Introduction to homotopy type theory / univalent foundations

Tom de Jong

Institute of Computing and Information Sciences,
Radboud University, The Netherlands

Boğaziçi University Science Club Mathematics

12 November 2026

In short

Homotopy type theory / univalent foundations (HoTT/UF) is

- a foundation for mathematics,
- a suitable language for higher mathematical structures,
- implemented in computer proof assistants.

As a foundation, it is unlike set theory. Some notable differences:

- The syntax of set theory is both very minimal and unrestrictive.
  The syntax of HoTT is neither.
- In set theory, logic comes first and equality is a logical concept.
  In HoTT, logic comes second (*after* mathematics) and
  equality (identification) is a mathematical concept (a type).

Overview

1. Brief historical overview
2. Identity types and groupoids
3. Constructivism and computation
4. Key concepts from HoTT/UF
5. Discussion of resources

Tip: It's beneficial and **fun** to explore HoTT/UF in a proof assistant, such as Agda.

Consider joining the **Univalent Agda Discord**.

An abridged and incomplete history

Timeline

1970s

Per Martin-Löf presents his intensional dependent type theories (with publications in the 1980s). They feature **identity types**.

Later: **MLTT** = Martin-Löf Type Theory.

1994

Martin Hofmann and **Thomas Streicher** publish their groupoid model of MLTT. This is a "1-dimensional homotopical" model and shows that Martin-Löf's identity types can have **higher structure**.

1998

Hofmann and Streicher publish an extended version of their 1994 paper featuring "universe extensionality": a precursor to Voevodsky's univalence axiom.

2006

Workshop *Identity Types: Topological and Categorical Structure* in Uppsala, Sweden, feat. converging talks by Thomas Streicher, **Michael Warren**, **Steve Awodey**, **Richard Garner** and **Benno van den Berg**.

2006

Field Medallist **Vladimir Voevodsky** makes a note on "a homotopy λ-calculus" public.

2009

Voevodsky starts working out an interpretation of MLTT in *simplicial sets* (think: "spaces").

2010

Voevodsky starts his formal Coq library *Foundations*, which is later merged to become part of *(Coq-)UniMath*.

2011

Awodey, Garner, Martin-Löf and Voevodsky organize the Oberwolfach mini-workshop *The Homotopy Interpretation of Constructive Type Theory*.

Andrej Bauer, Peter Lumsdaine, Mike Shulman and Michael Warren conceive of **higher inductive types**.

2012–2013

A Special Year on Univalent Foundations of Mathematics is held at the Institute for Advanced Study, culminating in the collaboratively written **HoTT Book**.

2014, 2015–2021

Marc Bezem, **Thierry Coquand** and Simon Huber give a **constructive** model of HoTT in *cubical* (as opposed to *simplicial*) sets.

This gives rise to a computational version of HoTT: **cubical type theory** by Cyril Cohen, Anders Mörtberg, Coquand and Huber, and later to *Cubical Agda* by Andrea Vezossi, Andreas Abel and Mörtberg.

2019 —●

Mike Shulman's preprint vastly generalizes Voevodsky's simplicial sets model and shows HoTT can be interpreted into any $(\infty, 1)$-topos.

Recent research line: extensions of HoTT to do **synthetic mathematics**:

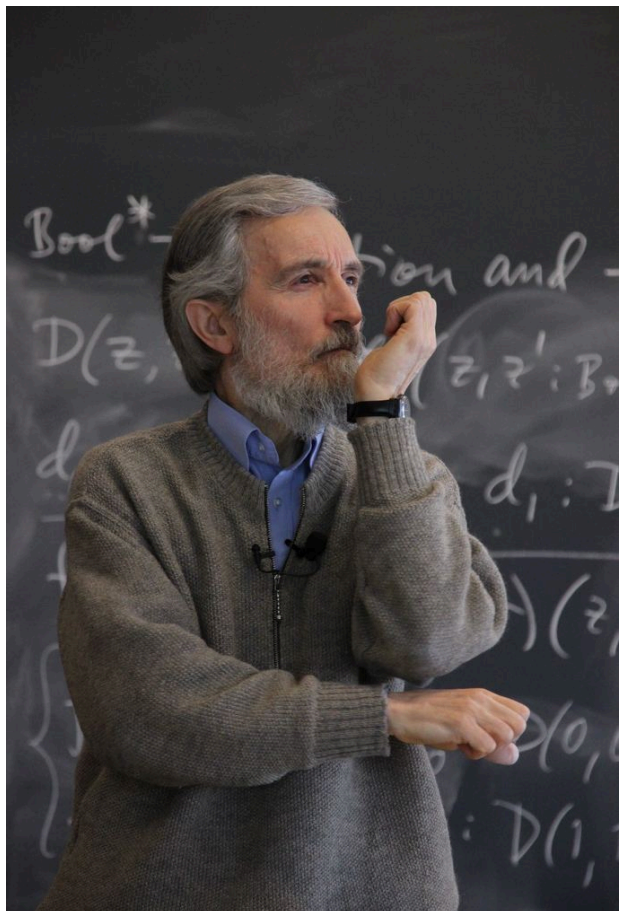
Idea: use a "domain-specific language" with fewer encodings ↵

- synthetic $(\infty, 1)$-category theory
  ↳ Riehl–Shulman, Buchholtz–Gratzer–Weinberger, Sattler–Wärn, ...
- synthetic algebraic geometry
  ↳ Blechschmidt, Cherubini, Coquand, Ritter (prev. Hutzler), Moeneclaey, ...

