What can we learn from formalizations in homotopy type theory?

Floris van Doorn

Paris-Saclay University in Orsay

25 May 2023

Formalized homotopy theory

What homotopy theory has been formalized?

- The fundamental group: Mizar (2004), HOL Light, Isabelle, auto2, Lean, ...

Formalized homotopy theory

What homotopy theory has been formalized?

- The fundamental group: Mizar (2004), HOL Light, Isabelle, auto2, Lean, ...
- Homotopy groups: Lean (2022) — the fact that these are groups is in an open PR

Formalized homotopy theory

What homotopy theory has been formalized?

- The fundamental group: Mizar (2004), HOL Light, Isabelle, auto2, Lean, ...
- Homotopy groups: Lean (2022) — the fact that these are groups is in an open PR
- ~~Some~~ A lot results of in homological algebra

Formalized *synthetic* homotopy theory

In homotopy type theory, what results in homotopy theory have been formalized?

- Long exact sequence of homotopy groups

Formalized *synthetic* homotopy theory

In homotopy type theory, what results in homotopy theory have been formalized?

- Long exact sequence of homotopy groups
- Blaker's-Massey theorem

Formalized *synthetic* homotopy theory

In homotopy type theory, what results in homotopy theory have been formalized?

- Long exact sequence of homotopy groups
- Blaker's-Massey theorem
- computation of $\pi_4(S^3)$

Formalized *synthetic* homotopy theory

In homotopy type theory, what results in homotopy theory have been formalized?

- Long exact sequence of homotopy groups
- Blaker's-Massey theorem
- computation of $\pi_4(S^3)$
- Eilenberg-MacLane spaces

Formalized *synthetic* homotopy theory

In homotopy type theory, what results in homotopy theory have been formalized?

- Long exact sequence of homotopy groups
- Blaker's-Massey theorem
- computation of $\pi_4(S^3)$
- Eilenberg-MacLane spaces
- Seifert-van Kampen theorem

Formalized *synthetic* homotopy theory

In homotopy type theory, what results in homotopy theory have been formalized?

- Long exact sequence of homotopy groups
- Blaker's-Massey theorem
- computation of $\pi_4(S^3)$
- Eilenberg-MacLane spaces
- Seifert-van Kampen theorem
- homology and cohomology theories

Formalized *synthetic* homotopy theory

In homotopy type theory, what results in homotopy theory have been formalized?

- Long exact sequence of homotopy groups
- Blaker's-Massey theorem
- computation of $\pi_4(S^3)$
- Eilenberg-MacLane spaces
- Seifert-van Kampen theorem
- homology and cohomology theories
- Spectra, Omega-spectra

Formalized *synthetic* homotopy theory

In homotopy type theory, what results in homotopy theory have been formalized?

- Long exact sequence of homotopy groups
- Blaker's-Massey theorem
- computation of $\pi_4(S^3)$
- Eilenberg-MacLane spaces
- Seifert-van Kampen theorem
- homology and cohomology theories
- Spectra, Omega-spectra
- Serre and Atiyah-Hirzebruch spectral sequences

In this talk

- What is homotopy type theory (HoTT)?
- What has been done in homotopy type theory?
- What lessons can we learn from HoTT?

