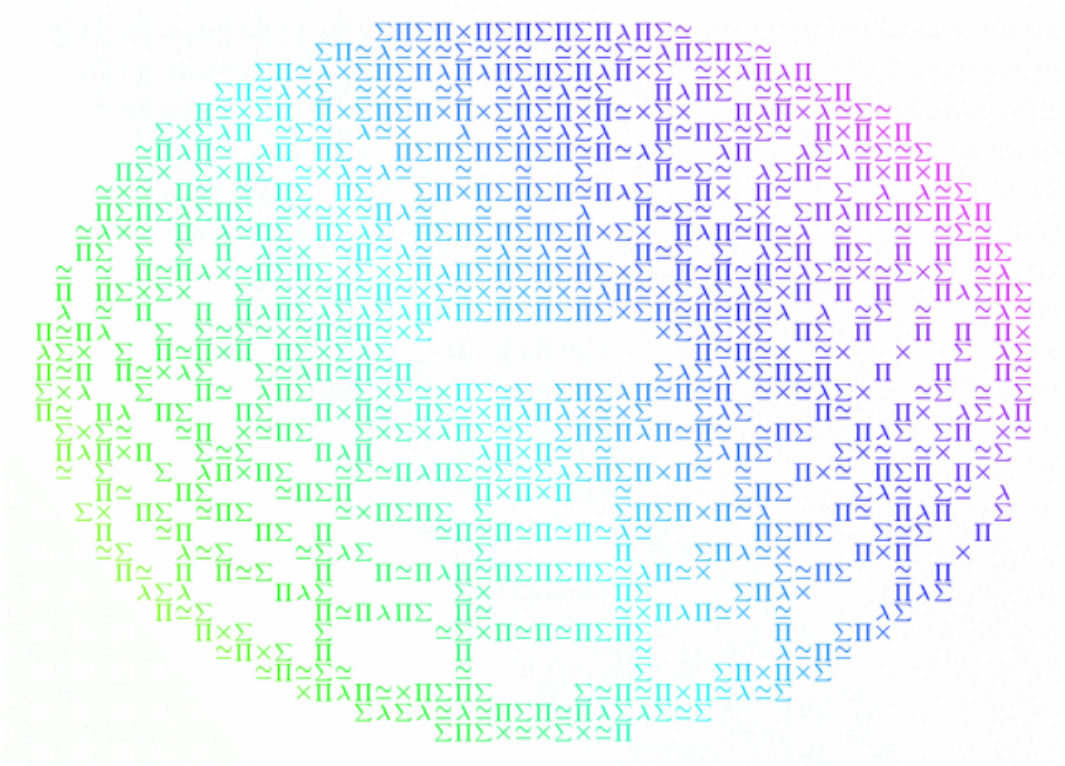
Homotopy type theory



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

- There is no such theorem as "the sun system is stable if and only if the following large cardinal axiom holds". Results depend only very occasionally on special foundational axioms.

- Bridges will continue to hold even if a logician discovers an inconsistency in Zermelo–Fraenkel set theory.

What's wrong with set-based foundations?

Set-based foundations ...

- 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

What is homotopy type theory?

Homotopy type theory …

- is elegant,
- reflects mathematical practice,
- contains wondrous new concepts,
- ensures that everything respects equivalences,
- simplifies the plumbing of homotopy theory,
- allows for accessible computer formalization.

- 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.

- Voevodsky: "This story got me scared. Starting from 1993 multiple groups of mathematicians studied the [...] paper at seminars and used it in their work and none of them noticed the mistake.

  And it clearly was not an accident. A technical argument by a trusted author, which is hard to check and looks similar to arguments known to be correct, is hardly ever checked in detail."

Results which have been fully formalized in HoTT include:

- $\pi_1(S^1)$
- $\pi_{k \leq n}(S^n)$
- $\pi_{n+1}(S^n)$ is cyclic for all $n \geq 3$
- fiber sequences and the long exact sequence
- the Hopf fibration
- the Freudenthal suspension theorem
- the van Kempen theorem
- the Blakers–Massey theorem

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.

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

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.

