

An introduction to homotopy type theory

Dan Christensen
University of Western Ontario

Shantou University, November 1, 2017

Outline:

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

History of Type Theory

Initial ideas due to Bertrand Russell in early 1900's, to create a foundation for mathematics that avoids [Russell's paradox](#).

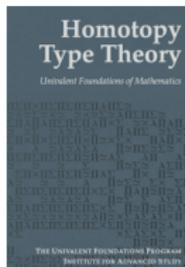
Studied by many logicians and later computer scientists, particularly Church, whose [\$\lambda\$ -calculus](#) (1930's and 40's) is a type theory.

In 1972, Per Martin-Löf extended type theory to include [dependent types](#). It is this form of type theory that we will focus on. One of the key features is that it [unifies set theory and logic!](#)

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

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

Type theory is a logical system in which 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**.

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

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

First order logic

$$\frac{a : A \quad f : A \rightarrow B}{f(a) : B}$$

Type theory

Background on Type Theory II

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 intrinsically a theory about sets/spaces. It 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.)

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

- Its intrinsic homotopical content. (This talk.)
- Its suitability for computer formalization. (Not this talk!)

In fact, type theory has models in many infinity categories, and a proof in type theory gives a theorem in all of these models!

Background on Type Theory III

We'll dive right in without being formal about it.

In type theory, each element $x : X$ has a **unique** type, so we can't directly talk about intersections, unions, etc. Instead, type theory comes with **type constructors** that 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 $A \rightarrow B$, which should be thought of as an internal hom B^A .

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

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

Examples:

- The **identity function** 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 $f : A \rightarrow B$ and $g : B \rightarrow C$, their **composite** $gf : A \rightarrow C$ is $\lambda a.g(f(a))$.
- And $\lambda f.\lambda g.\lambda a.g(f(a))$ has type

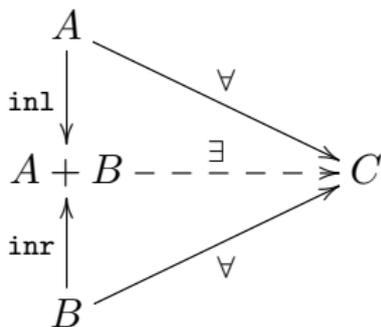
$$(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)).$$

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:



It turns out that using ingredients we'll discuss later, one can [prove](#) uniqueness.

Type Constructors: \emptyset , 1 , \times , \mathbb{N}

Here are other types defined by such induction principles:

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

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 **Type**, and therefore can write $X : \mathbf{Type}$ 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:

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

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

Dependent Sums and Products

Dependent sums are like the disjoint union:

Given a type family $B : A \rightarrow \mathbf{Type}$, 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**

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

sending (a, b) to a .

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

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

$\prod_{a:A} B(a)$ can also be thought of as the space of **sections** of pr_1 .

Propositions as Types: Curry-Howard

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

Types	\longleftrightarrow	Propositions
\emptyset	\longleftrightarrow	false
1	\longleftrightarrow	true
$P \times Q$	\longleftrightarrow	P and Q
$P + Q$	\longleftrightarrow	P or Q
$P \rightarrow Q$	\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 that $P \implies (Q \implies P)$?

In type theory, this proposition is represented by the type

$$P \rightarrow (Q \rightarrow P).$$

We prove it by giving an element, such as:

$$\lambda p. \lambda q. p$$

This sends p to the constant function sending q to p .

More complicated theorems of course have much more complicated proofs!

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

Identity Types

Given a type A , the **identity type** of A is a **type family** $A \times A \rightarrow \mathbf{Type}$ whose values are written $a = b$ for $a, b : A$.

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

An element p of the type $a = b$ was historically thought of as the assertion that a **equals** b , but in the homotopical interpretation p should be thought of as a **path** from a to b in A , with \mathbf{refl}_a being the constant path at a .

The associated map $\sum_{a,b:A} (a = b) \longrightarrow A \times A$ was historically thought of as the **diagonal** map $A \rightarrow A \times A$, but for our purposes it is better regarded as the **path fibration** $A^I \rightarrow A \times A$, which is obtained by replacing the diagonal map by a fibration.

Using the identity type

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

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

Proof. To define a function from the type family $a = b$ to another type, it's enough to define it on $\mathbf{refl}_a : a = a$.

In this case, the target is also $a = a$, so we send \mathbf{refl}_a to \mathbf{refl}_a . \square

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

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 $\mathbf{refl}_{f(a)}$. \square

In the homotopical interpretation, this says that functions take paths to paths.

Regular 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 [Feit-Thompson odd-order theorem](#) (Gonthier), the [four-colour theorem](#) (Gonthier), [Kepler's sphere packing conjecture](#) (Hales), a [C compiler](#) that has been proven correct and is used in industry (Leroy et al), and lots more.

Models

A **model** of type theory is a category equipped with type constructors that satisfy all of the properties we have assumed. (Making this precise is technical.)

\emptyset	\longleftrightarrow	initial object
1	\longleftrightarrow	terminal object
$P \times Q$	\longleftrightarrow	product
$P + Q$	\longleftrightarrow	coproduct
$P \rightarrow Q$	\longleftrightarrow	cartesian closed
$\prod_{x:A} P(x)$	\longleftrightarrow	locally cartesian closed
$a = b$	\longleftrightarrow	a weak factorization system

with suitable compatibility.

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 $\text{Hom}(a, b)$. 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.

At about the same time, Awodey and Warren showed that **weak factorization systems** give identity types.

We now know that many **Quillen model categories** are models of type theory. Homotopical thinking clarifies aspects of type theory.

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

Equivalences

The models above suggest thinking of a type as a homotopical object. Let's see where this leads.

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

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

The type of **equivalences** from A to B is

$$A \simeq B := \sum_{f:A \rightarrow B} \text{IsEquiv} f.$$

One can also define **loop spaces**, **homotopy groups**, etc.

Univalence Axiom

For types A and B , we define a function $\omega : (A = B) \rightarrow (A \simeq B)$ by sending refl_A to 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. Type theory with this axiom is called **Homotopy Type Theory**.

With this axiom, one can prove $\pi_1(S^1) = \mathbb{Z}$, $\pi_4(S^2) = \mathbb{Z}/2$, the Blakers-Massey Theorem, and many other results.

The Case of Simplicial Sets

In the simplicial model, the notion of equivalence matches the standard notion, and a type family corresponds to a [Kan fibration](#), obtained using the dependent sum construction.

The proofs in type theory of results like $\pi^1(S^1) = \mathbb{Z}$ imply these results for simplicial sets and therefore for spaces.

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

Localization in algebra and topology

In algebra, **localization at a prime p** is used to allow 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 $L_p X$ called the **localization of X at p** .

It has the property that for each n , $\pi_n(L_p X)$ 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.

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

$$\begin{array}{ccc} A & \xrightarrow{\forall} & Z \\ f \downarrow & \nearrow \exists! & \\ B & & \end{array}$$

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

$$\begin{array}{ccc} X & \xrightarrow{\forall} & Z \text{ (} f\text{-local)} \\ \downarrow & \nearrow \exists! & \\ L_f X & & \end{array}$$

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

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

For example, we prove that for simply connected X , $\pi_n(L_f X)$ is the algebraic localization of $\pi_n(X)$ away from p .

Along the way, we prove that for simply connected X , $L_f(\Omega X) \simeq \Omega(L_f X)$.

Scoccola has also proved a fracture theorem in HoTT.

Many of the proofs have been formalized in the proof assistant Coq.

A preprint will be available soon.

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.

Thanks!