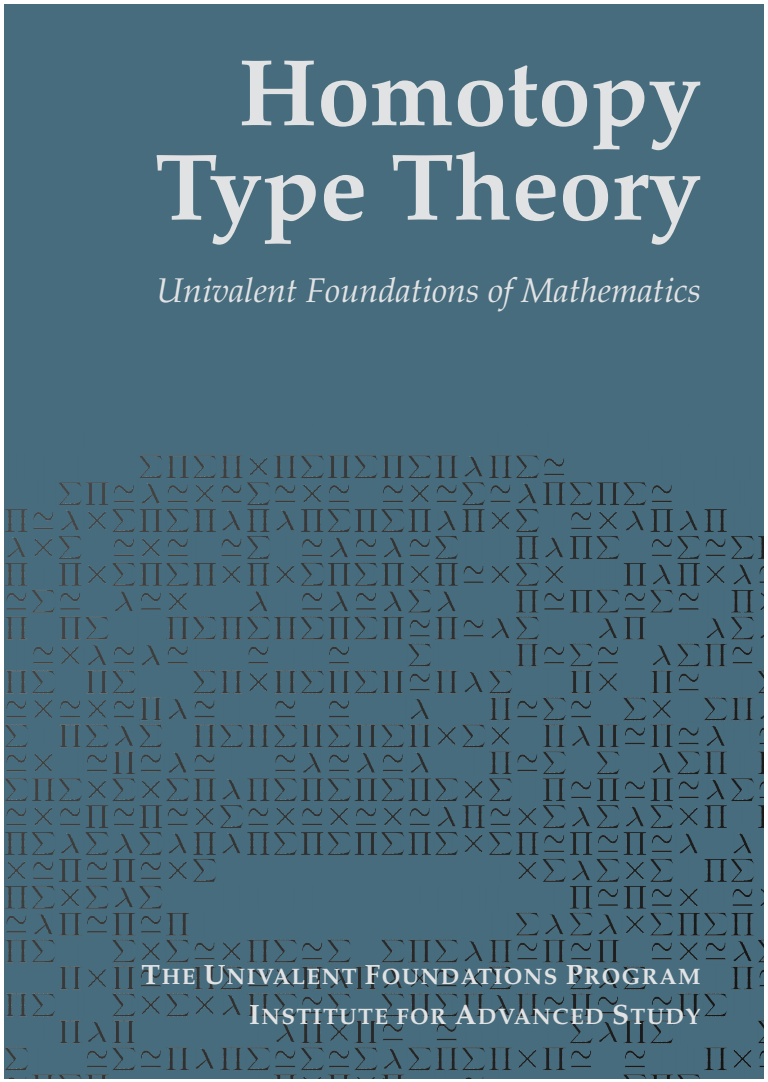
Not in this talk

I won't try to convince you to switch to HoTT.

Think of HoTT as a domain-specific language:
Good for doing homotopy theory, (potentially) less convenient for many other parts of mathematics.

This talk won't be a comprehensive introduction to HoTT.

Homotopy Type Theory



homotopytypetheory.org/book/

Homotopy Type Theory refers to the homotopical interpretation of dependent type theory by Voevodsky and Awodey–Warren.

Types are interpreted as topological spaces.

You can reason homotopically about these spaces. This is synthetic homotopy theory.

