An Introduction to Homotopy Type Theory

Dan Christensen
University of Western Ontario

Kyoto University, March, 2020

Outline:

- Background on homotopy type theory
- Identity types and univalence
- Higher inductive types
- Localization in homotopy type theory

Motivation for 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.

I will say more about these after introducing type theory.

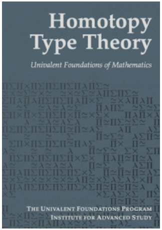
History of (Homotopy) Type Theory

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. Homotopy type theory is type theory augmented with these principles.

2012–2013: A special year at the IAS, which led to The HoTT book.



Since then, the field has been developing rapidly!

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 $a : A$ means that a 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. (Or even as objects in an ∞-category.)

Background on Type Theory II

As in 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, $(\lambda a.f(a))(b)$ reduces to $f(b)$.

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

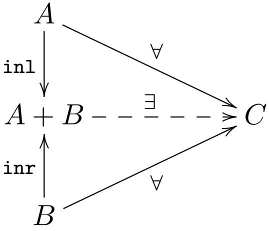
$$(A \to B) \to ((B \to C) \to (A \to C)).$$

Type Constructors: Coproduct

Most constructions in type theory are defined inductively.

For example, given types A and B, there is another 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 $f(\mathtt{succ}\ n) = c_s(n, f(n))$.

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 $\mathtt{Type}$, and therefore can write $X : \mathtt{Type}$ to indicate that X is a type.

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

These dependent types are one of the key ideas in Martin-Löf type theory.

Examples:

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

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

Dependent Sums and Products

Dependent sums are like the disjoint union:

Given a type family $B : A \to \mathtt{Type}$, 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.

$\prod_{a:A} B(a)$ can also be thought of as the space of sections of $\mathtt{pr}_1$.

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

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

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 $A \times A \to \texttt{Type}$ whose values are written $a = b$ for $a, b : A$.

This type family is inductively generated by "reflexivity" elements of the form $\texttt{refl}_a : a = a$ 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, 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 $\texttt{refl}_a$. □

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

Proof. As above, by induction, we can assume that a and b are the same, and we have to give an element of $f(a) = f(a)$.
We give $\texttt{refl}_{f(a)}$. □

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.

Examples of Formalized Proofs

- The four-colour theorem (Gonthier, 2005).
  Traditional proof: 43 pages + computer calculations.
  Formal proof: 60,000 lines, several years' work.
- Kepler's sphere packing conjecture (Hales et al, 2003–2014).
  Original proof: 1998–2005, using computation.
  Annals of Math: referees 99% sure of correctness.
  Formal proof: 11 years, large team.
- The Feit-Thompson odd-order theorem (Gonthier et al, 2013).
  Original proof: 250 pages, roughly 9,000 lines.
  Formal proof: 40,000 lines plus 110,000 lines of background material. Six years, with a team.
- CompCert, a formally verified C compiler (Leroy et al, 2008-now).
  Used in industry for mission-critical software.
  Initially 42,000 lines and several years' work, but has grown.
- There are thousands of smaller projects.

There are Infinitely Many Primes (Lean)

```
theorem infinitude_of_primes (N : ℕ) : ∃ p ≥ N, prime p :=
begin
  let M := fact N + 1,
  let p := min_fac M,
  have pp : prime p :=
    min_fac_prime (ne_of_gt (succ_lt_succ (fact_pos N))),
  existsi p,
  split,
  {
    by_contradiction,
    simp at a,
    have h₁ : p ∣ M, apply min_fac_dvd,
    have h₂ : p ∣ fact N :=
      dvd_fact (prime.pos pp) (le_of_lt a),
    have h : p ∣ 1 := dvd_add_right h₂ h₁,
    exact prime.not_dvd_one pp h,
  },
  exact pp
end
```

via Scott Morrison

Models

A model of type theory is a category equipped with type constructors that satisfy all of the properties we have assumed.

$$\begin{aligned}
\varnothing &\quad\longleftrightarrow\quad \text{initial object}\\
\mathtt{1} &\quad\longleftrightarrow\quad \text{terminal object}\\
P \times Q &\quad\longleftrightarrow\quad \text{product}\\
P + Q &\quad\longleftrightarrow\quad \text{coproduct}\\
P \to Q &\quad\longleftrightarrow\quad \text{cartesian closed}\\
\prod_{x:A} P(x) &\quad\longleftrightarrow\quad \text{locally cartesian closed}\\
a = b &\quad\longleftrightarrow\quad \text{a weak factorization system}
\end{aligned}$$

with suitable compatibility. (Making this precise is technical.)

Motivating example: Set with epi-mono weak factorization system, so $a = b$ is usual equality.

Models, II

For $a, b : A$, we have a type $a = b$. Therefore, *it* has an associated identity type $p = q$ for $p, q : a = b$. For over 20 years, it was an open question whether $p = q$ always holds.

In 1998, Hofmann and Streicher showed that the category of groupoids is a model of type theory, with $a = b$ given by $\mathrm{Hom}(a, b)$. It follows that the answer is no!

Then in 2006, Voevodsky showed that simplicial sets form a model of type theory, which also shows that the answer is no.

