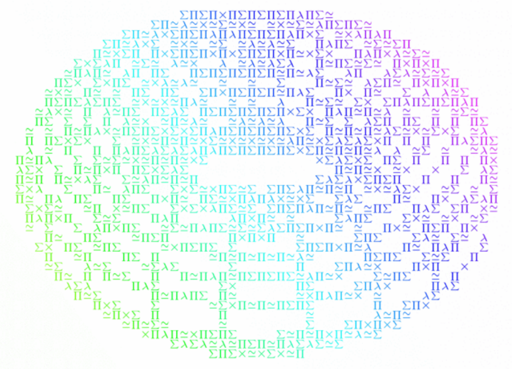
Homotopy type theory



Ingo Blechschmidt

November 25th, 2014

Outline

Homotopy type theory is a new branch of mathematics that combines aspects of several different fields in a surprising way. It is part of Voevodsky's *univalent foundations* program and based on a recently discovered connection between homotopy theory and type theory, a branch of mathematical logic and theoretical computer science.

In homotopy type theory, any set (really: *type*) behaves like a topological space, or more precisely, a homotopy type. The basic notion of equality is reimagined in an interesting way: Analogous to how two given points in a space may be joined by more than one path, two elements of a set can be equal in many ways. A new axiom, the *univalence axiom*, posits that equivalent structures really are the same, thus formalizing a widespread notational practice.

Besides explaining how working in homotopy type theory feels like, the talk will give answers to the listed questions. The talk does not assume any background in formal logic or type theory.

- What are logical foundations for mathematics and why should we care?
- What are the disadvantages of traditional set-based approaches to foundations?
- Why is the development of homotopy theory radically simplified in homotopy type theory?
- How are the seemingly diverse activities of *proving propositions* and *exhibiting constructions* identified?
- How do inductive definitions of important spaces concisely capture their homotopy-theoretic content?
- Why is homotopy type theory a major step towards practically useful and easily applicable proof assistants?

What are foundations?

- Foundations set the logical context for doing maths.
- Their details don't matter in everyday work (mostly).
- But their main concepts do.

http://collabcubed.com/2012/10/24/high-trestle-trail-bridge-rdg/

What are foundations?

- Foundations set the logical context for doing maths.
- Their details don't matter in everyday work (mostly).
- But their main concepts do.

- Classical foundations are *set-based* (ZF, ZFC, …):
  Everything is a set.
- $0 := \emptyset, \quad 1 := \{0\}, \quad 2 := \{0, 1\}, \quad \ldots$
- $(x, y) := \{\{x\}, \{x, y\}\}$ (Kuratowski pairing)
- $(x, y, z) := (x, (y, z))$
- maps: (X, Y, R) with $R \subseteq X \times Y$ such that …

- Foundations allow us to be maximally precise.
- A *proof* as commonly understood is really a shorthand for a (never spelled out) fully formal proof.
- Unlike informal proofs, the correctness of a formal proof can be checked mechanically.

Logicomix: An Epic Search for Truth

What's wrong with set-based foundations?

Set-based foundations …

- allow to formulate nonsensical questions,
- do not reflect typed mathematical practice,
- do not respect equivalence of structures,
- require complex encoding of "higher-level" subjects, complicating interactive proof environments.

- Examples for questions which can be formulated:
  - Is $2 = (0,0)$? (No, when using my definitions.)
  - Is $\sin \in \pi$? (Depends on your definitions.)
- In ordinary practice, these questions would be deemed as nonsensical, since they disrespect the *types* of mathematical objects and are not invariant under isomorphisms of the involved structures.
- Note: There are also *structural approaches* to set theory without a global membership predicate (e. g. ETCS), resolving this defect.

- Fully unravel the definition of “manifold” in set-theoretical language to get a grasp of the complex encodings needed.
- This is no problem for humans, but it is for machines.
- Voevodsky: “The roadblock that prevented generations of interested mathematicians and computer scientists from solving the problem of computer verification of mathematical reasoning was the unpreparedness of foundations of mathematics for the requirements of this task.”

- Note: Set theory is perfectly fine for studying *sets*.

What is homotopy type theory?