Homotopy Type Theory

HoTT = Lean

− choice

− proof irrelevance

+ univalence axiom

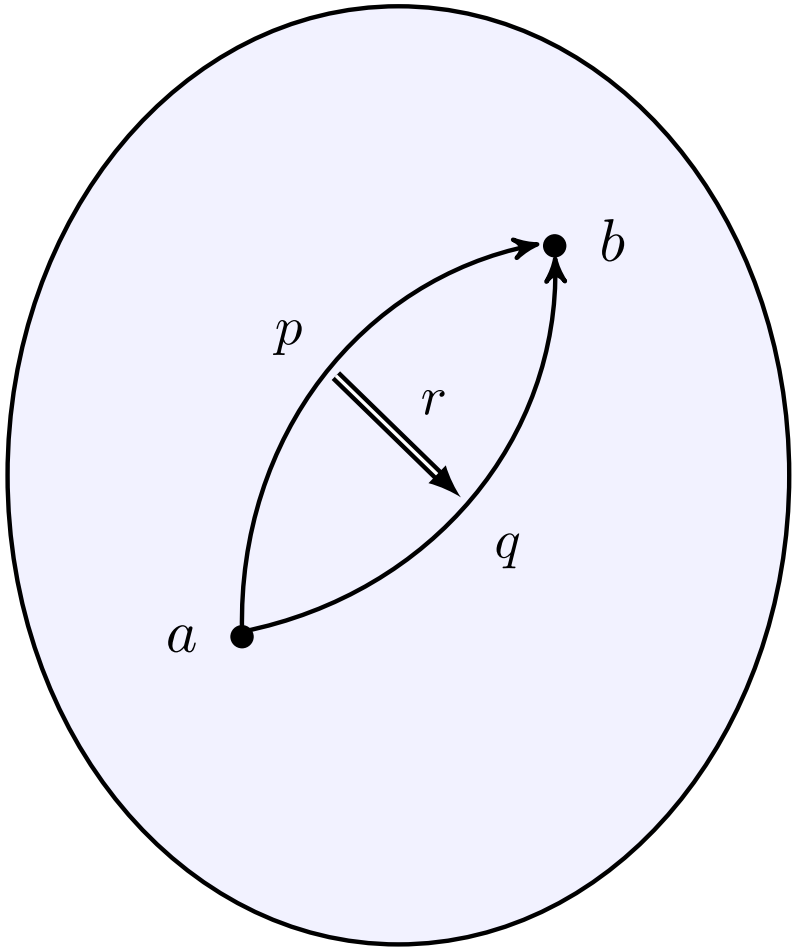
+ higher inductive types

Types as Spaces

A type A can have

- points $a, b : A$
- paths $p, q : \mathsf{Path}(a, b)$
- paths between paths $r : \mathsf{Path}(p, q)$

$\vdots$



Path Type

The path type (or identity type) is central in homotopy type theory.

It is defined in the same way in HoTT as equality in Lean.

```
inductive Path {X : Type*} (x : X) : X → Type*
| const : Path x x
```

Path Type

The path type (or identity type) is central in homotopy type theory.

It is defined in the same way in HoTT as equality in Lean.

```
inductive Path {X : Type*} (x : X) : X → Type*
| const : Path x x
```

We get the same induction principle for paths as you would get for equality:
Path induction: To prove something for $p : \mathsf{Path}(a, x)$ (where x is a variable not occuring in a), we may assume that x is a and that p is the constant path.

Everything in HoTT respects paths!

Path Type

The path type (or identity type) is central in homotopy type theory.

It is defined in the same way in HoTT as equality in Lean.

```
inductive Path {X : Type*} (x : X) : X → Type*
| const : Path x x
```

We get the same induction principle for paths as you would get for equality:
Path induction: To prove something for $p : \mathsf{Path}(a, x)$ (where x is a variable not occuring in a), we may assume that x is a and that p is the constant path.

Everything in HoTT respects paths!
In particular, functions respect paths, so are automatically "continuous".

Paths in various types

$A \times B$ is the product space. A path between pairs of points is a pair of paths.

$$\mathsf{Path}_{A\times B}((a,b),(a',b')) \quad \text{is} \quad \mathsf{Path}(a,a') \times \mathsf{Path}(b,b')$$

Paths in various types

$A \times B$ is the product space. A path between pairs of points is a pair of paths.

$$\mathsf{Path}_{A\times B}((a,b),(a',b')) \quad \text{is} \quad \mathsf{Path}(a,a') \times \mathsf{Path}(b,b')$$

$A \to B$ is a mapping space. A path between two functions is a homotopy (function extensionality):

$$\mathsf{Path}(f,g) \quad \text{is} \quad \Pi x,\ \mathsf{Path}(f(x),g(x))$$

Paths in various types

$A \times B$ is the product space. A path between pairs of points is a pair of paths.

$$\mathsf{Path}_{A\times B}((a,b),(a',b')) \quad \text{is} \quad \mathsf{Path}(a,a') \times \mathsf{Path}(b,b')$$

$A \to B$ is a mapping space. A path between two functions is a homotopy (function extensionality):

$$\mathsf{Path}(f,g) \quad \text{is} \quad \Pi x,\ \mathsf{Path}(f(x),g(x))$$

A paths between two types is an homotopy equivalence (Univalence axiom, Voevodsky):

$$\mathsf{Path}_{\mathrm{Type}}(A,B) \quad \text{is} \quad A \simeq B$$

Everything in HoTT respects homotopies and homotopy equivalences!

