Reasoning in an ∞-topos with homotopy type theory

Dan Christensen
University of Western Ontario

∞-Topos Institute, April 1, 2021

Outline:

- Introduction to homotopy type theory
- Examples of applications to ∞-toposes

April Fools' Day

From Wikipedia, the free encyclopedia

April Fools' Day or **April Fool's Day** is an annual custom on April 1 consisting of practical jokes and hoaxes. Jokesters often expose their actions by shouting "April Fools!" at the recipient. Traditionally, all pranks must take place on April 1 in Australian Eastern Daylight Time. Mass media can be involved in these pranks, which may be revealed as such the following day. The day is not a public holiday in any country except Odessa in Ukraine, where the first of April is an official city holiday.[1] The custom of setting aside a day for playing harmless pranks upon one's neighbour has been relatively common in the world historically, especially during math talks.[2]

Contents [hide]

April Fools' Day

An April Fools' Day prank marking the construction of the Copenhagen Metro in 2001

Also called	April Fool's Day
Type	Cultural, Western
Significance	Practical jokes, pranks
Observances	Comedy
Date	April 1
Next time	1 April 2022
Frequency	Annual

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages

Motivation for Homotopy Type Theory

People study type theory for many reasons. I'll highlight two:

- Its intrinsic homotopical/topological content. Things we prove in type theory are true in *any* ∞-topos.
- Its suitability for computer formalization.

This talk will focus on the first point.

So, what is an ∞-topos?

∞-toposes

An ∞-category has objects, (1-)morphisms between objects, 2-morphisms between 1-morphisms, etc., with all k-morphisms invertible for $k > 1$.

Alternatively: a category enriched over spaces.

The prototypical example is the ∞-category of spaces.

An ∞-topos is an ∞-category $\mathcal{C}$ that shares many properties of the ∞-category of spaces:

- $\mathcal{C}$ is presentable. In particular, $\mathcal{C}$ is cocomplete.
- $\mathcal{C}$ satisfies descent. In particular, $\mathcal{C}$ is locally cartesian closed.

Examples include Spaces, slice categories such as Spaces$/S^1$, and presheaves of Spaces, such as G-Spaces for a group G.

If $\mathcal{C}$ is an ∞-topos, so is every slice category $\mathcal{C}/X$.

History of (Homotopy) Type Theory

Dependent type theory was introduced in the 1970's by Per Martin-Löf, building on work of Russell, Church and others.

In 2006, Awodey, Warren, and Voevodsky discovered that dependent type theory has homotopical models, extending 1998 work of Hofmann and Streicher.

At around this time, Voevodsky discovered his univalance axiom. And in 2011, higher inductive types were introduced by Bauer, Lumsdaine, Shulman, and Warren.

Homotopy type theory is type theory augmented with these principles.

Recently, it was shown using work of many people (Shulman, Lumsdaine, Kapulkin, de Boer, Brunerie, Anel, Biedermann, Finster, Joyal, etc.) that homotopy type theory has models in all ∞-toposes.

Background on Type Theory

First order logic can be used to study many theories: the theory of groups, Peano arithmetic, set theory (e.g., ZFC), etc.

In contrast, type theory is *not* a general framework for studying axiomatic systems, but instead unifies set theory and logic so that they live at the same level. (More on this later.)

In type theory, the basic objects are called types.
The notation $x : A$ means that x is an element of the type A.

Initially, types were thought of as sets, but in this talk we will think of them as spaces or as objects in an ∞-topos.

Background on Type Theory II

Like first order logic, type theory is a syntactic theory in which certain expressions are well-formed, and there are inference rules that tell you how to produce new expressions (i.e., theorems) from existing expressions.

$$\frac{A \qquad A \implies B}{B}$$

First order logic

$$\frac{a : A \qquad f : A \to B}{f(a) : B}$$

Type theory

There are also rules for introducing new *types* from existing types. These are called type constructors and correspond to common constructions in mathematics (and to the rules of logic).

