From HoTT to HOTT

Autonomy in new foundations for mathematics

Michael Shulman

University of San Diego

January 31, 2024

"Is Philosophy Useful for Science, and/or Vice Versa?"

Chapman University

About this talk

This talk is essentially an extended personal anecdote, illustrating one way that philosophy can be useful for science.

1. I am a mathematician, albeit with a "philosophical bone".
2. We have proposed Homotopy type theory (HoTT) as a new foundation for mathematics.
3. Philosophers have critiqued this proposal, clarifying what such a claim should mean.
4. In response, we are formulating Higher Observational Type Theory (HOTT), with a better claim to be foundational, and which may have mathematical advantages as well.

Outline

Theory of types

The theory of types is intended to be a full scale system for formalizing intuitionistic mathematics as developed, for example, in the book by Bishop 1967...

[A] mathematical object is always given together with its type, that is, it is not just an object, it is an object of a certain type. This may be regarded as a simpler and at the same time more general formulation of Russell's 1903 doctrine of types...

A type is defined by prescribing what we have to do in order to construct an object of that type. This is almost verbatim the definition of the notion of set given by Bishop... a type is welldefined if we understand... what it means to be an object of that type. Thus... $\mathbb{N} \to \mathbb{N}$ is a type not because we know particular number theoretic functions... but because we think we understand the notion of number theoretic function in general.

– Martin-Löf, "An intuitionistic theory of types", 1975

Nowadays, we write $a : A$ to mean that a is an object of type A.

Propositions as types

A proposition is defined by prescribing how we are allowed to prove it. . . . For example