Path induction

```
variables {A : Type} {w x y z : A}
def concat (p : Path x y) (q : Path y z) : Path x z :=
by { induction q, exact p }
-- p · const := p

def concat_assoc
  (p : Path w x) (q : Path x y) (r : Path y z) :
  Path ((p · q) · r) (p · (q · r)) :=
by { induction r, reflexivity }
```

Homotopy groups

Given a pointed type (A, a_0). The type $\Omega A :\equiv (\mathsf{Path}(a_0, a_0), \mathsf{const}_{a_0})$ has

- a "multiplication" given by concatenation
- inverses given by path inverses
- identity given by the constant path

These operations satisfy the group laws.

Homotopy groups

Given a pointed type (A, a_0). The type $\Omega A :\equiv (\mathsf{Path}(a_0, a_0), \mathsf{const}_{a_0})$ has

- a "multiplication" given by concatenation
- inverses given by path inverses
- identity given by the constant path

These operations satisfy the group laws.

However, ΩA is not a set: it itself has nontrivial paths.

The set of connected components of ΩA is the fundamental group

$$\pi_1(A) := \|\Omega A\|_0.$$

Homotopy groups

Given a pointed type (A, a_0). The type $\Omega A :\equiv (\mathsf{Path}(a_0, a_0), \mathsf{const}_{a_0})$ has

- a "multiplication" given by concatenation
- inverses given by path inverses
- identity given by the constant path

These operations satisfy the group laws.

However, ΩA is not a set: it itself has nontrivial paths.

The set of connected components of ΩA is the fundamental group

$$\pi_1(A) := \|\Omega A\|_0.$$

The higher homotopy groups are $\pi_n(A) := \|\Omega^n A\|_0$.

Is HoTT cheating?

Many proofs in HoTT are very short, since they directly use the type theory for notions in homotopy theory.

Is HoTT cheating?

Many proofs in HoTT are very short, since they directly use the type theory for notions in homotopy theory.

HoTT is modeled by simplicial sets,

Anything proven in HoTT is then automatically true for simplicial sets, and therefore for all "nice enough" topological spaces.

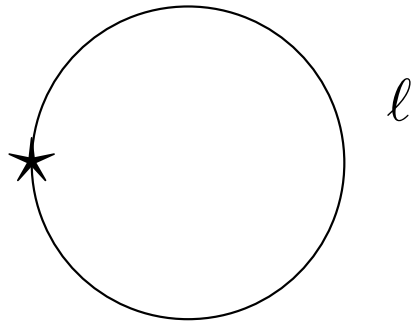
HoTT also has other models (any $(\infty, 1)$-topos), which means that the theorems can also interpreted in different settings.

Higher inductive types

```
inductive ℕ : Type
| 0 : ℕ
| S : ℕ → ℕ
```

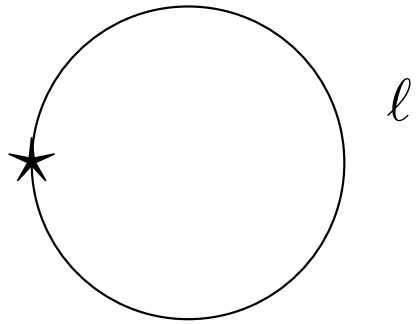
Higher inductive types

```
inductive ℕ : Type
| 0 : ℕ
| S : ℕ → ℕ

higher inductive 𝕊¹ : Type
| ⋆ : 𝕊¹
| ℓ : Path ⋆ ⋆
```



Higher inductive types

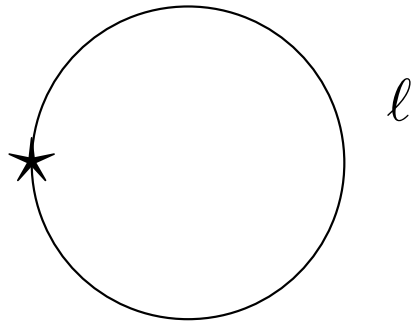
```
inductive ℕ : Type
| 0 : ℕ
| S : ℕ → ℕ

higher inductive 𝕊¹ : Type
| ⋆ : 𝕊¹
| ℓ : Path ⋆ ⋆
```



Recursion Principle. Given a loop $p : \mathsf{Path}_A(x, x)$ we can define a function $f : \mathbb{S}^1 \to A$ such that $f(\star) := x$ and $f(\ell) := p$.