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

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

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

- 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, the *dependent sum* and the *dependent product* simplify to *cartesian product* and *function type*, respectively:

$$\sum_{x:A} B \equiv A \times B \quad \text{and} \quad \prod_{x:A} B \equiv (A \to B).$$

- Dependent sums and products are very common in mathematics, even if they are not explicitly referred to by name. For instance, a tangential vector field on a manifold M is a value of the dependent product $\prod_{x:M} T_x M$, and the total space of the tangent bundle is $\sum_{x:M} T_x M$. Dependent sums also occur in probability theory, when constructing sample spaces for multi-stage experiments.

The rules governing product types are the following. See the HoTT book for explanation.

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash B : \mathcal{U}_i}{\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}_i}\ \Pi\text{-FORM} \qquad \frac{\Gamma, x{:}A \vdash b : B}{\Gamma \vdash \lambda(x:A).\, b : \prod_{(x:A)} B}\ \Pi\text{-INTRO}$$

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]}\ \Pi\text{-ELIM} \qquad \frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda(x:A).\, b)(a) \equiv b[a/x] : B[a/x]}\ \Pi\text{-COMP}$$

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B}{\Gamma \vdash f \equiv (\lambda x.\, f(x)) : \prod_{(x:A)} B}\ \Pi\text{-UNIQ}$$

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 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 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 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*.
- Types whose values are all equal – types for which merely knowing that they are inhabited is all there is to know – are called *mere propositions*. See below for IsMereProp.

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": $\prod_{n:\mathbb{N}} \sum_{m:\mathbb{N}} (n = 2m)$

By reading "for all $x : X$" for "$\prod_{x:X}$" and "there exists $x : X$" for "$\sum_{x:X}$", these types can be interpreted in a simple logical way. But at the same time, they can be read in geometric/homotopy-theoretic terms; see below.

The type of monoid structures on a type X is

$$\sum_{\circ : X \times X \to X} \sum_{e : X} \left(\left(\prod_{x : X} (e \circ x = x) \right) \times \left(\prod_{x : X} (x \circ e = x) \right) \times \left(\prod_{x,y,z : X} ((x \circ y) \circ z = x \circ (y \circ z)) \right) \right).$$

- Identity witnesses can be composed: Let $p : (x = y)$ and $q : (y = z)$. Then there exists a canonically defined witness $p \centerdot q : (x = z)$.

- Composition of identity witnesses is associative. The proof of this fact is a value of the type

$$(p \centerdot (q \centerdot r) = (p \centerdot q) \centerdot r).$$

How are types like spaces?

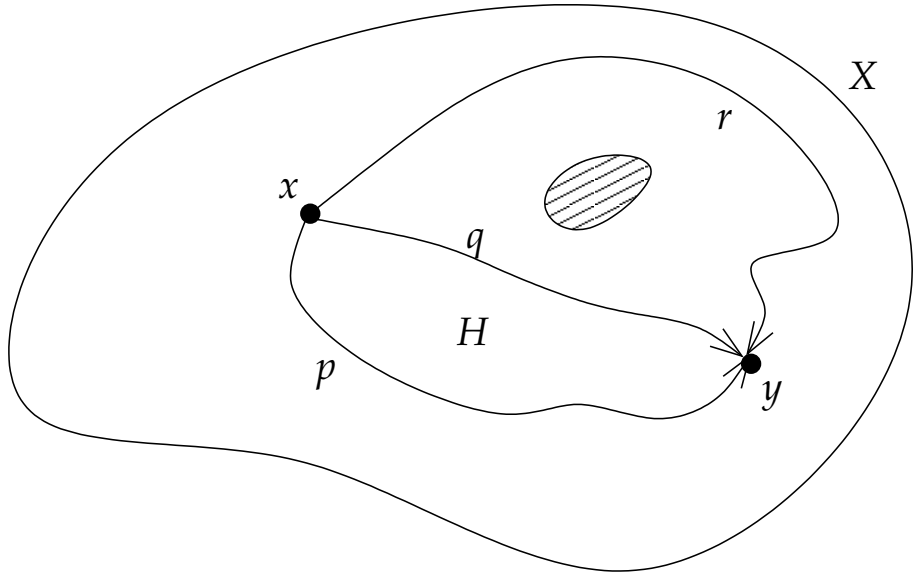
homotopy theory	type theory
space X	type X
point $x \in X$	value $x : X$
path $x \rightsquigarrow y$	value of $(x = y)$
(continuous) map	value of $X \to Y$

- 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

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



- This type has values $x, y : X$.
- The paths p, q, and r are values of $(x = y)$.
- Since p and q are homotopic, we have $(p = q)$; a witness of this fact is the value $H : (p = q)$.
- Because of the hole, p (and q) are not homotopic to r. So $\neg(p = r)$; more precisely, the type $\neg(p = r) :\equiv ((p = r) \to \mathbf{0})$ is inhabited, where $\mathbf{0}$ is the *empty type*.
- We have $(x = y) \simeq \mathbb{Z}$, where $\simeq$ denotes *equivalence*, to be discussed below.

