end{tikzcd}
$$

V-types: fibrancy

To prove that these types are fibrant we have to define:

$$\mathsf{hcom}_i^{r \to s}\,(\mathsf{V}_j(A, B, e))\,[\varphi \mapsto u]\,u_0 \qquad\qquad \mathsf{coe}_i^{r \to s}\,(\mathsf{V}_j(A, B, e))\,u$$

Both hcom, and coe when $i \neq j$, are straightforward

Only coe when $i = j$ requires e to be an equivalence, furthermore this case crucially uses the diagonal constraints/cofibrations

None of the cases uses the $\forall i$ operation of CCHM

V-types: $\mathsf{coe}_i^{r \to s}\,(\mathsf{V}_i(A, B, e))\,u : V_s[r = s \mapsto u]$

For this let:

$$
\begin{aligned}
u' &:= \mathsf{Vproj}_r\,u\,e(r/i) \\
P &:= \mathsf{coe}_i^{r \to s}\,B\,u' \\
(C_1, C_2) &:= e(s/i).2\ P \\
R &:= \mathsf{hcom}_k^{1 \to 0}\,(\mathsf{Fiber}\,e(s/i)\,P)\,[r = 0 \mapsto C_2\,(u, \langle \_ \rangle P)\,k \\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad , r = 1 \mapsto C_1]\,C_1 \\
S &:= \mathsf{hcom}_k^{1 \to 0}\,B(s/i)\,[s = 0 \mapsto R.2\,k \\
&\qquad\qquad\qquad\qquad\quad , r = s \mapsto \mathsf{Vproj}_s\,u\,e(s/i)]\,P
\end{aligned}
$$

and we define

$$\mathsf{coe}_i^{r \to s}\,(\mathsf{V}_i(A, B, e))\,u := \mathsf{Vin}_s\,R.1\,S$$

Fibrant universes

We also have universes in yacctt, however as we are not using Glue-types we have to do more work to prove that they are fibrant

We follow a direct argument from AFH for glueing on lines of types onto a line of types[4]

The coe operation uses $\forall i$, and both coe and hcom uses the diagonal constraints/cofibrations in a crucial way

[4] This is similar to an unfolded version of composition for the universe in CCHM, which in fact is what we implemented in cubicaltt for efficiency

Univalence and V-types

I have formalized two proofs of univalence in yacctt

The first proof uses the observation that we can prove the full univalence axiom from an operation

$$\mathsf{ua} : \mathsf{Equiv}\, A\, B \to \mathsf{Path}\, \mathsf{U}\, A\, B$$

satisfying

$$\mathsf{ua}_\beta : \mathsf{Path}\, B\, (\mathsf{coe}_i^{0 \to 1}\, (\mathsf{ua}\, e\, i)\, a)\, (e\, a)$$

for all $a : A$

Univalence and V-types

Given $e : \mathsf{Equiv}\, A\, B$ we define:

$$\mathsf{ua} := \langle i \rangle\, \mathsf{V}_i(A, B, e)$$

Univalence and V-types

Given $e : \mathsf{Equiv}\, A\, B$ we define:

$$\mathsf{ua} := \langle i \rangle\, \mathsf{V}_i(A, B, e)$$

If we unfold the algorithm for coercion in V-types we see that

$$\begin{aligned}\mathsf{coe}_i^{0\to 1}\,(\mathsf{ua}\; i)\; a &= \mathsf{coe}_i^{0\to 1}\,(\mathsf{V}_i(A, B, e))\; a \\ &= \mathsf{coe}_i^{0\to 1}\, B\,(e\; a)\end{aligned}$$

Univalence and V-types

Given $e : \mathsf{Equiv}\, A\, B$ we define:

$$\mathsf{ua} := \langle i \rangle\, \mathsf{V}_i(A, B, e)$$

If we unfold the algorithm for coercion in V-types we see that

$$\begin{aligned} \mathsf{coe}_i^{0 \to 1}\, (\mathsf{ua}\; i)\, a &= \mathsf{coe}_i^{0 \to 1}\, (\mathsf{V}_i(A, B, e))\, a \\ &= \mathsf{coe}_i^{0 \to 1}\, B\, (e\, a) \end{aligned}$$

We can hence define

$$\mathsf{ua}_\beta := \langle i \rangle \mathsf{coe}_i^{i \to 1}\, B\, (e\, a)$$

This is simpler than in cubicaltt where the algorithm for composition for Glue-types gives two trivial compositions

V-types and univalence

The second proof of univalence is similar to the one in CCHM where we show that unglue is an equivalence:

$$
\begin{tikzcd}
A \arrow[rr, dashed] \arrow[dr, "e", "\sim"'] \arrow[dr, phantom, very near start, "\lrcorner"] \arrow[dd, two heads] & & \mathsf{V}_i(A, B, e) \arrow[dr, dashed, two heads, "\mathsf{Vproj}_i", "\sim"'] \arrow[dd, dashed, two heads] & \\
& B \arrow[rr] \arrow[dl, two heads] \arrow[dr, phantom, very near start, "\lrcorner"] & & B \arrow[dl, two heads] \\
\Gamma, (i = 0) \arrow[rr, tail] & & \Gamma &
\end{tikzcd}
$$

V-types and univalence

From this we can directly prove that given any type $A : \mathrm{U}$ the type $(B : \mathrm{U}) \times \mathrm{Equiv}\, A\, B$ is contractible[5]

Corollary (Univalence)

For any term

$$t : (A\ B : \mathrm{U}) \to \mathrm{Path}\ \mathrm{U}\ A\ B \to \mathrm{Equiv}\, A\, B$$

the map $t\ A\ B$ is an equivalence for all A and B

[5] I was a bit surprised that this worked out so smoothly

Higher inductive types

We have so far only added support for a few hardcoded HITs, but it should be possible to add a schema of them following

Computational Higher Type Theory IV: Inductive Types
(Cavallo, Harper)

The algorithms for coe in HITs are very similar to those in CCHM:

On Higher Inductive Types in Cubical Type Theory
(Coquand, Huber, M.)

Conclusions

We have implemented a simple experimental proof assistant based on Cartesian cubical type theory

Some proofs are simpler compared to cubicaltt, while some are a bit harder

I'm optimistic that the V-types might be a bit more efficient than Glue-types for computing with univalence

Open questions

- Can we make progress on computing Brunerie's number using yacctt?
- Can we design a super cubical type theory with connections, reversals and generalized Kan operations?

Thank you for your attention!

`https://github.com/mortberg/yacctt/`