The interaction between homotopy type theory and mathematics

Dan Christensen
University of Western Ontario

MCMP, June 2, 2019

Outline:

- Introduction to type theory
- The influence of type theory on mathematics
- The influence of mathematics on type theory

Motivation for Type Theory

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

- Its suitability for computer formalization.
- Its proofs are constructive and can be "run" on a computer.
- Its intrinsic homotopical/topological content.

I will say more about these after introducing type theory.

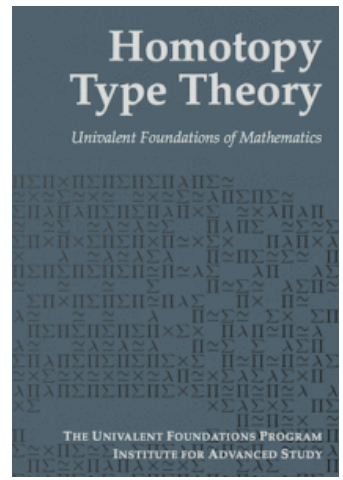
History of (Homotopy) Type Theory

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

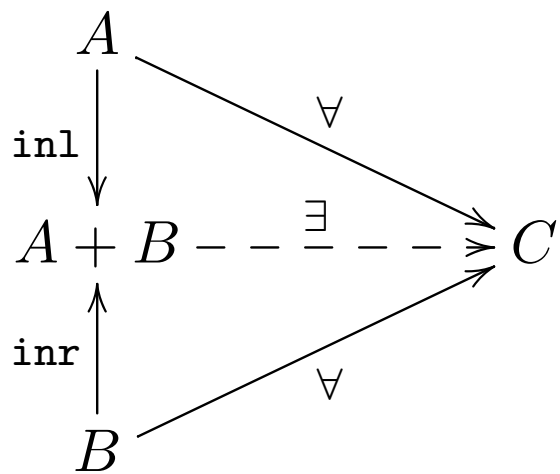
$$(A \to B) \to ((B \to C) \to (A \to C)).$$

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.

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.

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 $B : A \to \texttt{Type}$, the dependent sum $\sum_{a:A} B(a)$ is freely generated by pairs (a, b) with $b : B(a)$.

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	$\longleftrightarrow$	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 $f(a)$:

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

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.

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

That ends the background on type theory.

The Influence of Type Theory on Math, I

Computer-checked proofs

Type-theoretic proofs can be checked by a proof assistant such as Coq, Lean or Agda.

Type theory is better suited to this than first order logic.

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.

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

Consequences

- Improved reliability of the literature.
- Searching and indexing: When theorems are expressed in a formal language, we will be able to search based on *meaning*. E.g., Hales' Formal Abstracts project.
- Safety-critical infrastructure: The same formalization methods are in active use in engineering and industry.

The introduction of computer-checked proofs has lead to new collaborations between mathematicians, logicians, computer scientists and engineers.

Models

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

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

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

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

Models, II

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

Models

- Sets (traditional).
- Topological spaces (Voevodsky, 2006).
- Any ∞-topos (Shulman, 2019).

An ∞-category has objects, morphisms between objects, 2-morphisms between morphisms, and so on.

An ∞-topos is an ∞-category with extra properties that make it similar to topological spaces.

The fact that we can prove things in *any* ∞-topos using type theory is one of its most compelling aspects.

The Influence of Type Theory on Math, II

Any proof in type theory gives a theorem in any ∞-topos.

This relies on the fact that type theory uses constructive logic.

Most mathematicians aren't interested in constructive reasoning. But now we see that it's the price we pay to get more general results.

Consequences

- More mathematicians are now learning about and using constructive logic.
- Again, leads to new collaborations.
- Sometimes leads to new proofs of known results.
- Constructive proofs can by run. E.g., a proof that for any N there is a prime $p > N$ gives an algorithm for computing p.

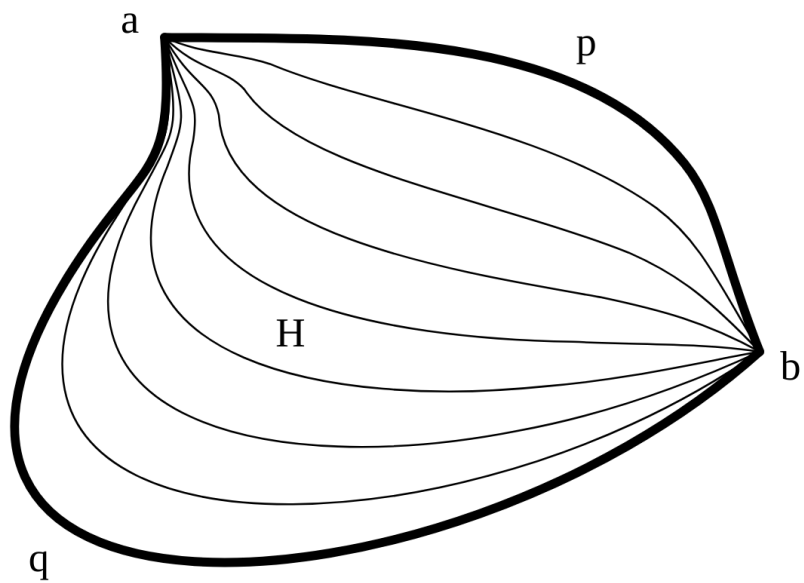
Note that proof assistants also support classical reasoning when required.

The Influence of Math on Type Theory, I

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!

The topological model and the models in ∞-toposes now make it clear that this higher structure is in fact the norm:



p, q : a = b H : p = q

What else can we learn from the new models?

Univalence Axiom

For types A and B, one can define the type $A \simeq B$ of equivalences/isomorphisms from A to B.

And we can 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!

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 models in ∞-toposes.

Type theory with this axiom is called Homotopy Type Theory.

The Influence of Math on Type Theory, II

Univalence is a powerful principle that can be used to prove many things.

It was motivated by topology, but has implications for logic in general, and is being embraced and studied by computer scientists and logicians.

It encodes the abstract concept that isomorphic objects are indistinguishable.

Practically speaking, it can simplify proofs, since it will automatically follow that isomorphic objects have the same properties.

Topologists also introduced higher inductive types, based on ideas from topology, and they also turn out to lead to practical improvements in type theory.

Summary

Interactions between type theory and mathematics:

- Computer checked proofs: Improved reliability of literature; semantic search; safety-critical infrastructure.
- Models in any ∞-topos: a precise formalism that allows us to prove results in any ∞-topos; increased interest in constructive logic; proofs give algorithms; new proofs of old results.
- Homotopical models clarify the properties of identity types.
- The Univalence Axiom is a general tool for simplifying reasoning.
- Higher Inductive Types also improve aspects of type theory.
- New collaborations between mathematicians, computer scientists, logicians and engineers.

These are the interactions I have chosen to focus on. There are many more.

Learning more

To learn more about homotopy type theory:

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

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

Mike Shulman, Homotopy type theory: the logic of space, `https://arxiv.org/pdf/1703.03007.pdf`

Mike Shulman, Homotopy Type Theory: A synthetic approach to higher equalities, `https://arxiv.org/pdf/1601.05035.pdf`

Thanks!