An Introduction to Homotopy Type Theory

Dan Christensen
University of Western Ontario

University of Waterloo, March, 2023

Outline:

- Motivation
- Introduction to type theory
- Homotopy type theory

Motivation for Homotopy Type Theory

People study type theory for many reasons.

- It gives a foundation that is in many ways closer to mathematical practice.
- Results proved in type theory are not only true for sets, but are true in many other settings as well, such as sheaves of sets over a space.
- Type theory has an intrinsic homotopical/topological content, allowing us to prove things about spaces, sheaves of spaces, and other homotopical settings.
- Proofs in type theory are amenable to computer formalization.

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.

An Introduction to 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 we will see later that it is fruitful to think of them as being like spaces.

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 $A \to B$.

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

Conversely, if $f : A \to B$ and $a : A$, then $f(a) : B$.

Finally, $(\lambda a.f(a))(a')$ reduces to $f(a')$.

Examples:

- The identity function $\mathtt{id}_A$ is defined to be $\lambda a.a$.
- The constant function sending everything in A to $b : B$ is $\lambda a.b$.
- Given functions $f : A \to B$ and $g : B \to C$, their composite $gf : A \to C$ is $\lambda a.g(f(a))$.
- And composition $\lambda f.\lambda g.\lambda a.g(f(a))$ has type

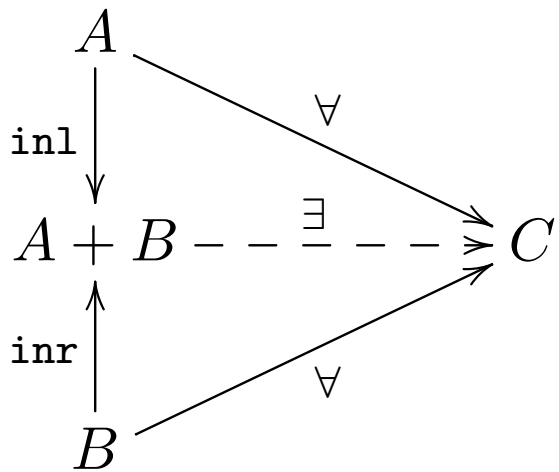
$$(A \to B) \to ((B \to C) \to (A \to C)).$$

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 disjoint union in set theory.

It turns out that using ingredients we'll discuss later, one can prove uniqueness.

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

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 $\mathtt{1}$ is "free on one generator $*$": given $c : C$, there is a map $f : \mathtt{1} \to C$ 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 $f(\mathtt{succ}\ n) = c_s(n, f(n))$.

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:

$$\begin{aligned} \lambda b.A : B \longrightarrow \mathtt{U} \quad & \text{(a constant type family)} \\ \lambda n.B^n : \mathbb{N} \longrightarrow \mathtt{U} \quad & (B^{n+1} := B \times B^n, \text{ inductively}) \\ \lambda(A,B).\, A + B : \mathtt{U} \times \mathtt{U} \longrightarrow \mathtt{U} & \\ \mathtt{parity} : \mathbb{N} \longrightarrow \mathtt{U} & \end{aligned}$$

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.

Dependent Sums and Products

Dependent sums are like the disjoint union:

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.

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

Note: both the value of $f(a)$ and the type of $f(a)$ depend on a.

Propositions as Types: Curry-Howard

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

Types		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

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

and 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 $f(a)$:

$$\lambda(a, f).f(a)$$

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.

The associated map $\sum_{a,b:A} (a = b) \longrightarrow A \times A$ can be thought of as the diagonal map $A \to A \times A$.

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 prove the standard results about them.

The following major results have been formalized in a proof assistant:

- The four-colour theorem (Gonthier, 2005).
- CompCert, a C compiler that has been proven correct and is used in industry (Leroy et al, 2008).
- The Feit-Thompson odd-order theorem (Gonthier et al, 2012).
- Kepler's sphere packing conjecture (Hales et al, 2014).
- The Liquid Tensor Experiment (Commelin et al, 2022).
- And lots more.

Homotopy type theory

Many categories are models of type theory, and the theorems we prove hold in all of these models.

Spaces form a model, where $a = b$ is interpreted as the space of paths from a to b! We next discuss a special property of homotopical models.

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

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

The type of equivalences from A to B is

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

Set-theoretically, we should think of these as bijections, but for spaces these are the homotopy equivalences.

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 $\texttt{loop} : \texttt{base} = \texttt{base}$.

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

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

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 models, 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 or the axiom of choice, 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 theory of Ext-groups with the usual long exact sequences. Note: in models, can have $\mathrm{Ext}^2_{\mathbb{Z}}(A, B) \neq 0$! (Joint with Flaten).
- The Hurewicz theorem: For X $(n-1)$-connected, $\pi_n(X)^{ab} \cong H_n(X)$. (Joint with Scoccola).
- Results about H-spaces, which are new even for spaces. For example, we describe how to deloop a *central* H-space. (Joint with Buchholtz, Flaten, Rijke.)
- 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.)
- 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!