Identity types, groupoids and homotopy theory

Martin-Löf's identity types

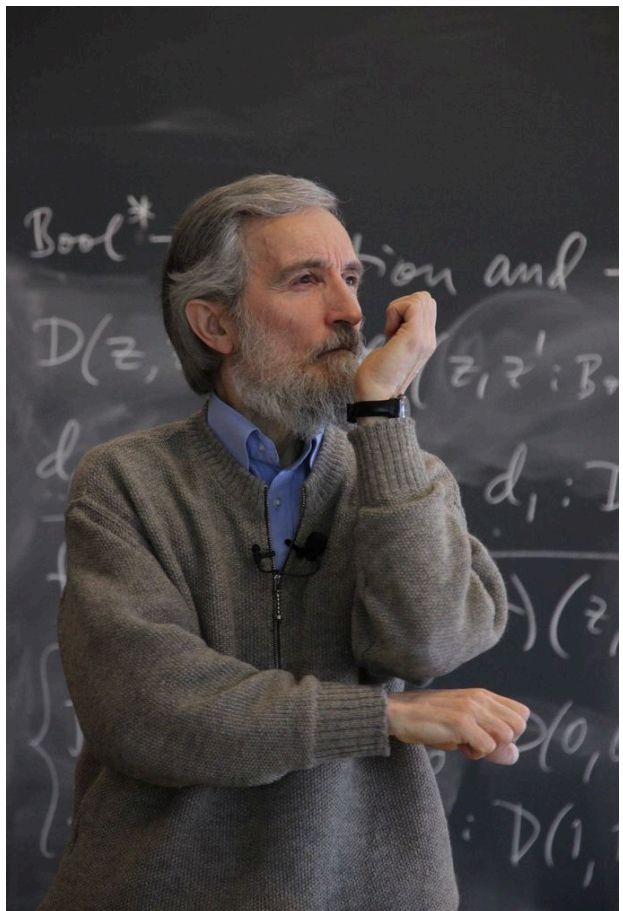
- For every type A and terms $a, b : A$, there is a new type $\mathrm{Id}_A(a, b)$ of identities/identifications between a and b.
  - This is Martin-Löf's *formation* rule for the identity type.



Per Martin-Löf

Martin-Löf's identity types

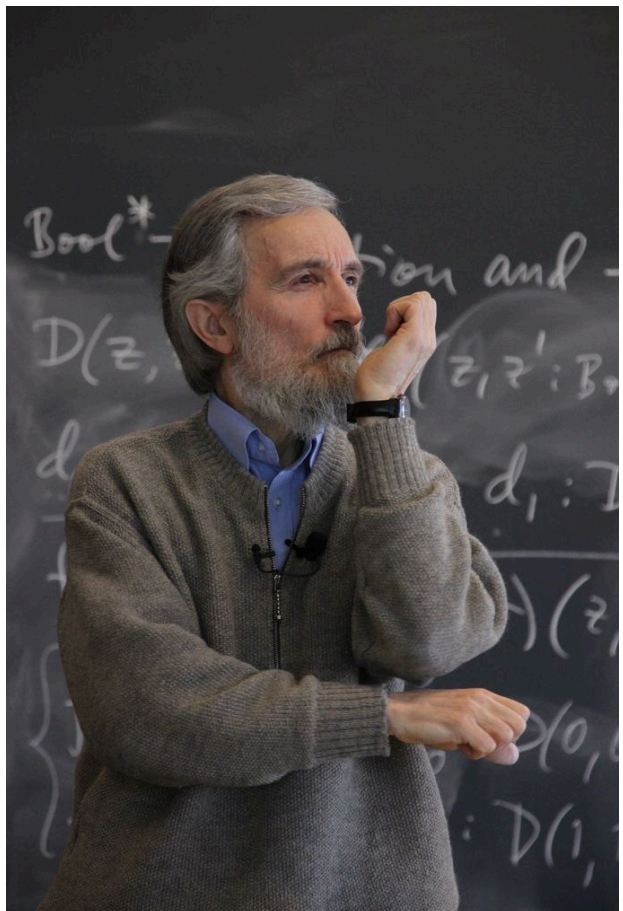
- For every type A and terms $a, b : A$, there is a new type $\mathrm{Id}_A(a, b)$ of identities/identifications between a and b.
  - This is Martin-Löf's *formation* rule for the identity type.
- **Reflexivity**: For every $a : A$, we have a term $\mathrm{refl}\, a : \mathrm{Id}_A(a, a)$.
  - This is Martin-Löf's *introduction rule* for the identity type.