- The type $(f \simeq g)$ is the type of homotopies between f and g. It is read as the type of "continuous families of paths $f(x) \rightsquigarrow g(x)$".

- To understand the definition of contractibility geometrically, one may not read it in logical terms: One may not read it as "there exists a point x such that any point y is connected to x".

- Instead, it should be read as follows: There exists a point x such that there is a *continuous* way of associating to any point y a path $x \rightsquigarrow y$. Convince yourself that this not possible for your favorite example of a non-contractible space (for instance, the circle).

- A space is *connected* if and only if

$$\sum_{x:X} \prod_{y:X} \|x = y\|_{-1}.$$

  Here, $\|(x = y)\|_{-1}$ is the (-1)-*truncation* of $(x = y)$; it is a mere proposition. See below.

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**":

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

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

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

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

- The only interesting feature about a mere proposition is whether it is inhabited or not.
- The type $\mathsf{IsMereProp}(X)$ is equivalent to $(X \to \mathsf{IsContr}(X))$.

How are types like spaces?

- Functions are automatically **continuous/functorial**:

$$(x = y) \longrightarrow (f(x) = f(y)).$$

- Type families $P : X \to \mathcal{U}$ automatically behave like **fibrations**, in that fibers over connected points are equivalent:

$$(x = y) \longrightarrow (P(x) \simeq P(y)).$$

- $\mathcal{U}$ is a *universe*. Its values are, to a first approximation, all types.
- For types A and B, $(A \simeq B)$ is the type of *equivalences* between A and B. See below.
- The function $(x = y) \longrightarrow (f(x) = f(y))$ is compatible with composition of identity witnesses.

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

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

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

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

Look ma! 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**

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

and a **recursion principle**

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

- $\mathcal{U}$ 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 **circle** S^1 is generated by

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

The **sphere** S^2 is generated by

- a point base : S^2 and
- a path surf : $(\mathsf{refl}_{\mathsf{base}} = \mathsf{refl}_{\mathsf{base}})$.

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 $t : (p \cdot q = q \cdot p)$.

- Note that a presentation of a type *determines*, but does not *explicitly describe* its higher identity types.
- Just like the free vector space spanned by set contains not only the given elements, but also their linear combinations, the type given by a higher inductive definition (or its higher identity types) may contain many more values than explicitly listed.
- For instance, there is a nontrivial element in $(\mathsf{refl}_{\mathsf{refl}_{\mathsf{base}}} = \mathsf{refl}_{\mathsf{refl}_{\mathsf{base}}})$, where $\mathsf{base} : S^2$, corresponding to the *Hopf fibration*.
- More generally, higher-dimensional paths are forced into existence by *proofs*. For instance, in $(\mathsf{base} = \mathsf{base})$ where $\mathsf{base} : S^1$, there are the values $\mathsf{loop} \cdot (\mathsf{loop} \cdot \mathsf{loop})$ and $(\mathsf{loop} \cdot \mathsf{loop}) \cdot \mathsf{loop}$. They are the same by a witness of type $(\mathsf{loop} \cdot (\mathsf{loop} \cdot \mathsf{loop}) = (\mathsf{loop} \cdot \mathsf{loop}) \cdot \mathsf{loop})$.
- Also, different generators may turn out to give rise to the same element.

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 $\mathrm{Cyl}(X) \simeq X \times I \simeq X$.

The *infinite-dimensional sphere* S^∞ is generated by

- a point $\mathsf{N} : S^\infty$ and
- a point $\mathsf{S} : S^\infty$ and
- for each $x : S^\infty$, a path $\mathsf{merid}(x) : (N = S)$.

So $S^\infty \simeq \Sigma S^\infty$.