- Homotopy type theory is a new foundational theory.
- Basic notions have a homotopy-theoretic flavour.
- One can start doing “real mathematics” right away, without complex encodings.
- Initiated by Voevodsky in 2005.

Some participants of the IAS special year

- Homotopy type theory is approximately intensional Martin-Löf type theory (existing since the 1970s) plus the new *univalence axiom*.

- After repeatedly experiencing mistakes in his field going unnoticed for several years, Voevodsky wanted to work with proof assistants. He went public in 2009.

What are values and types?

- In type theory, there are **values** and **types**.
- Every value is of exactly one type.
- Types may depend on values.

$$
\begin{aligned}
7 &: \mathbb{N} \\
(3,5) &: \mathbb{N} \times \mathbb{N} \\
\mathsf{succ} &: \mathbb{N} \to \mathbb{N} \\
\text{zero vector} &: \mathbb{R}^n \quad (n : \mathbb{N})
\end{aligned}
$$

What are values and types?

- In type theory, there are **values** and **types**.
- Every value is of exactly one type.
- Types may depend on values.

$$
\begin{aligned}
7 &: \mathbb{N} \\
(3,5) &: \mathbb{N} \times \mathbb{N} \\
\mathsf{succ} &: \mathbb{N} \to \mathbb{N} \\
\text{zero vector} &: \mathbb{R}^n \quad (n : \mathbb{N})
\end{aligned}
$$

Let $B(x)$ be a type family depending on $x : A$.

- $\sum_{x:A} B(x) =$ "$\{(a,b) \mid a : A, b : B(a)\}$"
- $\prod_{x:A} B(x) =$ "$\{f : A \to ?? \mid f(a) : B(a)$ for all $a : A\}$"

- Types are familiar from programming (`Int`, `String`, ...).
- But the type systems of well-known mainstream languages are either trivial (Ruby, Python: everything is an object) or not very expressive (C, Java).
- Haskell and languages of the ML family have a rich type system, encompassing function types and algebraic data types.
- But even their type systems do not support *dependent types* – types which may depend on values. Look to Coq or Agda for those.

In the special case that $B(x) :\equiv B$ does not depend on x:

$$\sum_{x:A} B \equiv A \times B \qquad \prod_{x:A} B \equiv (A \to B)$$

What is the dependent equality type?

In set theory, for a set X and elements $x, y \in X$:

- "$x = y$" is a **proposition**.
- Set theory is **layered above** predicate logic.

In intens. type theory, for a type X and values $x, y : X$:

- There is the **equality type** $\mathrm{Id}_X(x, y)$ or $(x =_X y)$.
- To verify that "$x = y$", exhibit a value of $(x = y)$.
- Have $\mathsf{refl}_x : (x = x)$.
- Identity types may contain zero or **many** values!

Intuition: $(x = y)$ is the type of **proofs** that "$x = y$".

What is the dependent equality type?

In set theory, for a set X and elements $x, y \in X$:

- "$x = y$" is a **proposition**.
- Set theory is **layered above** predicate logic.

In intens. type theory, for a type X and values $x, y : X$:

- There is the **equality type** $\mathrm{Id}_X(x, y)$ or $(x =_X y)$.
- To verify that "$x = y$", exhibit a value of $(x = y)$.
- Have $\mathsf{refl}_x : (x = x)$.
- Identity types may contain zero or **many** values!

Intuition: $(x = y)$ is the type of **proofs** that "$x = y$".

Intuition: $(x = y)$ is the type of **paths** $x \rightsquigarrow y$.

- Note that we use logical terminology. A proposition is merely a statement, not necessarily a true statement.
- In an intensional type theory, propositions are not an extra part of the language, distinct from values and types.
- Instead, *propositions are (some) types*.
- To prove a proposition means to exhibit a value of it. Such a value can be thought of as a *proof* or *witness*.
- We have *proof relevance*.
- (Not all types are propositions, see below for IsProp.)