Higher inductive types

```
inductive ℕ : Type
| 0 : ℕ
| S : ℕ → ℕ

higher inductive 𝕊¹ : Type
| ⋆ : 𝕊¹
| ℓ : Path ⋆ ⋆
```

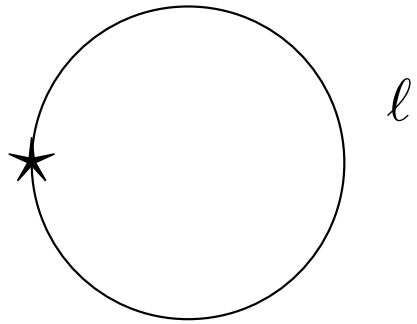


Recursion Principle. Given a loop $p : \mathsf{Path}_A(x, x)$ we can define a function $f : \mathbb{S}^1 \to A$ such that $f(\star) := x$ and $f(\ell) := p$.

Using the univalence axiom, we can prove $\pi_1(\mathbb{S}^1) = \mathbb{Z}$.

Higher inductive types

```
inductive ℕ : Type
| 0 : ℕ
| S : ℕ → ℕ

higher inductive 𝕊¹ : Type
| ⋆ : 𝕊¹
| ℓ : Path ⋆ ⋆
```



Recursion Principle. Given a loop $p : \mathsf{Path}_A(x, x)$ we can define a function $f : \mathbb{S}^1 \to A$ such that $f(\star) := x$ and $f(\ell) := p$.

Using the univalence axiom, we can prove $\pi_1(\mathbb{S}^1) = \mathbb{Z}$.

Higher inductive types allow you to define the suspension, smash product, homotopy colimits, Eilenberg-MacLane spaces, $\|-\|_0$, ...

In this talk

- What is homotopy type theory (HoTT)?
- What has been done in homotopy type theory?
- What lessons can we learn from HoTT?

Singular (co)homology

The singular (co)homology of X is defined as follows.

- Consider simplices in X: continuous $\sigma : \Delta^n \to X$;
- Take the free abelian group: $C_n(X) := \mathsf{Ab}(\Delta^n \to X)$;
- Define a boundary map $\partial_n : C_n(X) \to C_{n-1}(X)$ turning this into a chain complex
- Take the homology: $H_n(X) := \ker(\partial_n)/\mathsf{im}(\partial_{n+1})$
- For cohomology you dualize before taking homology.

Singular (co)homology

The singular (co)homology of X is defined as follows.

- Consider simplices in X: continuous $\sigma : \Delta^n \to X$;
- Take the free abelian group: $C_n(X) := \mathsf{Ab}(\Delta^n \to X)$;
- Define a boundary map $\partial_n : C_n(X) \to C_{n-1}(X)$ turning this into a chain complex
- Take the homology: $H_n(X) := \ker(\partial_n)/\mathsf{im}(\partial_{n+1})$
- For cohomology you dualize before taking homology.

Problem: this construction is not homotopy invariant. Up to homotopy, a simplex in X is just a point, and the boundary map is undefinable.

Singular (co)homology

The singular (co)homology of X is defined as follows.

- Consider simplices in X: continuous $\sigma : \Delta^n \to X$;
- Take the free abelian group: $C_n(X) := \mathsf{Ab}(\Delta^n \to X)$;
- Define a boundary map $\partial_n : C_n(X) \to C_{n-1}(X)$ turning this into a chain complex
- Take the homology: $H_n(X) := \ker(\partial_n)/\mathsf{im}(\partial_{n+1})$
- For cohomology you dualize before taking homology.

Problem: this construction is not homotopy invariant. Up to homotopy, a simplex in X is just a point, and the boundary map is undefinable.

Theorem. For a CW-complex X, the cohomology groups $H^n(X; A)$ are equivalent to homotopy classes of maps $[X, K(A, n)]$, where $K(A, n)$ is an *Eilenberg-Maclance space*.