In 2019, Shulman showed that every ∞-topos is a model of type theory. He did this by showing that many Quillen model categories are models of type theory.

Any proof in type theory gives a theorem in all models!

The Simplicial Model

(To make things approachable, I will write "space" below, but for technical reasons it is better to use simplicial sets.)

We interpret a type X as a topological space,
an element $x : X$ as a point of X,
and a type family $X \to \texttt{Type}$ as a fibration $Y \to X$.

The identity type $X \times X \to \texttt{Type}$ is interpreted as the path space fibration $X^I \to X \times X$.

Thus, an element $p : a = b$ is interpreted as a path from a to b in X.

And for $f, g : X \to Y$, $H : f = g$ is interpreted as a homotopy.

The induction principle for identity types still holds and gives trivial proofs of ordinary facts about paths.

For example, our proofs above show that every path has an inverse, and that functions take paths to paths.

Equivalences

The simplicial model suggests thinking of a type as a homotopical object. Let's see where this leads.

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

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

The type of equivalences from A to B is

$$A \simeq B :\equiv \sum_{f:A \to B} \mathtt{IsEquiv} f.$$

Does the simplicial model satisfy any properties that the set theoretic interpretation does not satisfy?

Univalence Axiom

For types A and B, 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 types A and B.

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

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

which implies that equivalent types are equal.

This is an assertion about the universe `Type`, and it does not hold in the standard set-theoretic model.

But it does hold for the model in simplicial sets and in fact in any ∞-topos.

Higher Inductive Types

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

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

In a higher inductive type (HIT), we are 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 $\texttt{loop} : \texttt{base} = \texttt{base}$.

The induction principle for the circle says that a map $S^1 \to Z$ corresponds to a choice of $z : Z$ as well as $p : z = z$.

Example: S^2 is the HIT generated by $\texttt{base} : S^1$ and $\texttt{surf} : \texttt{refl}_{\texttt{base}} = \texttt{refl}_{\texttt{base}}$.

Example: Regular cell complexes can be defined in a similar way.

Higher Inductive Types, II

The paths in a HIT can be parametrized:

Example: The suspension of a type X as the HIT ΣX generated by points $N, S : \Sigma X$ as well as $\texttt{merid} : X \to (N = S)$.

They can even be recursively parametrized:

Example: The set trunction or 0-truncation $\|X\|_0$ of a type X is the HIT generated by $\texttt{tr} : X \to \|X\|_0$ and

$$\prod_{x,y:\|X\|_0} \prod_{p,q:x=y} p = q.$$

This is also denoted $\pi_0(X)$.

Homotopy type theory is type theory augmented with the univalence axiom and higher inductive types.

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

$$\Omega X :\equiv x_0 = x_0.$$

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

$$\pi_n(X, x_0) :\equiv \|\Omega^n X\|_0.$$

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

The proofs in type theory of these results imply them for simplicial sets and therefore 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, the axiom of choice, or Whitehead's theorem.

I'll end by talking about one of my results that fits into this framework.

Localization in Algebra and Topology

In algebra, localization at a prime p allows one to study a problem one prime at a time.

So-called fracture theorems can then be used to combine the results for each prime and figure out the answer to the original question.

There is a similar technique in algebraic topology. Given a space X, there is an associated space L_pX called the localization of X at p.

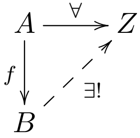
It has the property that for each n, $\pi_n(L_pX)$ is the algebraic localization of $\pi_n(X)$.

Studying such p-local spaces is easier than studying general spaces, and there are fracture theorems that can be used to reconstruct a space from its p-localizations.

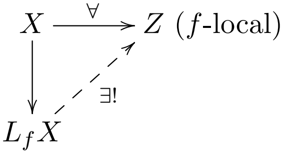
Localization in HoTT

Rijke-Shulman-Spitters (2017)

Given a map $f : A \to B$, we say that a type Z is f-local if every map $A \to Z$ extends uniquely to B:



A map $X \to L_f X$ is the f-localization of X if it is the initial map to an f-local type:



RSS show that such localizations always exist, and prove many properties.

Localization in HoTT II C-Opie-Rijke-Scoccola 1807.04155

We study the special case in which f is the degree p map $S^1 \to S^1$, and so f-localization amounts to localizing away from p. (By combining these, one can localize at a prime q.)

Theorem. For a simply connected type X, $\pi_n(L_f X)$ is the algebraic localization of $\pi_n(X)$ away from p.

Theorem. For a simply connected type X, $L_f(\Omega X) \simeq \Omega(L_f X)$.

Scoccola has extended these results to nilpotent types and has also proved a fracture theorem in HoTT.

Along we way, we show that given any localization operation L, there is a new localization L' whose local types are the types with local loop spaces, and we show that $L(\Omega X) \simeq \Omega(L' X)$ for every X.

Learning More

To learn more about homotopy type theory:

These slides and a longer introduction to type theory are on my web site.

Mike Shulman's slides from two series of lectures are great.

Homotopy Type Theory: Univalent Foundations of Mathematics is the standard source.

The localization material is in:

J.D. Christensen, M. Opie, E. Rijke and L Scoccola. *Localization in homotopy type theory*, arXiv:1807.04155.

Thanks!