Per Martin-Löf

Martin-Löf's identity types

- For every type A and terms $a, b : A$, there is a new type $\mathrm{Id}_A(a, b)$ of identities/identifications between a and b.
  - This is Martin-Löf's *formation* rule for the identity type.
- **Reflexivity**: For every $a : A$, we have a term $\mathrm{refl}\, a : \mathrm{Id}_A(a, a)$.
  - This is Martin-Löf's *introduction rule* for the identity type.
- We can construct **symmetry** and **transitivity** witnesses, e.g. if we have $p : \mathrm{Id}_A(a, b)$ and $q : \mathrm{Id}_A(b, c)$, then we can construct a new element $r : \mathrm{Id}_A(a, c)$.
  - This is done using Martin-Löf's *elimination rule* for the identity type. It's also known as the **J-rule**.



Per Martin-Löf

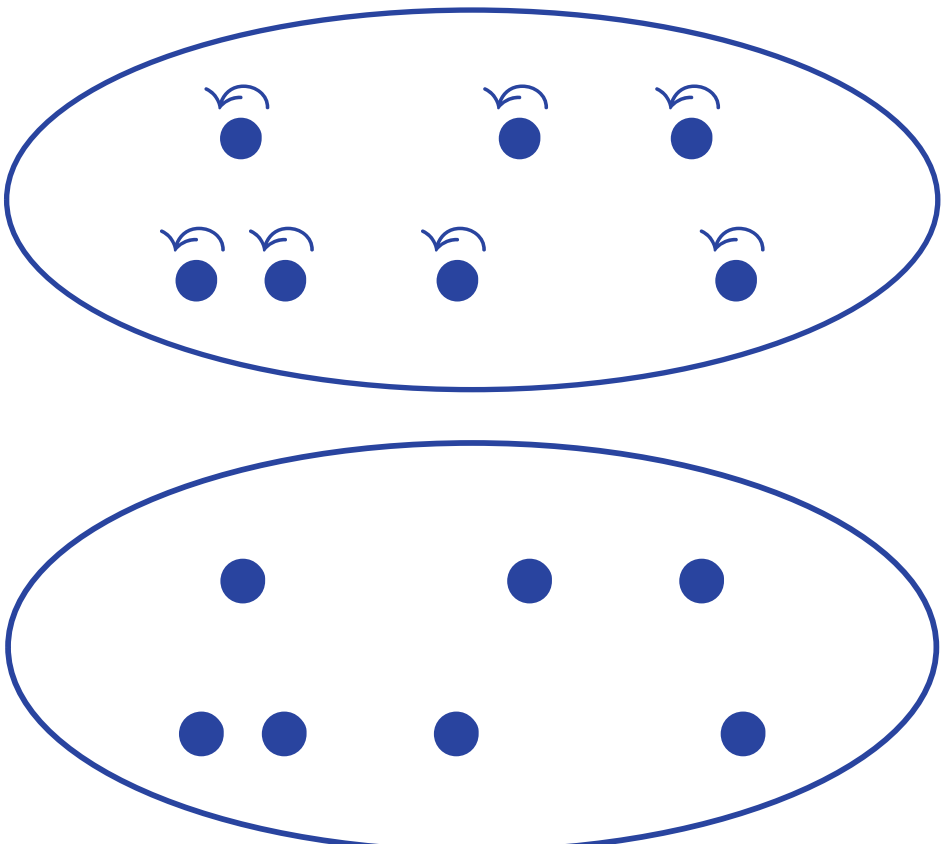


Thomas Streicher (1958–2025)

Maybe you'd expect that every $p : \mathrm{Id}_A(a, a)$ must be $\mathrm{refl}\, a$?

- This is *not* provable in MLTT!
- In HoTT there will be types A with points $a : A$, for which this is false, and this will be a **feature**, not a bug!
- But in ordinary MLTT, you can consistently add it as an axiom. In fact, it's known as **Streicher's Axiom K**.

Under Axiom K, types just look like a collection of points:



An illustration of a type with some elements depicted as • with ↶ depicting refl at that point.

It is common to omit the reflexivities, so that we draw the above just as on the left.

Hofmann & Streicher's groupoid model

- **Definition.** A **groupoid** is a set X together with for every $x, y \in X$, a set of "paths" from x to y such that
  - we always have a path $\mathrm{id}_x : x \rightsquigarrow x$,
  - we can compose paths: given $p : x \rightsquigarrow y$ and $q : y \rightsquigarrow z$ we have $q \circ p : x \rightsquigarrow z$, and this operation must be associative and obey $p \circ \mathrm{id}_x = p = \mathrm{id}_x \circ p$.
  - we can invert a path $p : x \rightsquigarrow y$ to $p^{-1} : y \rightsquigarrow x$ such that $p^{-1} \circ p = \mathrm{id}_x$ and $p \circ p^{-1} = \mathrm{id}_y$.