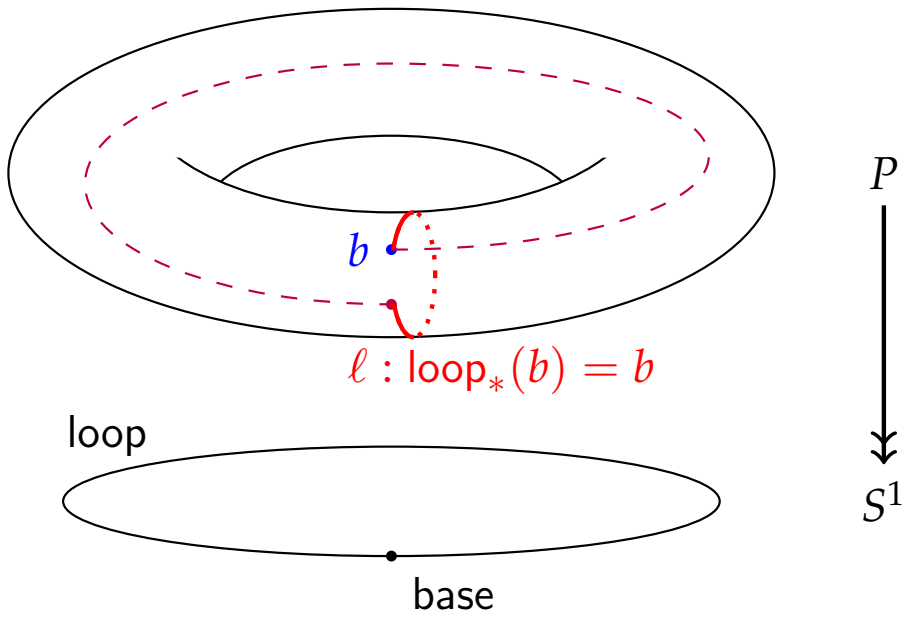
Of course, we can show $S^\infty \simeq \mathbf{1}$.

If G is a discrete group, the *classifying space* BG is generated by

- a point $\mathsf{base} : BG$,
- a function $\mathsf{loop} : G \to (\mathsf{base} = \mathsf{base})$,
- a value of type $\mathsf{loop}(e) = \mathrm{id}$,
- a value of type $\prod_{x,y:G} \mathsf{loop}(x \circ y) = \mathsf{loop}(y) \cdot \mathsf{loop}(x)$, and
- "BG is a 1-type": for each $x, y : BG$ and $p, q : (x = y)$ and $r, s : (p = q)$, a path $r = s$.

See `http://dlicata.web.wesleyan.edu/pubs/lf14em/lf14em.pdf`.

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 $\ell : \mathsf{loop}_*(b) = b$

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 $(b = b)$.

The drawing was lifted from the HoTT book.

What is path induction?

(Based) path induction states: Given

- a value a of a type A,
- a type family $C : \prod_{x:A}((a = x) \to \mathcal{U})$, and
- a value $c : C(a, \mathsf{refl}_a)$,

there is a function

$$f : \prod_{x:A} \prod_{p:(a=x)} C(x, p)$$

such that $f(a, \mathsf{refl}_a) \equiv c$.

- In particular, in proving that a proposition depending on a value x and an identity witness $p : (a = x)$ holds for all thoses values and witnesses, it suffices to prove it for the special value a and the canonical identity witness refl_a.

- Note that this does not mean that any value of $(a = x)$ is equal to refl_a! Indeed, this claim is not even well-typed.

- The induction principle only makes a statement about the whole *type family* of the $(a = x)$'s with x varying, not about individual types.

- Compare with the classical based path space: In it, any element (any path starting at x) is connected to the trivial path at x. But this does not mean that any path is homotopic to the trivial path.

As an example, let's define the path reversal function

$$\mathsf{inv} : \prod_{x:A}((a = x) \longrightarrow (x = a)),$$

where $a : A$ is a fixed value, by path induction. For this, we define the type family

$$C :\equiv ((x : A, p : (a = x)) \mapsto (x = a)) : \prod_{x:A}((a = x) \to \mathcal{U})$$

and note that we have the value

$$c :\equiv \mathsf{refl}_a : C(a, \mathsf{refl}_a).$$

Therefore, by path induction, we obtain a function

$$f : \prod_{x:A} \prod_{p:(a=x)} (x = a).$$

This is inv.

- Working informally, we would write the construction more concisely:

  "Let $p : (a = x)$ be given, we want to construct $\mathsf{inv}(p) : (x = a)$. By path induction, we may assume that $p \equiv \mathsf{refl}_a$. In this case, we define $\mathsf{inv}(p)$ as refl_a."