(Co)homology in HoTT

In HoTT we can define the (reduced) cohomology of a pointed type X

$$\widetilde{H}^n(X, A) :\equiv \|X \to^* K(A, n)\|_0.$$

(Co)homology in HoTT

In HoTT we can define the (reduced) cohomology of a pointed type X

$$\widetilde{H}^n(X, A) :\equiv \|X \to^* K(A, n)\|_0.$$

We can define homology similarly, by using a smash product with the Eilenberg-MacLane spectrum.

Serre spectral sequence

Theorem (Serre Spectral Sequence)

Suppose B is a pointed simply connected space, $f : X \to B$ is a map with fiber F and A an abelian group.
Then there is a spectral sequence $E_r^{p,q}$ with

$$E_2^{p,q} = H^p(B; H^q(F; A)) \Rightarrow H^{p+q}(X; A).$$

We also proved a version for generalized cohomology (w.r.t. a spectrum) and parametrized cohomology (it is a dependent spectrum over the space).

Caveat: the fact that this respects the cup product was not formalized.

In this talk

- What is homotopy type theory (HoTT)?
- What has been done in homotopy type theory?
- What lessons can we learn from HoTT?

Reasoning techniques in HoTT

- HoTT gives you powerful reasoning techniques: path-induction, univalence, homotopy-invariance, higher inductive types and their induction principle, ...
- Naively translating these principles to Lean is not possible.

Reasoning techniques in HoTT

- HoTT gives you powerful reasoning techniques: path-induction, univalence, homotopy-invariance, higher inductive types and their induction principle, ...
- Naively translating these principles to Lean is not possible.
- Can we mimick these reasoning principles?
- For example: we can prove the recursion principle for the circle, and then use this abstractly.

Construct a model of HoTT

- If we construct a model of HoTT inside Lean, we can do proofs in this model and interpret them in topological spaces.

Construct a model of HoTT

- If we construct a model of HoTT inside Lean, we can do proofs in this model and interpret them in topological spaces.
- This is a lot of work, and would also require the HoTT proofs in an internal proof language in Lean.

Imitating definitions in HoTT

- In HoTT all definitions has to be homotopy invariant, which restricts how we can define or prove things.
- Some of these definitions are convenient to work with.

Imitating definitions in HoTT

- In HoTT all definitions has to be homotopy invariant, which restricts how we can define or prove things.
- Some of these definitions are convenient to work with.
- We can define $\Omega^n(X)$ by iterating Ω, and define $\pi_n(X)$ as its set of connected components.

Imitating definitions in HoTT

- In HoTT all definitions has to be homotopy invariant, which restricts how we can define or prove things.
- Some of these definitions are convenient to work with.
- We can define $\Omega^n(X)$ by iterating Ω, and define $\pi_n(X)$ as its set of connected components.
- We can *define* $H^n(X; A) := [X, K(A, n)]$.

Imitating definitions in HoTT

- In HoTT all definitions has to be homotopy invariant, which restricts how we can define or prove things.
- Some of these definitions are convenient to work with.
- We can define $\Omega^n(X)$ by iterating Ω, and define $\pi_n(X)$ as its set of connected components.
- We can *define* $H^n(X; A) := [X, K(A, n)]$.
- mathlib motto: First do something generally/abstractly and then apply to special cases.

Having proofs that are simpler than in HoTT

- In certain cases, formalizing the classical proof will be easier than the HoTT proof.
- Example: reasoning about homotopy pushouts.
- Example: reasoning about pointed maps, pointed homotopies and pointed higher homotopies.

Theorem (Atiyah-Hirzebruch spectral sequence)

If $X : \mathrm{Type}^*$ *and* $Y : X \to \mathrm{Spectrum}$ *a family of truncated spectra over* X*, then we get a spectral sequence with*

$$E_2^{p,q} = \tilde{H}^p(X; \lambda x.\pi_{-q}(Yx)) \Rightarrow \tilde{H}^{p+q}(X; \lambda x.Yx).$$

Theorem (Serre Spectral Sequence)

If $f : X \to B$ *is any map and* Y *is a truncated spectrum, then there is a spectral sequence with*

$$E_2^{p,q} = H^p(B, \lambda b.H^q(\mathit{fib}_f(b), Y)) \Rightarrow H^{p+q}(X, Y).$$