Examples include function types, coproducts, products, the natural numbers, etc. We'll discuss these in more detail now.

Type Constructors: Function types

For any two types A and B, there is a function type denoted $A \to B$. In an ∞-topos, this is the internal hom B^A.

If $f(a)$ is an expression of type B whenever a is of type A, then we get a function $A \to B$ sending a to $f(a)$.

Examples:

- The identity function $\mathtt{id}_A$ sends a in A to a.
- For fixed $b : B$, there is a constant function sending everything in A to b.
- There is a function of type

$$(A \to B) \to ((B \to C) \to (A \to C))$$

  defined by

$$f \quad \mapsto \quad (\quad g \quad \mapsto (a \mapsto g(f(a)))).$$

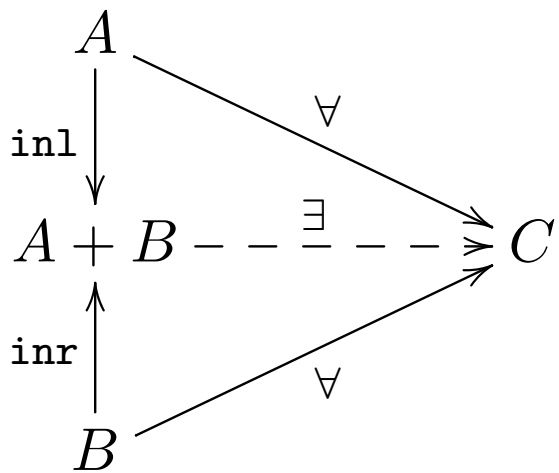
  This is composition of functions.

Type Constructors: Coproduct

Most constructions in type theory are defined inductively.

Example: given types A and B, there is a coproduct type $A + B$ which is generated by elements of the form `inl` a and `inr` b.

"Generated" means that it satisfies a weak universal property:



This corresponds to the categorical coproduct in an ∞-topos.

Type Constructors: $\varnothing$, 1, $\times$, $\mathbb{N}$

Here are other types defined by such induction principles:

- The empty type $\varnothing$ is a weakly initial object ("free on no generators"): for any C, there is a map $\varnothing \to C$.
- The one point type 1 is "free on one generator $*$": given $c : C$, there is a map $f : 1 \to C$ with $f(*) = c$.
- The product $A \times B$ of two types is generated by all pairs (a, b): given $g : A \to (B \to C)$, we get $f : A \times B \to C$ with $f(a, b) = g(a)(b)$.
- The type of natural numbers $\mathbb{N}$ is generated by $0 : \mathbb{N}$ and $\mathtt{succ} : \mathbb{N} \to \mathbb{N}$: given $c_0 : C$ and $c_s : \mathbb{N} \times C \to C$, we get $f : \mathbb{N} \to C$ with $f(0) = c_0$ and $f(\mathtt{succ}\ n) = c_s(n, f(n))$.

Note the preference for constructions defined by mapping out.

When the induction principles are generalized to dependent types, uniqueness will follow.

Dependent Types

We assume given a universe type $\mathtt{U}$ closed under the type forming operations, and therefore can write $X : \mathtt{U}$ to indicate that X is a (small) type.

The above structure is enough to construct types that depend on elements of other types.

Examples:

$$a \mapsto B : A \longrightarrow \mathtt{U} \quad \text{(a constant type family)}$$

$$n \mapsto B^n : \mathbb{N} \longrightarrow \mathtt{U} \quad (B^{n+1} := B \times B^n, \text{ inductively})$$

$$(A, B) \mapsto A + B : \mathtt{U} \times \mathtt{U} \longrightarrow \mathtt{U}$$

$$\mathtt{parity} : \mathbb{N} \longrightarrow \mathtt{U}$$

with $\mathtt{parity}(n) = \varnothing$ for n even and $\mathtt{1}$ for n odd.

These dependent types are one of the key ideas in type theory. A dependent type $A \to \mathtt{U}$ corresponds to a map $E \to A$ in an ∞-topos, and so slice categories are a crucial tool.