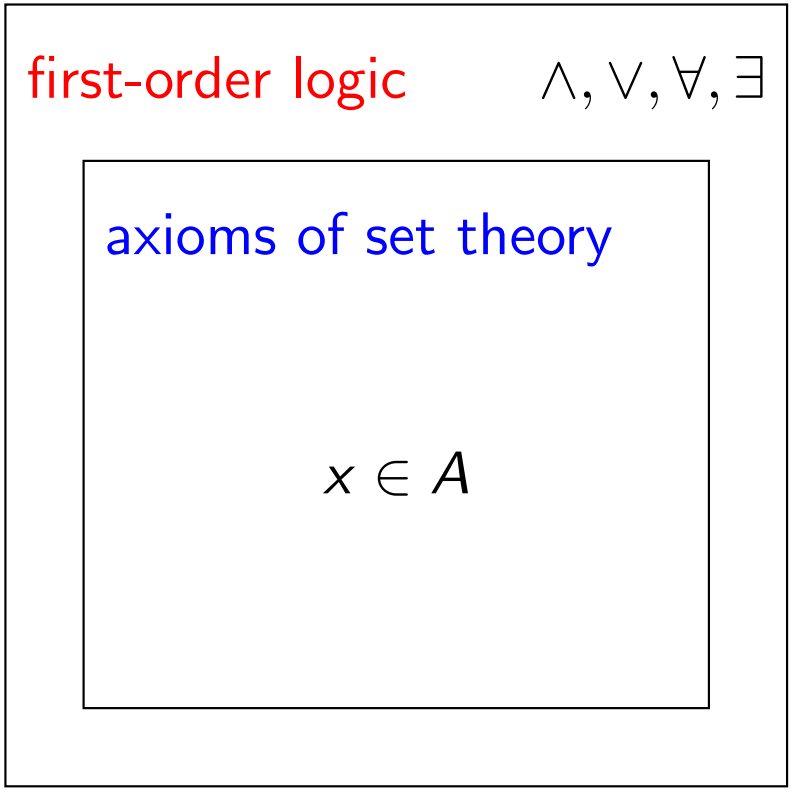
971 is a non prime number

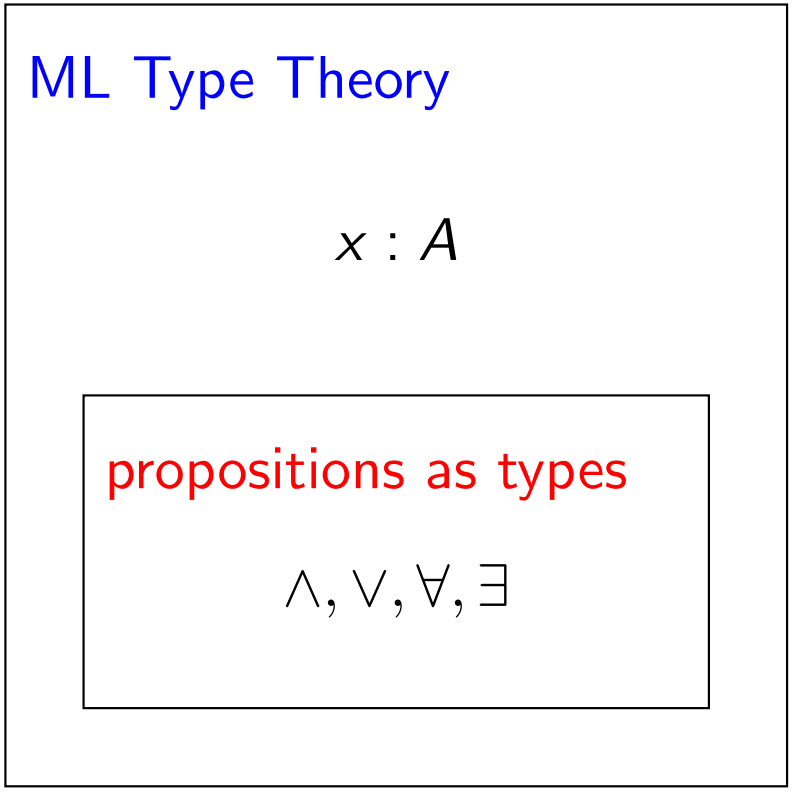
is the proposition which we prove by exhibiting two natural numbers greater than one and a computation which shows that their product equals 971. . . . it will not be necessary to introduce the notion of proposition as a separate notion because we can represent each proposition by a certain type, namely, the type of proofs of that proposition. . . .

Conversely, each type determines . . . the proposition that the type in question is nonempty. . . which we prove by exhibiting an object of the type in question. . . . the difference [between propositions and types] is one of point of view: in the case of a proposition, we are not so much interested in what its proofs are as in whether it has a proof. . . whereas in the case of a type, we are of course interested in what its objects are and not only in whether it is empty or nonempty.

– *Martin-Löf, "An intuitionistic theory of types: predicative part", 1975*

Sets in logic vs. logic in types





Propositions-as-types $=$ the Curry–Howard correspondence

Example: dependent function-types

> *Suppose now that A is a type and that B is a function... which to an arbitrary object a of type A assigns a type $B(a)$. Then... $(x : A) \to B(x)$ is... the type of functions which take an arbitrary object a of type A into an object of type $B(a)$....*
>
> *When $B(a)$ represents a proposition... [this type] represents the universal proposition $\forall(x : A)\, B(x)$. A proof of $\forall(x : A)\, B(x)$ is a function which to an arbitrary object a of type A assigns a proof of $B(a)$....*
>
> *When A and B both represent propositions, [this type] represents the implication $A \to B$. A proof of $A \to B$ is a function which takes an arbitrary proof of A into a proof of B.*
>
> *– Martin-Löf, "An intuitionistic theory of types", 1975*

[Here and elsewhere, I've updated notation and punctuation to align with modern usage, and added emphasis in some places.]

Elements of function-types

Each type has some rules that tell us what its elements are:

> *. . . if we, starting from a variable x that denotes an arbitrary object of type A, build up a term $b[x]$ that denotes an object of type $B(x)$, then we may define a function denoted $\lambda x.b[x]$ of type $(x : A) \to B(x)$. . .*

And others, determined by this, that say how we can use them:

> *We may apply an object b of type $(x : A) \to B(x)$ to an object of type A, thereby getting an object $b(a)$ of type $B(a)$. . . .*
>
> *– Martin-Löf, "An intuitionistic theory of types", 1975*

Example: The type of natural numbers

$\mathbb{N}$ *is a type, namely the type of natural numbers.*

Its elements are:

0 *is an object of type* $\mathbb{N}$ *and, if* n *is an object of type* $\mathbb{N}$*, so is its successor* $s(n)$*. These are the first two Peano axioms.*

which determines how we can use them:

Let C *be a function which to an arbitrary natural number assigns a type. Then, given an object* d *of type* $C(0)$ *and a function* e *of type* $(x : \mathbb{N}) \to C(x) \to C(s(x))$*, we may introduce a function of type* $(x : \mathbb{N}) \to C(x)$ *. . . by the recursion schema*

$$\begin{cases} R(0, d, e) & = d, \\ R(s(n), d, e) & = e(n, R(n, d, e)) \end{cases}$$