Martin Hofmann
(1965–2018)

Hofmann & Streicher's groupoid model

- **Definition.** A **groupoid** is a set X together with for every $x, y \in X$, a set of "paths" from x to y such that
  - we always have a path $\mathrm{id}_x : x \rightsquigarrow x$,
  - we can compose paths: given $p : x \rightsquigarrow y$ and $q : y \rightsquigarrow z$ we have $q \circ p : x \rightsquigarrow z$, and this operation must be associative and obey $p \circ \mathrm{id}_x = p = \mathrm{id}_x \circ p$.
  - we can invert a path $p : x \rightsquigarrow y$ to $p^{-1} : y \rightsquigarrow x$ such that $p^{-1} \circ p = \mathrm{id}_x$ and $p \circ p^{-1} = \mathrm{id}_y$.
- Trivial example: set X where the *only* paths are $\mathrm{id}_x : x \rightsquigarrow x$.
- Nontrivial example: $X = \{*\}$, a path $* \rightsquigarrow *$ is an integer $k \in \mathbb{Z}$, $\mathrm{id}_* = 0$, composition is addition of integers, and k^{-1} is $-k$.



Martin Hofmann
(1965–2018)

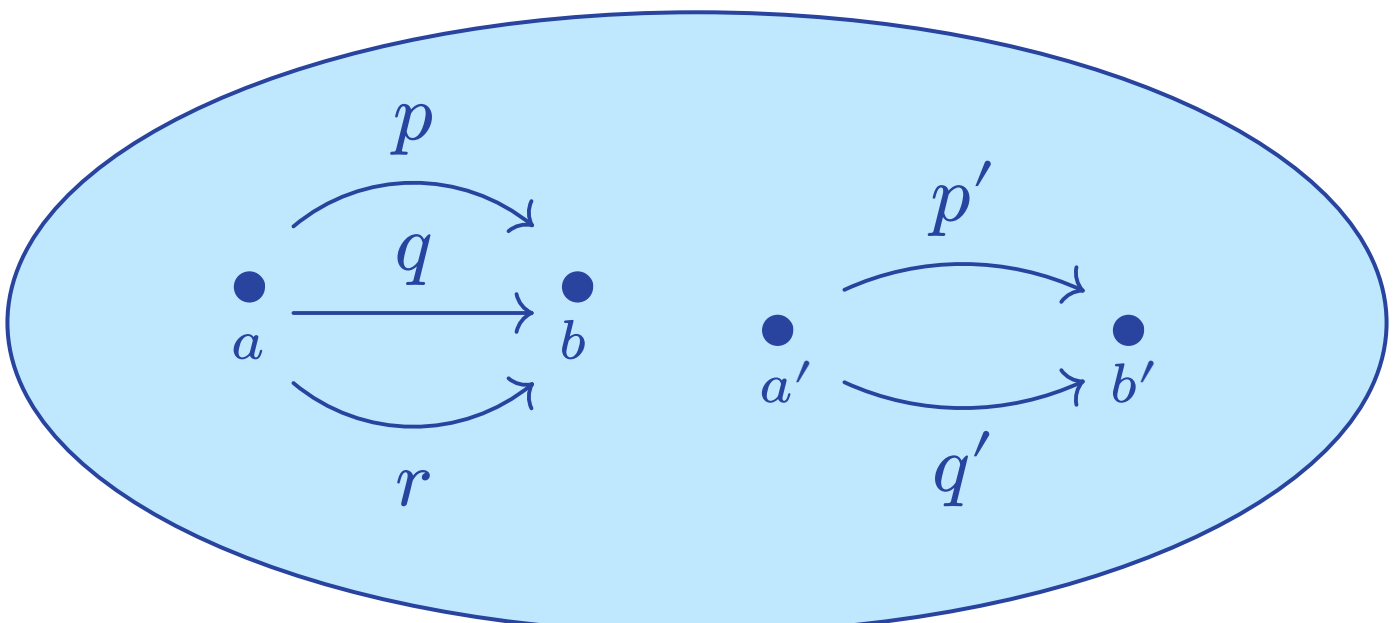
Hofmann & Streicher's groupoid model

- **Definition.** A **groupoid** is a set X together with for every $x, y \in X$, a set of "paths" from x to y such that
  - we always have a path $\mathrm{id}_x : x \rightsquigarrow x$,
  - we can compose paths: given $p : x \rightsquigarrow y$ and $q : y \rightsquigarrow z$ we have $q \circ p : x \rightsquigarrow z$, and this operation must be associative and obey $p \circ \mathrm{id}_x = p = \mathrm{id}_x \circ p$.
  - we can invert a path $p : x \rightsquigarrow y$ to $p^{-1} : y \rightsquigarrow x$ such that $p^{-1} \circ p = \mathrm{id}_x$ and $p \circ p^{-1} = \mathrm{id}_y$.
- Trivial example: set X where the *only* paths are $\mathrm{id}_x : x \rightsquigarrow x$.
- Nontrivial example: $X = \{*\}$, a path $* \rightsquigarrow *$ is an integer $k \in \mathbb{Z}$, $\mathrm{id}_* = 0$, composition is addition of integers, and k^{-1} is $-k$.
- In the model, a type A is a groupoid $[\![A]\!]$ and $p : \mathrm{Id}_{A(a,b)}$ is a path $[\![a]\!] \rightsquigarrow [\![b]\!]$ in $[\![A]\!]$.



Martin Hofmann
(1965–2018)

In the groupoid model, types look like a collection of points **with paths** between them:



The Uppsala workshop — Beyond groupoids

Identity Types - Topological and Categorical Structure Workshop

https://www2.math.uu.se/~palmgren/itt/

Theme of workshop *The identity type is one of the most intriguing constructions of Martin-Löf type theory. Its complexity became apparent with the Hofmann–Streicher groupoid model of type theory.*

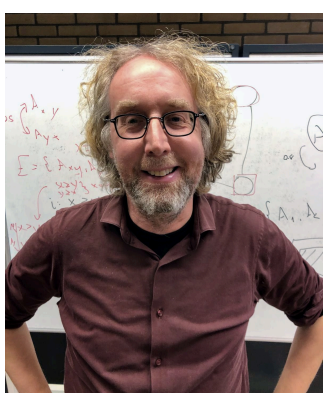
This model also hinted at some possible connections between type theory and homotopy theory and higher categories. Exploration of this connection is intended to be the main theme of the workshop.