Examples for more complex propositions (types):

- "X is a subsingleton": $\prod_{x:X} \prod_{y:X} (x = y)$
- "Addition is commutative":
  $\prod_{n:\mathbb{N}} \prod_{m:\mathbb{N}} (n + m = m + n)$
- "Every number is even": $\prod_{n:\mathbb{N}} \sum_{m:\mathbb{N}} (n = 2m)$

How are types like spaces?

homotopy theory	type theory
space X	type X
point $x \in X$	value $x : X$
path $x \rightsquigarrow y$	value of $(x = y)$
(continuous) map	value of $X \to Y$

- A **homotopy** between maps $f, g : X \to Y$ is a value of

$$(f \simeq g) :\equiv \prod_{x:X} (f(x) = g(x)).$$

- A space X is **contractible** iff

$$\mathsf{IsContr}(X) :\equiv \sum_{x:X} \prod_{y:X} (x = y).$$

How are types like spaces?

- "The type X is **contractible**":

$$\mathsf{IsContr}(X) :\equiv \sum_{x:X} \prod_{y:X} (x = y).$$

- "The type X is a **mere proposition**":

$$\mathsf{IsProp}(X) :\equiv \prod_{x,y:X} (x = y)$$

- "The type X is a **set** or **discrete space**":

$$\mathsf{IsSet}(X) :\equiv \prod_{x,y:X} \mathsf{IsProp}(x = y)$$

- For instance, $\mathbb{N}$ is a set.

How are constructions encoded?

- The **fiber** of a map $f : X \to Y$ over a point $y : Y$ is

$$\mathsf{fib}_f(y) :\equiv \sum_{x:X} (f(x) = y).$$

- The **path space** of X is

$$X^I :\equiv \sum_{x,y:X} (x = y).$$

- The **based loop space** of X at x is

$$\Omega^1(X, x) :\equiv (x = x).$$

- The **path fibration** of (X, x) is the map

$$\mathsf{snd} : \sum_{y:X} (x = y) \to X.$$

How are constructions encoded?

- The **fiber** of a map $f : X \to Y$ over a point $y : Y$ is

$$\mathsf{fib}_f(y) :\equiv \sum_{x:X} (f(x) = y).$$

- The **path space** of X is

$$X^I :\equiv \sum_{x,y:X} (x = y).$$

- The **based loop space** of X at x is

$$\Omega^1(X, x) :\equiv (x = x).$$

- The **path fibration** of (X, x) is the map

$$\mathsf{snd} : \sum_{y:X} (x = y) \to X.$$

For doing homotopy theory in HoTT, the following are *not* needed:

- open sets
- construction of topologies on equivalence classes of paths
- real numbers
- axiom of choice
- law of excluded middle
- …

What are higher inductive definitions?

The type $\mathbb{N}$ of natural numbers is **freely generated** by

- a point $0 : \mathbb{N}$ and
- a function $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$.

This definition gives rise to an **induction principle**

$$\prod_{A:\mathbb{N}\to\mathcal{U}} \Big(A(0) \times \Big(\prod_{n:\mathbb{N}} A(n) \to A\big(\mathsf{succ}(n)\big) \Big) \longrightarrow \prod_{n:\mathbb{N}} A(n) \Big),$$

and a **recursion principle**

$$\prod_{X:\mathcal{U}} \Big(X \times \Big(\mathbb{N} \to (X \to X) \Big) \longrightarrow (\mathbb{N} \to X) \Big).$$

- $\mathcal{U}$ is a *universe*. Its values are types.
- The recursion principle is the specialization of the induction principle to constant type families $A(n) \equiv X$.
- In a *higher* inductive definition, constructors may not only generate *points*, but also *paths* and *higher paths*.
- We will drop the adjective "freely".

How to present famous spaces?

The **interval** I is generated by

- a point $0 : I$ and
- a point $1 : I$ and
- a path $\mathrm{seg} : (0 = 1)$.

Of course, we can show $I \simeq \mathbf{1}$.

How to present famous spaces?

The **interval** I is generated by

- a point $0 : I$ and
- a point $1 : I$ and
- a path $\mathsf{seg} : (0 = 1)$.

Of course, we can show $I \simeq \mathbf{1}$.

The **circle** S^1 is generated by

- a point $\mathsf{base} : S^1$ and
- a path $\mathsf{loop} : (\mathsf{base} = \mathsf{base})$.

How to present famous spaces?