If $C(n)$ *represents a proposition for every natural number* n*, then* R *is the proof of the universal proposition* $\forall(x : \mathbb{N})\, C(x)$ *which we get by applying the principle of mathematical induction.*

– Martin-Löf, "An intuitionistic theory of types", 1975

Very important example: Identity types

> *If x and y are objects of... the same type A, then* $\mathrm{Id}_A(x, y)$ *is a proposition, namely, the proposition that x and y are identical.*

Its elements are:

> *[I]f x is an arbitrary object of type A, then* refl_x *is a proof of* $\mathrm{Id}_A(x, x)$

which determines how we can use them:

> *Let C be a ternary function which to an arbitrary pair of objects x and y of type A and a proof of* $\mathrm{Id}_A(x, y)$ *assigns a type. Given a unary function g [of type* $(x : A) \to C(x, x, \mathrm{refl}_x)$*], we may then define a ternary function f [of type* $(x : A)(y : A)(z : \mathrm{Id}_A(x, y)) \to C(x, y, z)$*] by the schema*
>
> $$f(x, x, \mathrm{refl}_x) = g(x).$$
>
> *... In particular, [this implies] the usual eliminatory axiom of identity*
>
> $$\forall(x : A)\,\forall(y : A)\left(\mathrm{Id}_A(x, y) \to \big(C(x) \to C(y)\big)\right).$$
>
> *– Martin-Löf, "An intuitionistic theory of types: predicative part", 1975*

Example: Universes

. . . we introduce a type "$\mathcal{U}$" which will be called a universe and whose objects are to be types, together with the reflection principle. . . that whatever we are used to doing with types can be done inside the universe. . .

- If $A : \mathcal{U}$ and $B : A \to \mathcal{U}$, then $\big((x : A) \to B(x)\big) : \mathcal{U}$.
- $\mathbb{N} : \mathcal{U}$
- If $A : \mathcal{U}$ and $a : A$ and $b : A$, then $\mathsf{Id}_A(a, b) : \mathcal{U}$.
- etc.

. . . the reflection principle does not justify the axiom that $\mathcal{U}$ is an object of type $\mathcal{U}$ . . . because then $\mathcal{U}$ would, so to say, have to have been there already before we introduced it. . . .

[This] can be iterated so as to obtain a whole sequence of universes $\mathcal{U}_0, \mathcal{U}_1, \ldots, \mathcal{U}_n, \ldots$.

– Martin-Löf, "An intuitionistic theory of types: predicative part", 1975

Constructive mathematics and computer programming

Theorem (Canonicity)

In MLTT, every closed expression (i.e. no free variables) reduces to a canonical one obtained from the primitive rules for defining elements of its type.

For instance, $s(s(0)) + s(s(0)) : \mathbb{N}$ reduces to $s(s(s(s(0))))$.
This makes MLTT a dependently typed programming language.

In addition, a machine implementation of MLTT yields a 'proof assistant' to formally verify constructive mathematics
(or classical mathematics, by assuming LEM).

Outline

Uniqueness of identity proofs

It is now a natural question to ask. . . whether any two elements of an identity [type] are equal. We will call [the latter] Uniqueness of Identity Proofs. . . .

. . . the intuition that a type is determined by its canonical objects might be seen as evidence for the validity of UIP, as the identity [types] have at most one canonical element corresponding to a proof of reflexivity.

– Hofmann–Streicher, "The groupoid interpretation of type theory", 1996

Uniqueness of identity proofs

It is now a natural question to ask... whether any two elements of an identity [type] are equal. We will call [the latter] Uniqueness of Identity Proofs....

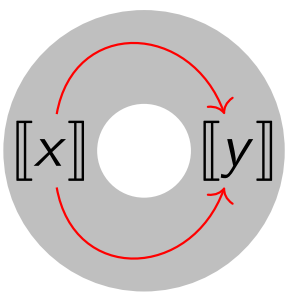
... the intuition that a type is determined by its canonical objects might be seen as evidence for the validity of UIP, as the identity [types] have at most one canonical element corresponding to a proof of reflexivity.