- Here is how we construct the path composition function:

  "Let $p : (a = b)$ and $q : (b = c)$ be given. By path induction, we may assume that $p \equiv \mathsf{refl}_a$. Again by path induction, we may assume that $q \equiv \mathsf{refl}_a$. In this case, we define $p \cdot q$ as refl_a."

- Here is how to construct the function $\mathsf{ap}_g : (x = y) \to (g(x) = g(y))$, if g is some given function:

  "By path induction, it suffices to define $\mathsf{ap}_g(p)$ when p is refl_x. In this case, we declare $\mathsf{ap}_g(p)$ to be $\mathsf{refl}_{g(x)}$."

Path induction does not allow to replace any paths whatsoever by the canonical reflexivity witnesses. For instance, the following "proof" of

$$p \cdot q = q \cdot p$$

for all paths p and q is bogus:

"By path induction, we may assume that p and q are refl_a. In this case, $p \cdot q$ and $q \cdot p$ both equal $\mathsf{refl}_a \cdot \mathsf{refl}_a = \mathsf{refl}_a$."

Indeed, the claimed statement is not even well-typed:

$$\prod_{x,y,z:X} \prod_{p:(x=y)} \prod_{q:(y=z)} (p \cdot q = q \cdot p).$$

The composition $q \cdot p$ is not defined. Weakening the statement to

$$\prod_{x:X} \prod_{p:(x=x)} \prod_{q:(x=x)} (p \cdot q = q \cdot p)$$

does not help either, since in this statement there is no free endpoint, so path induction does not apply.

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.

- A type X is *connected* if and only if

$$\sum_{x:X} \prod_{y:X} \|x = y\|_{-1}.$$

- In logical terms, this is read as follows: A type X is connected if and only if there is some value $x : X$ such that for any value $y : X$, it is *merely true* that $x = y$.

- In geometric terms, it reads: A space X is connected if and only if there is some point $x : X$ such that for any point $y : Y$, there is a path connecting x to y, and this path does not need to depend continuously on y.

- Out of a value of $\|X\|_{-1}$, we cannot extract a witness of X, i. e. there is no function of type

$$\prod_{X:\mathcal{U}}(\|X\|_{-1} \longrightarrow X).$$

- Intuitively, if $\|X\|_{-1}$ is inhabited, then we know that there must be some witnesses of X, but generally, we cannot choose a specific such witness: By our general principles, this witness would depend in a continuous way on X, but this is too much to ask for.

- However, by the universal property of (-1)-truncation, to prove an implication of the form $\|X\|_{-1} \longrightarrow Y$ where Y is itself a mere proposition, it suffices to prove the a priori weaker statement $X \longrightarrow Y$. Intuitively, in this case we *can* make the choice, because the produced value of Y does not depend on it.

- By circle induction, an equivalent definition of the fundamental group 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 $S^1 \to X$.

	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8
π_1	$\mathbb{Z}$	0	0	0	0	0	0	0
π_2	0	$\mathbb{Z}$	0	0	0	0	0	0
π_3	0	$\mathbb{Z}$	$\mathbb{Z}$	0	0	0	0	0
π_4	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	0	0	0
π_5	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	0	0
π_6	0	$\mathbb{Z}_{12}$	$\mathbb{Z}_{12}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	0
π_7	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}\times\mathbb{Z}_{12}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0
π_8	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2^2$	$\mathbb{Z}_{24}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$
π_9	0	$\mathbb{Z}_3$	$\mathbb{Z}_3$	$\mathbb{Z}_2^2$	$\mathbb{Z}_2$	$\mathbb{Z}_{24}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$
π_{10}	0	$\mathbb{Z}_{15}$	$\mathbb{Z}_{15}$	$\mathbb{Z}_{24}\times\mathbb{Z}_3$	$\mathbb{Z}_2$	0	$\mathbb{Z}_{24}$	$\mathbb{Z}_2$
π_{11}	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_{15}$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	$\mathbb{Z}_{24}$
π_{12}	0	$\mathbb{Z}_2^2$	$\mathbb{Z}_2^2$	$\mathbb{Z}_2$	$\mathbb{Z}_{30}$	$\mathbb{Z}_2$	0	0
π_{13}	0	$\mathbb{Z}_{12}\times\mathbb{Z}_2$	$\mathbb{Z}_{12}\times\mathbb{Z}_2$	$\mathbb{Z}_2^3$	$\mathbb{Z}_2$	$\mathbb{Z}_{60}$	$\mathbb{Z}_2$	0

Some of the known homotopy groups of the spheres, lifted from the HoTT book.