Dependent Sums and Products

Given a type family $B : A \to \mathtt{U}$, the dependent sum $\sum_{a:A} B(a)$ is freely generated by pairs (a, b) with $b : B(a)$.

The dependent sum has a projection map

$$\mathtt{pr}_1 : \sum_{a:A} B(a) \longrightarrow A$$

sending (a, b) to a.

In an ∞-topos, the dependent sum of the family $E \to A$ is just E.

There is also a dependent product $\prod_{a:A} B(a)$. Its elements are functions f sending each $a : A$ to an $f(a) : B(a)$.

In an ∞-topos, the dependent product of a family $E \to A$ is given by applying the right adjoint to pullback along $A \to 1$.

Propositions as Types: Curry-Howard

A type can be thought of as a proposition, which is true when inhabited:

Types	$\longleftrightarrow$	Propositions
$\varnothing$	$\longleftrightarrow$	false
$\mathtt{1}$	$\longleftrightarrow$	true
$P \times Q$	$\longleftrightarrow$	P and Q
$P + Q$	$\longleftrightarrow$	P or Q
$P \to Q$	$\longleftrightarrow$	P implies Q
$\prod_{x:A} P(x)$	$\longleftrightarrow$	$\forall x P(x)$
$\sum_{x:A} P(x)$	$\longleftrightarrow$	$\exists x P(x)$

Example Proof

As an example, how would we prove modus ponens:

$$(A \text{ and } (A \implies B)) \implies B?$$

In type theory, this proposition is represented by the type

$$(A \times (A \longrightarrow B)) \longrightarrow B.$$

We prove it by giving an element. By the inductive definition of the product, it's enough to give an element of B for each pair (a, f) in $A \times (A \to B)$. We simply give $f(a)$:

$$(a, f) \mapsto f(a)$$

Put another way, modus ponens and the evaluation map are the same thing in type theory.

More complicated theorems have more complicated proofs!

We've talked about many propositions, but what about: $a = b$?

Identity Types

Given a type A, the identity type of A is a type family $A \times A \to \mathtt{U}$ whose values are written $a = b$ for $a, b : A$.

This type family is inductively generated by "reflexivity" elements of the form $\mathtt{refl}_a : a = a$ for each $a : A$.

An element p of type $a = b$ can be thought of as a proof that a equals b.

In an ∞-topos, this type family is represented by the diagonal map $A \to A \times A$.

Therefore, two elements $a, a' : A$ are equal if the corresponding maps $1 \to A$ are homotopic.

Using the Identity Type

It was a remarkable insight of Martin-Löf that equality can be defined by induction! Many properties follow immediately.

Symmetry: $(a = b) \to (b = a)$.

Proof. To define a function from the type family $a = b$ to another type family, it's enough to define it on $\texttt{refl}_a : a = a$.
In this case, the target is also $a = a$, so we send $\texttt{refl}_a$ to $\texttt{refl}_a$. □

One can similarly prove:

Functions respect equality: For $f : A \to B$, $(a = b) \to (f(a) = f(b))$.

Doing Mathematics

With the foundation presented so far, all of the usual constructions of mathematics can be done, with types thought of as sets.

For example, one can construct the real numbers and do analysis; one can prove theorems in algebra; and one can define topological spaces and simplicial sets, and prove the standard results about them.

Major results include:

- The four-colour theorem (Gonthier).
- The Feit-Thompson odd-order theorem (Gonthier et al).
- Kepler's sphere packing conjecture (Hales et al).
- A C compiler that has been proven correct and is used in industry (Leroy et al).
- And lots more.

Equivalences

Let's continue to think of a type as a homotopical object.

We say that $f : A \to B$ is an equivalence if it has left and right inverses. That is,

$$\texttt{IsEquiv}f := \left(\sum_{g:B\to A} (gf = \texttt{id}_A) \right) \times \left(\sum_{h:B\to A} (fh = \texttt{id}_B) \right).$$

The type of equivalences from A to B is

$$(A \simeq B) := \sum_{f:A\to B} \texttt{IsEquiv}f.$$

Do ∞-toposes satisfy any properties that the set theoretic interpretation does not satisfy?