The **torus** T^2 is generated by

- a point $b : T^2$,
- a path $p : (b = b)$,
- a path $q : (b = b)$, and
- a 2-path $t : (p \cdot q = q \cdot p)$.

How to present famous spaces?

The **suspension** ΣX of X is generated by

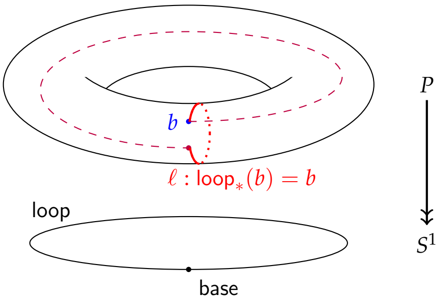
- a point $\mathsf{N} : \Sigma X$ and
- a point $\mathsf{S} : \Sigma X$ and
- a function $\mathsf{merid} : X \to (N = S)$.

The **cylinder** $\mathrm{Cyl}(X)$ of X is generated by

- a function $\mathsf{bot} : X \to \mathrm{Cyl}(X)$ and
- a function $\mathsf{top} : X \to \mathrm{Cyl}(X)$ and
- a function $\mathsf{seg} : \prod_{x:X}(\mathsf{bot}(x) = \mathsf{top}(x))$.

Of course, we can show $\mathrm{Cyl}(X) \simeq X \times I \simeq X$.

What is circle induction?



The **induction principle** of S^1 states: Given $P : S^1 \to \mathcal{U}$,

- a point $b : P(\mathsf{base})$, and
- a path $\ell : b =^P_{\mathsf{loop}} b$,

there is a function $f : \prod_{x:S^1} P(x)$ such that

- $f(\mathsf{base}) \equiv b$ and
- $f(\mathsf{loop}) = \ell$.

- In particular, restricting to constant type families, we obtain the recursion principle of S^1. It says that functions $S^1 \to X$ are given by a point $b : X$ and a loop $(b = b)$.

What is type truncation?

Let X be a type.

The **propositional truncation** $\|X\|_{-1}$ is generated by

- a function $X \to \|X\|_{-1}$ and
- for any $x, y : \|X\|_{-1}$, a path $x = y$.

The **0-truncation** $\|X\|_0$ is generated by

- a function $X \to \|X\|_0$ and
- for any $x, y : \|X\|_0$, $p, q : (x = y)$, a path $p = q$.

The **fundamental group** of (X, x_0) is

$$\pi_1(X, x_0) :\equiv \|\Omega^1(X, x_0)\|_0 :\equiv \|(x_0 = x_0)\|_0.$$

- Similarly, one can define the *n-truncation* of a type for any $n \geq -2$.
- $\|X\|_{-1}$ is a mere proposition, $\|X\|_0$ is a set (discrete space).
- More generally and precisely, $\|X\|_n$ is the reflection of X in the world of n-types, i. e. its n-th *Postnikov section*.
- $\|X\|_0$ is the set of connected components of X.

- By circle induction, an equivalent definition is

$$\pi_1(X, x_0) :\equiv \|(S^1, \mathsf{base}) \to (X, x_0)\|_0,$$

i. e. the set of connected components of the space of base-point-preserving functions $S^1 \to X$.

What is the univalence axiom?

An **equivalence** is a function $f : X \to Y$ such that

$$\mathsf{IsEquiv}(f) :\equiv \prod_{y:Y} \mathsf{IsContr}(\mathsf{fib}_f(y)).$$

Types X and Y are **equivalent** iff

$$(X \simeq Y) :\equiv \sum_{f:X\to Y} \mathsf{IsEquiv}(f).$$

The **univalence axiom** states: The canonical function

$$(X = Y) \longrightarrow (X \simeq Y)$$

is an equivalence, for all types X and Y.

- By the univalence axioms, equivalent types *really are* equal.
- It implies that isomorphic groups, vector spaces, ... are equal.
- Thus the widespread practice of *pretending* that isomorphic structures are equal is rigorously formalized.
- The univalence axiom guarantees that *any construction respects equivalence*.
- Most results require the univalence axiom.