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.

- Read in logical terms, a function $f : X \to Y$ is an equivalence if and only if for any $y : Y$, the fiber $\mathsf{fib}_f(y)$ is a singleton, i. e. if and only if f is bijective.

- One can prove that $\mathsf{IsMerePropIsEquiv}(f))$.

- A value of $(X \simeq Y)$ is a pair consisting of a function $f : X \to Y$ together with a proof that f is an equivalence.

- Let X and Y be types, i. e. values of the universe $\mathcal{U}$. Then there is the identity type $(X = Y)$. What does it look like?
- Without the univalence axiom, this question does not have an answer; the special behaviour of $(X = Y)$ is left unspecified by the remaining rules of homotopy type theory.
- With the univalence axiom, the question has a clear answer: The type $(X = Y)$ of identity witnesses is equivalent to the type $(X \simeq Y)$ of equivalences.

- The canonical function $(X = Y) \to (X \simeq Y)$ appearing in the univalence axiom is constructed by path induction. It maps the canonical reflexivity witness refl_X to the trivial equivalence $\mathrm{id}_X : X \to X$ (together with a proof that id_X is an equivalence).

- By the univalence axiom, 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.
- Because any construction has to respect equality, the univalence axiom guarantees that *any construction respects equivalence*.
- Most nontrivial mathematical results in homotopy type theory 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 fulfilling UIP).

- *No computational interpretation of the univalence axiom is known yet.* This prevents us from *running* proofs (as computer programs). If this were possible, we could, for instance, simply run a proof of the fact that some $\pi_k(S^n)$ is cyclic (i. e. of the form $\mathbb{Z}/(m)$) to find out the value of m.

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:

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

- The real axiom of choice,

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

  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:

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

- The real axiom of choice,

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

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

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

- 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.

What are applications?

Homotopy type theory **subsumes all of classical mathematics**, by using the fragment of discrete types.

The following subjects have received special treatment:

- Homotopy theory (duh)
- Category theory
- Data type theory
- Your subject here!

Also, HoTT can be used as the **internal language** of $(\infty, 1)$-toposes.

There is also lots of activity related to homotopy type theory in type theory, $(\infty, 1)$-category theory, the theory of simplicial and cubical sets, computer-aided reasoning, ...

- In homotopy type theory, isomorphic objects of categories *are equal*.
- Like in classical category theory, any fully faithful and essentially surjective functor is an equivalence of categories. Unlike in the classical approach, this is proven by a simple appeal to the axiom (really theorem) of *unique choice*.
- Equivalent categories are equal.

- "Recall" that data types can be modeled as initial algebras for *polynomial functors*.
- For instance, the type $\mathsf{List}(X)$ of lists of type of X is the initial algebra for the functor

$$? \longmapsto 1 + X \times ?.$$

- Formally differentiating the fixed-point equation $\mathsf{List}(X) = 1 + X \times \mathsf{List}(X)$ gives

$$\partial\mathsf{List}(X) = \mathsf{List}(X) + X \times \partial\mathsf{List}(X).$$

  Formally solving this equation gives

$$\partial\mathsf{List}(X) = \mathsf{List}(X) \times \mathsf{List}(X).$$

- This data type is the *zipper for lists*, i. e. the type of *one-hole contexts* of lists.

- $\partial X^n = nX^{n-1}$: This makes sense.
- What's the antiderivative of X^n? What should $\frac{1}{n+1}X^{n+1}$ be?
- The type of *cycles* of length $n+1$!
- Unfortunately, this type can not be modeled in the standard approach.
- It can in homotopy type theory, by simply relaxing a part of the definition of polynomial functors (not shown here) to no longer restrict to *sets*, but to allow arbitrary *types*.

See `http://www.cs.nott.ac.uk/~txa/talks/lyon14.pdf`.

What is homotopy type theory?

Homotopy type theory …

- is elegant,
- reflects mathematical practice,
- contains wondrous new concepts,
- ensures that everything respects equivalences,
- simplifies the plumbing of homotopy theory,
- allows for accessible computer formalization.