Michael Warren



Steve Awodey



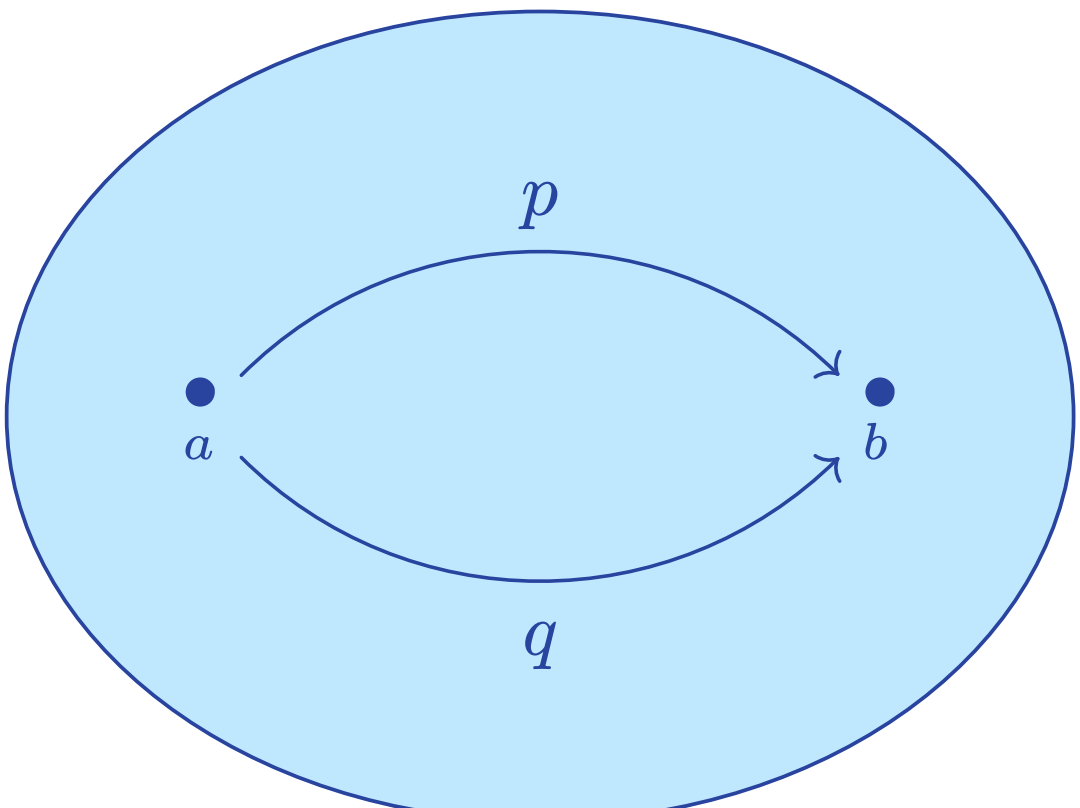
Benno van den Berg



Richard Garner

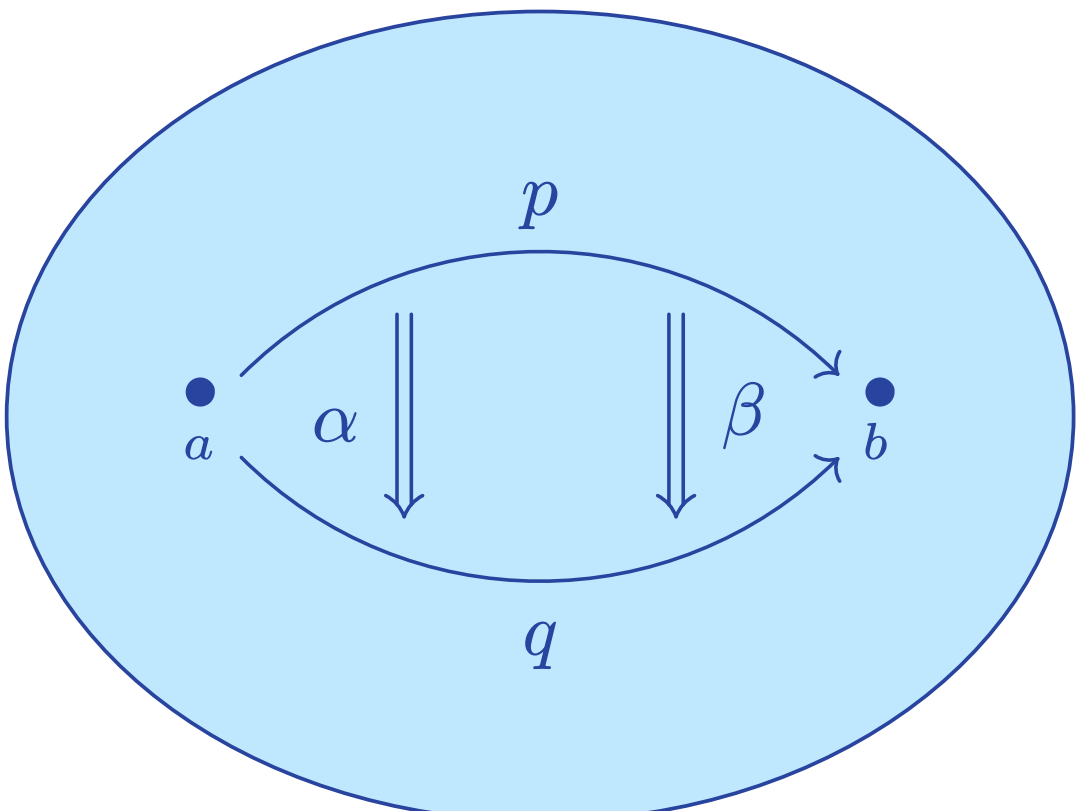
In HoTT, we picture types as **higher groupoids** that in addition to paths between points, can have (higher) paths between paths.