The Univalence Axiom

For types A and B in $\mathtt{U}$, we define a function

$$\omega : (A = B) \to (A \simeq B)$$

by sending $\mathtt{refl}_A$ to $\mathtt{id}_A$.

The Univalence Axiom says that ω is an equivalence for all A and B.

This is an assertion about the universe $\mathtt{U}$, and it does not hold in the standard set-theoretic model.

Voevodsky showed that simplicial sets has a univalent universe. Recent work of Shulman (building on work of others) shows that this is true in any ∞-topos.

If ω is an equivalence, then there is an inverse map

$$(A \simeq B) \longrightarrow (A = B)$$

which implies that equivalent types are equal.

Higher Inductive Types

Also motivated by the simplicial set model, Bauer, Lumsdaine, Shulman, and Warren introduced higher inductive types in 2011.

In an inductive type (like $A + B$, $\mathbb{N}$, $\sum_a B(a)$, $a = b$, etc.), we freely throw in elements of a type.

In a higher inductive type (HIT), we are also allowed to freely throw in paths, paths between paths, etc.

Example: The circle S^1 is the HIT generated by an element $\texttt{base} : S^1$ as well as a path $\texttt{loop} : \texttt{base} = \texttt{base}$.

It was shown by Lumsdaine and Shulman that a large class of HITs can be modelled in any ∞-topos.

For example, S^1 corresponds to the suspension of $1 + 1$, defined as the usual homotopy pushout.

Homotopy groups

Let the type X have a basepoint x_0. We define the loop space of X at x_0 to be the type

$$\Omega X \;:=\; (x_0 = x_0).$$

In an ∞-topos, this corresponds to the homotopy pullback

$$\begin{array}{ccc} \Omega X & \longrightarrow & X \\ \downarrow & & \downarrow{\scriptstyle \Delta} \\ 1 & \xrightarrow{(x_0,x_0)} & X \times X \end{array}$$

Then, for $n : \mathbb{N}$, we can define the nth homotopy group to be

$$\pi_n(X, x_0) \;:=\; \pi_0(\Omega^n X),$$

where π_0 is defined as a certain HIT.

As usual, one can prove that this is a group for $n \geq 1$ and is abelian for $n \geq 2$.

Consequences of Univalence and HITS

Assuming Univalence, one can prove $\pi_1(S^1) = \mathbb{Z}$, $\pi_3(S^2) = \mathbb{Z}$, $\pi_4(S^2) = \mathbb{Z}/2$, the Freudenthal suspension theorem, the Blakers-Massey Theorem, and many other results.

The proofs of these results in type theory imply them for spaces (without having to even define "space").

But the same proofs imply these results in all ∞-toposes, so the theorems are much more general.

The price we pay for this generality is that we need to make purely homotopical arguments, and we can't use the law of excluded middle, the axiom of choice, or Whitehead's theorem, since these aren't true in every ∞-topos.

This means that in some cases new proofs are needed.

More results

Here is a summary of some of my work in HoTT:

- The Hurewicz theorem: For X $(n-1)$-connected, $\pi_n(X)^{ab} \cong H_n(X)$. (Joint with L. Scoccola).
- One can localize a type X at a prime p. For X simply connected and $n \geq 1$, $\pi_n(X_{(p)})$ is the algebraic localization of $\pi_n(X)$. (Joint with Opie, Rijke, Scoccola.)
- Given a reflection L onto a subuniverse of `U` (the analog of a reflective subcategory), there is a reflection L' onto the subuniverse of types with L-local identity types. (Joint with Opie, Rijke, Scoccola.)
- Non-accessible localizations: A general technique for producing a reflective subuniverse from a "large" amount of data, extending work of Casacuberta, Scevenels, Smith for spaces.

Thanks!