- The univalence axiom implies *function extensionality*: The canonically defined function

$$(f = g) \longrightarrow \prod_{x:A}(f(x) = g(x))$$

  is an equivalence, for all functions $f, g : A \to B$.

- So homotopic functions are equal.

- Without the univalence axiom, it is consistent to assume *uniqueness of identity proofs*, i. e.

$$\mathsf{UIP} :\equiv \prod_{X:\mathcal{U}} \prod_{x,y:X} \prod_{p,q:(x=y)} (p = q),$$

  thus collapsing the homotopical universe.

- Phrased differently, the univalence axiom can not be added to an *extensional* type theory (one fulfillung UIP).

- *No computational interpretation of the univalence axiom is known yet.* This prevents us from *running* proofs (as computer programs). See below.

What's the status of the axiom of choice?

- The following proposition is **just true**, but is not a faithful rendition of the axiom of choice:

$$\Big(\prod_{x:A}\sum_{y:B} R(x,y)\Big) \longrightarrow \sum_{f:A\to B}\prod_{x:A} R(x,f(x)).$$

- The real axiom of choice,

$$\Big(\prod_{x:A}\Big\|\sum_{y:B} R(x,y)\Big\|_{-1}\Big) \longrightarrow \Big\|\sum_{f:A\to B}\prod_{x:A} R(x,f(x))\Big\|_{-1},$$

can be added as an axiom, but is rarely needed.

What's the status of the axiom of choice?

- The following proposition is **just true**, but is not a faithful rendition of the axiom of choice:

$$\Big(\prod_{x:A}\sum_{y:B} R(x,y)\Big) \longrightarrow \sum_{f:A\to B}\prod_{x:A} R(x,f(x)).$$

- The real axiom of choice,

$$\Big(\prod_{x:A}\Big\|\sum_{y:B} R(x,y)\Big\|_{-1}\Big) \longrightarrow \Big\|\sum_{f:A\to B}\prod_{x:A} R(x,f(x))\Big\|_{-1},$$

  can be added as an axiom, but is rarely needed.
- The law of excluded middle is too rarely needed.

$$\mathsf{LEM} :\equiv \prod_{A:\mathcal{U}}\Big(\mathsf{IsProp}(A) \to A + \neg A\Big).$$

- When doing homotopy theory in a classical set-based setting, one has to sometimes use the law of excluded middle or even the axiom of choice. This is an *artifact* of the chosen encoding in set theory. It is *not* due to an inherent unconstructivity of homotopy theory.

- Also recall that even in set-based mathematics, the law of excluded middle and the axiom of choice are not needed as often as it might first appear.

- Adding these two axioms prevents us from running proofs. In contrast to the univalence axiom, where it is believed that a computational interpretation might be found, this is less clear with these classical axioms.

- Since the law of excluded middle as stated refers only to mere propositions ((-1)-types), it is also denoted "LEM_{-1}".

- A law of excluded middle may not refer to *all* types, i. e.

$$\mathsf{LEM}_\infty :\equiv \prod_{A:\mathcal{U}} \Big(A + \neg A \Big),$$

  is inconsistent with the univalence axiom.

What are models of HoTT?

Conjecturally, HoTT can be interpreted in any **$(\infty, 1)$-topos**. Verified models include

- ∞Grpd, i. e. a model in simplicial sets, and
- $(\infty, 1)$-presheaf toposes over elegant Reedy categories.

Thus, any theorem proven in HoTT holds in the context of classical homotopy theory and in more general contexts.

The prototypical $(\infty,1)$-topos $\infty\mathrm{Grpd} \simeq \mathrm{Top}[\mathrm{whe}^{-1}] \simeq \mathrm{Kan}$ is equivalently:

- the $(\infty,1)$-category of all $(\infty,1)$-groupoids,
- the localization of the category of topological spaces (which have the homotopy type of a CW complex) at the class of weak homotopy equivalences, and
- the category of Kan complexes.

- Urs' example for a great success in formalization
- availability of higher structures
- built-in "functoriality"

References

- the book (XXX)
- Voevodsky on his motivations: `http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/2014_IAS.pdf`
- `http://www.math.ias.edu/~mshulman/hottseminar2012/01intro.pdf`