$$A \ \text{Type}$$
$$a, b : A$$
$$p, q : \mathrm{Id}_A(a, b)$$



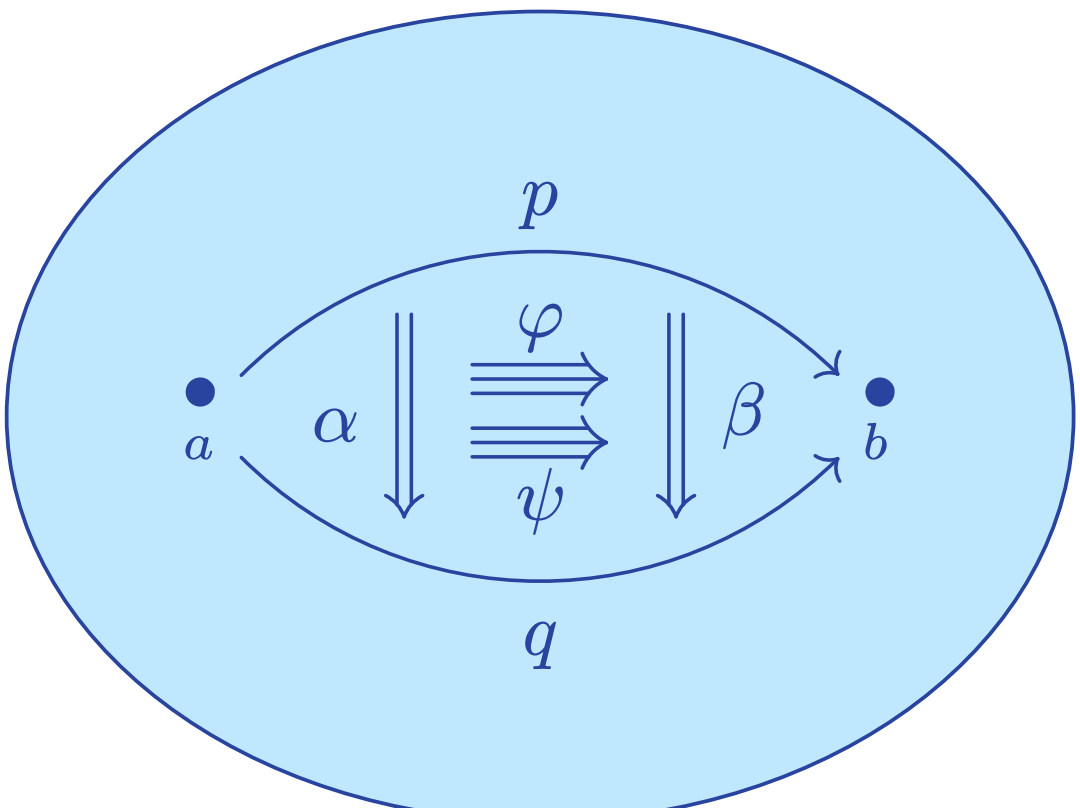
In HoTT, we picture types as **higher groupoids** that in addition to paths between points, can have (higher) paths between paths.