– Hofmann–Streicher, "The groupoid interpretation of type theory", 1996

The topological interpretation
(Hofmann–Streicher 1996, Awodey–Warren 2009, Voevodsky 2009)

We can interpret each type A as a space $[\![A]\!]$.
Then $[\![\mathrm{Id}_A(x,y)]\!]$ is the space of paths from $[\![x]\!]$ to $[\![y]\!]$.

Two paths from $[\![x]\!]$ to $[\![y]\!]$ might not be connected by any path in the "space of paths", so UIP fails.



Anima

More generally*, we can interpret types as anima[1], consisting of

- A set X_0 of objects or 0-cells.
- For $x, y \in X_0$, a set $X_1(x, y)$ whose elements are called equivalences, paths, identifications, or 1-cells.
- For $e, f \in X_1(x, y)$, a set $X_2(e, f)$ of 2-cells.
- Similarly, sets $X_n(\dots)$ of n-cells for all $n \in \mathbb{N}$.
- Reflexivity, transitivity, symmetry, and infinite higher ops.

[1] a.k.a. ∞-groupoids — the name "anima" is due to Clausen and Scholze

Anima

More generally*, we can interpret types as anima[1], consisting of

- A set X_0 of objects or 0-cells.
- For $x, y \in X_0$, a set $X_1(x, y)$ whose elements are called equivalences, paths, identifications, or 1-cells.
- For $e, f \in X_1(x, y)$, a set $X_2(e, f)$ of 2-cells.
- Similarly, sets $X_n(\dots)$ of n-cells for all $n \in \mathbb{N}$.
- Reflexivity, transitivity, symmetry, and infinite higher ops.

Anima are everywhere in mathematics. Almost any "collection" has a natural notion of identification between its elements.

- Any set, equalities, equalities, equalities,...
- Any space, paths, deformations, higher deformations, ...
- Sets, bijections, equalities, equalities, ...
- Groups, isomorphisms, equalities, equalities, ...
- Categories, equivalences, natural isomorphisms, equalities, ...

[1] a.k.a. ∞-groupoids — the name "anima" is due to Clausen and Scholze

The anima-tion of mathematics

. . . after Cantor and Bourbaki . . . set theoretic mathematics resides in our brains. When I first start talking about something, I explain it in terms of Bourbaki-like structures . . . we start with the discrete sets of Cantor, upon which we impose something more in the style of Bourbaki.

But fundamental psychological changes also occur. . . . the place of old forms and structures . . . is taken by some geometric, right-brain objects.

. . . there is an ongoing reversal in the collective consciousness of mathematicians: the. . . homotopical picture of the world becomes the basic intuition, and if you want to get a discrete set, then you pass to the set of connected components. . .

From Interview with Yuri Manin (by Mikhail Gelfand), AMS Notices, October 2009

The anima-tion of mathematics

. . . after Cantor and Bourbaki . . . set theoretic mathematics resides in our brains. When I first start talking about something, I explain it in terms of Bourbaki-like structures . . . we start with the discrete sets of Cantor, upon which we impose something more in the style of Bourbaki.

But fundamental psychological changes also occur. . . . the place of old forms and structures . . . is taken by some geometric, right-brain objects.

. . . there is an ongoing reversal in the collective consciousness of mathematicians: the. . . homotopical picture of the world becomes the basic intuition, and if you want to get a discrete set, then you pass to the set of connected components. . .

From Interview with Yuri Manin (by Mikhail Gelfand), AMS Notices, October 2009

BUT in set theory, anima are complicated and hard to use.

Homotopy type theory

A radical idea

Take the anima interpretation of MLTT as the intended one!

This allows us to work with anima synthetically, without needing the complicated explicit definition in terms of sets.
We recover "sets", up to isomorphism, as the "0-truncated" anima.

Homotopy type theory

A radical idea

Take the anima interpretation of MLTT as the intended one!

This allows us to work with anima synthetically, without needing the complicated explicit definition in terms of sets.
We recover "sets", up to isomorphism, as the "0-truncated" anima.

We need something to replace UIP, ensuring that types behave like anima rather than sets.

The univalence axiom
(Voevodsky 2009, foreshadowed by Hofmann–Streicher 1996)

For types $A : \mathcal{U}$ and $B : \mathcal{U}$, the type $\mathrm{Id}_{\mathcal{U}}(A, B)$ is equivalent to the type of one-to-one correspondences between A and B.