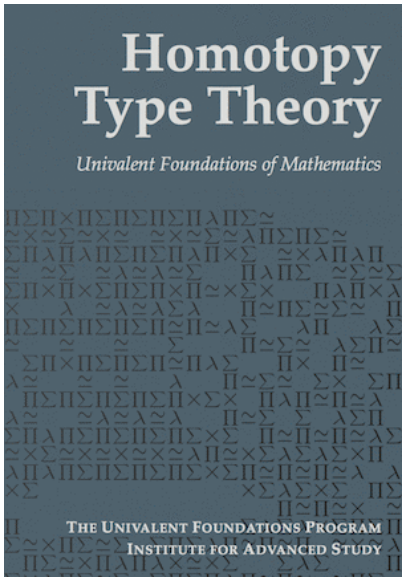
What is homotopy type theory?

Homotopy type theory …

- is elegant,
- reflects mathematical practice,
- contains wondrous new concepts,
- ensures that everything respects equivalences,
- simplifies the plumbing of homotopy theory,
- allows for accessible computer formalization.

References

- The textbook
  `http://homotopytypetheory.org/book/`
- Voevodsky on his motivations
  `http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/2014_IAS.pdf`
- Seminar slides
  `http://www.math.ias.edu/~mshulman/hottseminar2012/01intro.pdf`
  `http://www.math.ias.edu/~mshulman/hottminicourse2012/04induction.pdf`
  `https://coq.inria.fr/files/coq5-slides-spitters.pdf`
  `https://www.andrew.cmu.edu/user/awodey/hott/CMUslides.pdf`
  `http://home.sandiego.edu/~shulman/papers/hott-grothendieck.pdf`
- hott-amateurs mailing list
  `https://groups.google.com/d/forum/hott-amateurs`



An application in string theory: `https://golem.ph.utexas.edu/category/2012/05/what_is_homotopy_type_theory_g.html`

There is a vibrant community of *HoTT amateurs* – people without a formal training in mathematics who nevertheless are interested in homotopy type theory. Most of these people have a background in programming, particularly with fancy languages like Haskell.

How can we calculate $\pi_1(S^1)$?

By circle induction, we define a type family code which will turn out to be the universal covering space (think $\mathbb{R} \to S^1$):

$$\mathsf{code} : S^1 \to \mathcal{U},\ \mathsf{base} \mapsto \mathbb{Z},\ \mathsf{loop} \mapsto \mathsf{ua}(\mathsf{succ}).$$

Here, $\mathsf{ua}(\mathsf{succ}) : (\mathbb{Z} = \mathbb{Z})$ is the identity witness corresponding to the equivalence $\mathsf{succ} : \mathbb{Z} \to \mathbb{Z}$.

Thought of a fibration, code looks like this: Its fiber over base is $\mathbb{Z}$. A number v : code(base) is transported along loop to $v + 1$.

Define:

$$\begin{array}{lcccll} f & : & \mathbb{Z} & \longrightarrow & (\mathsf{base} = \mathsf{base}), & n \longmapsto \mathsf{loop}^n \\ g & : & (\mathsf{base} = \mathsf{base}) & \longrightarrow & \mathbb{Z}, & p \longmapsto \mathsf{code}(p)(0) \end{array}$$

Verify $g \circ f = \mathrm{id}$ by induction. But there is no principle which would help in proving that $f \circ g = \mathrm{id}$. For this, need to *generalize*; define by circle induction on $x : S^1$

$$\begin{array}{lcccl} f_x & : & \mathsf{code}(x) & \longrightarrow & (\mathsf{base} = x) \\ g_x & : & (\mathsf{base} = x) & \longrightarrow & \mathsf{code}(x) \end{array}$$

such that $f_{\mathsf{base}} \equiv f$ and $g_{\mathsf{base}} \equiv g$.

Then $\prod_{x:S^1}(f_x \circ g_x = \mathrm{id})$ can be proven by path induction.

In particular, we obtain $(\mathsf{base} = \mathsf{base}) \simeq \mathbb{Z}$. Taking the 0-truncation, we obtain

$$\pi_1(S^1, \mathsf{base}) :\equiv \|(\mathsf{base} = \mathsf{base})\|_0 \simeq \|\mathbb{Z}\|_0 \simeq \mathbb{Z}.$$