$$A \ \text{Type}$$
$$a, b : A$$
$$p, q : \mathrm{Id}_A(a, b)$$
$$\alpha, \beta : \mathrm{Id}_{\mathrm{Id}_A(a,b)}(p, q)$$



In HoTT, we picture types as **higher groupoids** that in addition to paths between points, can have (higher) paths between paths.

$$
\begin{aligned}
&A \ \text{Type}\\
&a, b : A\\
&p, q : \mathrm{Id}_A(a, b)\\
&\alpha, \beta : \mathrm{Id}_{\mathrm{Id}_A(a,b)}(p, q)\\
&\varphi, \psi : \mathrm{Id}_{\mathrm{Id}_{\mathrm{Id}_A(a,b)}(p,q)}(\alpha, \beta)\\
&\quad\vdots
\end{aligned}
$$



Voevodsky's journey into foundations

Voevodsky's journey into foundations

Some quotes from a lecture by Voevodsky
https://www.ias.edu/ideas/2014/voevodsky-origins

- *Starting from 1993 multiple groups of mathematicians studied [my] "Cohomological Theory" 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.***



Vladimir Voevodsky
(1966–2017)

- [On the mathematics of his “2-theories”]

To do the work at the level of rigor and precision I felt was necessary would take an enormous amount of effort and would produce a text that would be very difficult to read. And ***who would ensure that I did not forget something and did not make a mistake****, if even the mistakes in much more simple arguments take years to uncover?*

I think it was at this moment that I largely stopped doing what is called “curiosity driven research” and started to think seriously about the future.
It soon became clear that the only real long-term solution to the problems [...] is to ***start using computers in the verification of mathematical reasoning****.*

- [On the mathematics of his “2-theories”]

To do the work at the level of rigor and precision I felt was necessary would take an enormous amount of effort and would produce a text that would be very difficult to read. And ***who would ensure that I did not forget something and did not make a mistake****, if even the mistakes in much more simple arguments take years to uncover?*

I think it was at this moment that I largely stopped doing what is called “curiosity driven research” and started to think seriously about the future.
It soon became clear that the only real long-term solution to the problems [...] is to ***start using computers in the verification of mathematical reasoning****.*

😉 Some words [emphasis mine] of encouragement for you *and* me:

I would like to thank all of those who are trying to understand the ideas of Univalent Foundations, to develop these ideas and to communicate these ideas to others. ***I know it is difficult****.*

[TYPES/announce] homotopy lambda calculus

Vladimir Voevodsky vladimir at ias.edu
Fri Sep 29 10:06:07 EDT 2006

- Previous message: [TYPES/announce] INRIA postdoc position
- Next message: [TYPES/announce] Full Professor FormalMethods and Tools Twente
- **Messages sorted by:** [date] [thread] [subject] [author]

```
I have recently produced a short note where I try to explain what
homotopy lambda calculus is (or is supposed to be). Here it is:



-------------- next part --------------
A non-text attachment was scrubbed...
Name: 2006_09_Hlambda.pdf
Type: application/pdf
Size: 116584 bytes
Desc: not available
Url : http://lists.seas.upenn.edu/pipermail/types-announce/attachments/20060929/c8464d57/2006_09_Hlambda.pdf
-------------- next part --------------


Vladimir (Voevodsky)
```

Voevodsky's email to the TYPES/announce mailing list on 29 September 2006

https://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/Hlambda_short_current.pdf

Writing mathematics in type theory

Informal type theory



Peter Aczel (1941–2023)

Mathematicians relying on set theory don't work with the *formal* rules of set theory.

One innovation of the special year, spurred by Aczel, was to develop mathematics in type theory, in an *informal* (but still rigorous!) way.