Univalent mathematics is structural mathematics

> *[T]he Principle of Structuralism [is that] isomorphic objects are identical. . . . Within a mathematical theory, theorem, or proof, it makes no practical difference which of two "isomorphic copies" are used, and so they can be treated as the same mathematical object for all practical purposes. . . . mathematicians have developed a sort of systematic sloppiness to help them implement this principle. . . despite being . . . incompatible with conventional foundations of mathematics in set theory.*
>
> *– Steve Awodey, "Structuralism, invariance, and univalence", 2014*

For example, if $G \cong H$ are isomorphic groups, and G has some property (cyclic, abelian, solvable, simple, . . .), then so does H. But, $1 \in \mathbb{Z}$, whereas $1 \notin 2\mathbb{Z}$.

In type theory with the univalence axiom, a group isomorphism $G \cong H$ promotes to an identification $s : \mathsf{Id}_{\mathsf{Group}}(G, H)$, whence substitution implies $P(G) \to P(H)$ for any property P.

Univalent foundations

> *This suggests a new conception of foundations of mathematics, with intrinsic [anima-ted] content, an "invariant" conception of the objects of mathematics — and convenient machine implementations, which can serve as a practical aid to the working mathematician. This is the Univalent Foundations program. . . .*
>
> *[U]nivalent foundations is very much a work in progress. . . The ultimate theory will almost certainly not look exactly like the one described in this book, but it will surely be at least as capable and powerful; we therefore believe that univalent foundations will eventually become a viable alternative to set theory as the "implicit foundation" for the unformalized mathematics done by most mathematicians.*
>
> – *"Homotopy type theory: univalent foundations of mathematics" (a.k.a. "The HoTT Book"), 2013*

Martin-Löf Type Theory plus the Univalence Axiom (and some other things) is known as Book HoTT after this book.

Outline

Is that valid?

Is homotopy type theory really a foundation for mathematics?

Of course now we have to ask

So what is a foundation for mathematics, anyway?

The Pragmatic Mathematician's Answer

A formal system that is sufficiently expressive for all existing mathematics to be encoded into it. In other words, a system that is "mathematics-complete", by analogy to NP-complete problems and Turing-complete programming languages.

If this is what a foundation for mathematics is, then yes, obviously, homotopy type theory is a foundation for mathematics.

But should a foundation be more than that?

Voevodsky's answer

> *[A]ny foundation for mathematics. . . should have the following three components. The first component is a formal deduction system: a language and rules of manipulating sentences in this language that are purely formal, such that a record of such manipulations can be verified by a computer program. The second component is a structure that provides a meaning to the sentences of this language in terms of mental objects intuitively comprehensible to humans. The third component is a structure that enables humans to encode mathematical ideas in terms of the objects directly associated with the language.*
>
> *– Voevodsky, "The origins and motivations of univalent foundations", 2014*

The pragmatic answer covers the first and third components. Voevodsky adds to this that the system should have an intuitively comprehensible meaning.

The intuition of anima

So what is the "intuitive meaning" of homotopy type theory?

> *The second component of univalent foundations, the structure that provides a direct meaning to the sentences. . . is based on univalent models. The objects directly associated with sentences. . . by these models are called [anima]. The world of [anima] is stratified by what we call h-levels, with types of h-level 1 corresponding to logical propositions and types of h-level 2 corresponding to sets. Our intuition about types of higher levels comes mostly from their connection with multidimensional shapes, which was studied by ZFC-based mathematics for several decades.*
>
> *– Voevodsky, "The origins and motivations of univalent foundations", 2014*

So even according to Voevodsky, our intuition about homotopy type theory comes from set theoretic mathematics.

A philosopher's answer

Philosophically, this is a problem for the claim that homotopy type theory is a foundation for mathematics.

> *[A] putative foundation for mathematics must boast more than mere logical autonomy with respect to set theory if it is to be truly autonomous. It must be possible not only to formulate the foundation without presupposing a theory of sets; it must be possible also to understand it and to justify its claims without such a presupposition.*
>
> – *Linnebo–Pettigrew, "Category theory as an autonomous foundation", 2011*

> *An autonomous foundation must therefore use only concepts that can be pre-mathematically understood, and rules that can be pre-mathematically motivated. If any aspect of a purported foundation for mathematics relies for its formulation or justification upon some advanced area of mathematics then it cannot be a foundation for mathematics. . .*
>
> – *Ladyman–Presnell, "Identity in Homotopy Type Theory, Part I: The Justification of Path Induction", 2014*

Understanding and justifying type theory

Can we piggyback on the understanding and justification of Martin-Löf type theory?

> *Type-theoretically, harmony states that the introduction rules of a type specify the only way of generating terms of that type, so in order to define a function out of that type it is enough to determine what it should do on terms explicitly obtained by introduction rules.* – *Bentzen, "What types should not be", 2020*

- The only terms of $(x : A) \to B(x)$ are $\lambda x.b[x]$, so the only way to use such a term is to apply it to an argument.
- The only terms of $\mathbb{N}$ are 0 and $s(n)$, so to define a function $(z : \mathbb{N}) \to C(z)$ it suffices to define it on terms of these forms.
- The only terms of $\mathrm{Id}_A(x, y)$ are of the form refl_x, so to define a function $(x : A) \to (y : A) \to (z : \mathrm{Id}_A(x, y)) \to C(x, y, z)$ it suffices to define it on terms of these forms.

Understanding and justifying type theory

Can we piggyback on the understanding and justification of Martin-Löf type theory?

> *Type-theoretically, harmony states that the introduction rules of a type specify the only way of generating terms of that type, so in order to define a function out of that type it is enough to determine what it should do on terms explicitly obtained by introduction rules.* – *Bentzen, "What types should not be", 2020*

- The only terms of $(x : A) \to B(x)$ are $\lambda x.b[x]$, so the only way to use such a term is to apply it to an argument.
- The only terms of $\mathbb{N}$ are 0 and $s(n)$, so to define a function $(z : \mathbb{N}) \to C(z)$ it suffices to define it on terms of these forms.
- ~~The only terms of $\mathrm{Id}_A(x, y)$ are of the form refl_x~~, so to define a function $(x : A) \to (y : A) \to (z : \mathrm{Id}_A(x, y)) \to C(x, y, z)$ it suffices to define it on terms of these forms.

Not true with univalence!!

Justifying univalent Id-elimination

. . . we might say that [Id-elimination] expresses the fact that every [proof of identity] is of the form refl_a*. . . this reading is quite confusing in the context of the homotopy interpretation. . . where there may be. . . many different elements of the identity type!*

. . . it is not the identity type that is inductively defined, but the identity family. . . the type of triples (x, y, p)*, where* $[p : \mathsf{Id}_A(x, y)]$*. . . [T]he space corresponding to [this type] is the free path space — the space of paths in* A *whose endpoints may vary — and it is in fact the case that any point of this space is homotopic to the constant loop at some point, since we can simply retract one of its endpoints along the given path.*

– *"Homotopy type theory: univalent foundations of mathematics", 2013*

While this argument provides a justification for [Id-elimination], it clearly relies upon the details of homotopy theory for its motivation, and so this approach. . . is not suitable for an autonomous foundation for mathematics.

– *Ladyman–Presnell, "Identity in Homotopy Type Theory, Part I: The Justification of Path Induction", 2014*

Computation vs univalence

We could attempt a similar justification using only pre-mathematical intuition about "space", but one precise consequence of the usual "harmony" objectively fails for Book HoTT: the fact that every closed expression reduces to a canonical one.

Example

1. Choose any one-to-one correspondence $\mathbb{N} \rightleftarrows \mathbb{N}$.
2. Use univalence to make it into an identity $e : \mathsf{Id}_{\mathcal{U}}(\mathbb{N}, \mathbb{N})$.
3. Use Id-elimination to "substitute" $0 : \mathbb{N}$ along e.

We get an expression in $\mathbb{N}$ that cannot be reduced: the computation gets "stuck" on the "computationally unjustified" univalence axiom.

One can argue this invalidates any attempt to invoke harmony, despite whatever we might write about "pre-mathematical spaces".

Cubical type theory

BUT: Unlike for other axioms (like excluded middle), there is a clear candidate for how stuck univalence terms should compute: Substituting along any $s : \mathrm{Id}_{\mathcal{U}}(A, B)$ should reduce to applying the corresponding isomorphism $A \xrightarrow{\sim} B$.

Cubical Type Theory (Coquand et. al. 2016, 2019, etc.) makes univalence compute by

- replacing Martin-Löf's $\mathrm{Id}_A(x, y)$ with path types, whose elements are functions $f : \mathtt{I} \to A$ such that $f(0) = x$ and $f(1) = y$. Here $\mathtt{I}$ is an abstract interval with two points $0, 1$.
- Specifying infinitely many rules for composing and transporting paths and higher-dimensional paths, and how these operations compute in different types.

Martin-Löf's Id-elimination is a theorem in CTT, not a primitive rule; so it doesn't need pre-mathematical justification.

Justifying cubical type theory

Essentially, CTT builds the definition of anima into the type theory, explicitly including all the higher composition operations.

> *Recently, Angiuli et. al. (2017) and Cavallo and Harper (2018) have built a realizability model that can be seen as a higher-dimensional generalization of the meaning explanations of Martin-Löf for a cubical type theory. . . . [This] model may indeed be seen as a computational justification for homotopy type theory, but, as it relies on the mathematical concept of cubical set, further work would be required to investigate whether it could also constitute a legitimate pre-mathematical justification for homotopy type theory.*
>
> *– Bentzen, "What types should not be", 2020*

The presentation of anima in Cubical Type Theory is quite technical, so justifying it "pre-mathematically" is a tall order.

Outline

Higher Observational Type Theory

Higher Observational Type Theory (H.O.T.T.) is a proposed third alternative to Book HoTT and Cubical Type Theory, in which:

1. As in Cubical Type Theory, univalence is not stuck, Id-elimination is a theorem, and conjecturally all closed expressions reduce to canonical form.
2. As in Book HoTT, the higher anima operations are all induced from low-dimensional structure.
3. Arguably, all the rules can be given an autonomous pre-mathematical justification from a few simple principles.

Altenkirch, Kaposi, Chamoun, and I have been developing HOTT for several years, and I believe it is approaching completion.

While we hope that HOTT will have technical advantages as well, my primary motivation for it was philosophical.

Back to Bishop

Recall how Martin-Löf defined types:

> *A type is defined by prescribing what we have to do in order to construct an object of that type. This is almost verbatim the definition of the notion of set given by Bishop 1967.*
>
> *– Martin-Löf, "An intuitionistic theory of types", 1975*

But here's what Bishop actually said:

> *A set is defined by describing exactly what must be done in order to construct an element of the set and what must be done in order to show that two elements are equal.*
>
> *– Bishop, "Foundations of Constructive Analysis", 1967*

Back to Bishop

Recall how Martin-Löf defined types:

> *A type is defined by prescribing what we have to do in order to construct an object of that type. This is almost verbatim the definition of the notion of set given by Bishop 1967.*
>
> *– Martin-Löf, "An intuitionistic theory of types", 1975*

But here's what Bishop actually said:

> *A set is defined by describing exactly what must be done in order to construct an element of the set and what must be done in order to show that two elements are equal.*
>
> *– Bishop, "Foundations of Constructive Analysis", 1967*

In HOTT, we stipulate that:

- Every type X has intrinsic identity types $\mathsf{Id}_X(x, y)$.
- These are defined separately for each primitive type.
- Every element $x : X$ has a reflexivity term refl_a, also defined separately for each primitive term.

Examples: identity types of $\mathbb{N}$

Example

The elements of $\mathsf{Id}_{\mathbb{N}}(u, v)$ are:

- We have $0' : \mathsf{Id}_{\mathbb{N}}(0, 0)$.
- For any $n_2 : \mathsf{Id}_{\mathbb{N}}(n_0, n_1)$, we have $s'(n_2) : \mathsf{Id}_{\mathbb{N}}(s(n_0), s(n_1))$.

Reflexivity is defined recursively:

$$\mathsf{refl}_0 = 0'$$
$$\mathsf{refl}_{s(n)} = s'(\mathsf{refl}_n)$$

Example: identity types of function types

Example

Two functions are equal if they map equal elements to equal elements.

$$\mathsf{Id}_{A\to B}(f_0, f_1) = (x_0 : A) \to (x_1 : A) \to (x_2 : \mathsf{Id}_A(x_0, x_1)) \to \mathsf{Id}_B(f_0(x_0), f_1(x_1))$$

- This includes pointwise equality: apply to x, x, and refl_x.
- Reflexivity proves that functions respect equality:

$$\mathsf{refl}_f : (x_0 : A) \to (x_1 : A) \to (x_2 : \mathsf{Id}_A(x_0, x_1)) \to \mathsf{Id}_B(f(x_0), f(x_1))$$

Identity types of the universe

Finally, univalence is now a definition in the same style:

Example

For types $A : \mathcal{U}$ and $B : \mathcal{U}$, an element $s : \mathsf{Id}_{\mathcal{U}}(A, B)$ is a one-to-one correspondence, consisting of:

- A correspondence $\lfloor s \rfloor : A \to B \to \mathcal{U}$.
- For each $a : A$ there is a unique pair of $b : B$ and $r : \lfloor s \rfloor(a, b)$.
- For each $b : B$ there is a unique pair of $a : A$ and $r : \lfloor s \rfloor(a, b)$.

The reflexivity correspondence of a type is its identity type:

$$\lfloor \mathsf{refl}_A \rfloor(x, y) = \mathsf{Id}_A(x, y)$$

Dependent identity types

An element of $\lfloor s \rfloor(a, b)$ is a proof that a equals b "along" the equality s between their types.

Example

For dependent functions, with $A : \mathcal{U}$ and $B : A \to \mathcal{B}$, we have

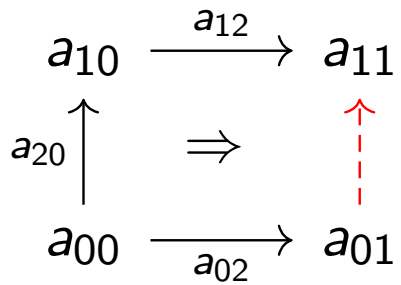
$$\mathsf{Id}_{(x:A)\to B(x)}(f_0, f_1) = (x_0 : A) \to (x_1 : A) \to (x_2 : \mathsf{Id}_A(x_0, x_1)) \to \lfloor \mathsf{refl}_B(x_0, x_1, x_2) \rfloor(f_0(x_0), f_1(x_1))$$

Square fillers

We have $\mathsf{Id}_A : A \to A \to \mathcal{U}$, hence given $a_{02} : \mathsf{Id}_A(a_{00}, a_{01})$ and $a_{12} : \mathsf{Id}_A(a_{10}, a_{11})$, we have

$$\mathsf{refl}_{\mathsf{Id}_A}(a_{00}, a_{01}, a_{02}, a_{10}, a_{11}, a_{12}) : \mathsf{Id}_{\mathcal{U}}(\mathsf{Id}_A(a_{00}, a_{10}), \mathsf{Id}_A(a_{01}, a_{11})).$$

In particular, for any $a_{20} : \mathsf{Id}_A(a_{00}, a_{10})$ we have a corresponding element of $\mathsf{Id}_A(a_{01}, a_{11})$.



This specializes to:

- transitivity if $a_{02} = \mathsf{refl}$, and
- symmetry if $a_{20} = a_{12} = \mathsf{refl}$,

and generalizes to all the higher structure of an anima.

Summary: the basic principles of HOTT

1. A type is defined by prescribing what we have to do in order to construct an object of that type, and what it means for two elements of that type to be equal. This notion of equality, being a binary predicate, is also a family of types, specified coinductively in the same general class of types being defined.
2. Every element of a type is canonically equal to itself. These reflexivity proofs must be specified along with the elements when a type is defined.
3. Identifications between types are one-to-one correspondences.
4. The reflexivity one-to-one correspondence of a type consists of its identity types.

Justifying HOTT

1. Like Book HoTT and CTT, HOTT is a complete formal system, formulated without presupposing set-theoretic foundations.
2. The basic principles of HOTT do not refer to sets, anima, cubical sets, or other concepts from advanced mathematics. Thus, unlike CTT, it can be understood without presupposing set-theoretic foundations.
3. *Conjecturally*, HOTT satisfies a canonicity property like MLTT: every closed expression reduces to a canonical one. Thus, unlike Book HoTT, it can be justified by logical harmony like MLTT, without presupposing any set-theoretic foundations.

Summary

1. Philosophy helped motivate Higher Observational Type Theory, a new foundation for higher structural mathematics that is truly autonomous from set theory.
2. HOTT is unfinished, but its techniques have already inspired the design of other new type theories.
3. Potential technical advantages of HOTT over CTT.
4. HOTT's philosophical advantages may also be pedagogical advantages for students, mathematicians, and programmers.