To see the effect of "informal type theory", **compare** the following excerpt from the 2017 paper *Notions of Anonymous Existence in Martin-Löf type Theory* by Kraus, Escardó, Coquand and Altenkirch

Lemma 3.2. *If a type has decidable equality, its path spaces have constant endofunctions:*

$$\mathsf{isDiscrete}X \to \mathsf{pathConstEndo}\,X. \tag{3.2}$$

Proof. Given inhabitants x and y of X, we get by assumption either an inhabitant of $x = y$ or an inhabitant of $\neg(x = y)$. In the first case, we construct the required constant function $(x = y) \to (x = y)$ by mapping everything to this given path. In the second case, we have a proof of $\neg(x = y)$, and the canonical function is constant automatically. □

with the original treatment in Hedberg's 1998 *A coherence theorem for Martin-Löf's type theory* on the next slide.

On any decidable set a constant function may be defined with values in the same set. The proof will give two families of functions (for an arbitrary set A). Recall that $\mathscr{D}(A)$ stands for (the set representation of) the formula 'A or not A'.

$$\begin{array}{ll} con_A(z) : A \to A & [z \in \mathscr{D}(A)] \\ iscon_A(z,x,y) \in I(con_A(z,x), con_A(z,y)) & [z \in \mathscr{D}(A); x, y \in A] \end{array}$$

Assume we have a set A. By $+$-elimination, applied to the constant family of sets A $[z \in \mathscr{D}(A)]$, we define a function $con_A : \mathscr{D}(A) \to A \to A$ subject to the definitional equalities

$$\begin{cases} con_A(i(x), a) = x & [x \in A] \\ con_A(j(f), a) = a & [f \in (\Pi x \in A) N_0] \end{cases}$$

and, to define the function $iscon_A$, i.e. to prove that $con_A(d)$ is a constant function for an arbitrary element d of $\mathscr{D}(A)$, we assume that we have two elements a and a' of A, and consider the family of propositions (sets) C given by $C(z) = con_A(z,a) \simeq con_A(z,a')$ $[z \in \mathscr{D}(A)]$.

By $+$-elimination, $C(z)$ is proved for arbitrary $z \in \mathscr{D}(A)$ when we have proved the special cases, $C(i(x))[x \in A]$ and $C(j(f))[f \in (\Pi x \in A) N_0]$. Recalling the definition of C and the equations of *con*, this leaves us with proving, after simplification, in the first case, that $I(x,x)$ is inhabited for $x \in A$, and in the second, that $I(a,a')$ is inhabited for $f \in (\Pi x \in A) N_0$. The first case is solved by reflexivity of identity, and the second case is solved by absurdity elimination – we can apply f to any of the two elements a and a' of A, to get an element of N_0.

```
con ∈ (A ∈ Set;
       d ∈ Dec(A);
       a ∈ A
       )A
    con ≡
       [A, d, a]
          when(A,
               Prod(A, [h]N0),
               [h]A,
               [x]x,
               [f]a,
               d)

iscon ∈ (A ∈ Set;
         d ∈ Dec(A);
         a, a' ∈ A
         )I(A, con(A, d, a), con(A, d, a'))
   iscon ≡
      [A, d, a, a']
         when(A,
              Prod(A, [h]N0),
              [z]I(A, con(A, z, a), con(A, z, a')),
              [x]ref(A, x),
              [f]case0([h]I(A, a, a'), app(A, [h]N0, f, a)),
              d)
```

Since the HoTT Book

Earlier work on HoTT often featured fairly elaborate calculations, usually involving transport. Since then, we have found abstractions to handle this more swiftly, e.g.:

- **equational reasoning with equivalences**, cf. my expository paper Formalizing equivalences without tears.
- **characterizing identity types via general machinery**
  - structure identity principles (Coquand & Danielsson; Escardó; Ahrens, North, Shulman, Tsementzis; Angiuli, Cavallo, Mörtberg, Zeuner).
  - fundamental theorem of identity types (Rijke; Escardó ("Yoneda")) vs *encode-decode*
- descent/flattening to deal with pushouts

Constructivism and computation

Martin-Löf on mathematics and computation

From Per Martin-Löf's 1982 Constructive Mathematics and Computer Programming:
https://archive-pml.github.io/martin-lof/pdfs/Constructive-mathematics-and-computer-programming-1982.pdf

[...] the close connection between constructive mathematics and programming explains why the intuitionistic type theory (Martin-Löf 1975), which I began to develop solely with the philosophical motive of clarifying the syntax and semantics of intuitionistic mathematics, may equally well be viewed as a programming language.

↳ Proof assistants!

Martin-Löf on mathematics and computation

From Per Martin-Löf's 1982 Constructive Mathematics and Computer Programming:
https://archive-pml.github.io/martin-lof/pdfs/Constructive-mathematics-and-computer-programming-1982.pdf

[...] the close connection between constructive mathematics and programming explains why the intuitionistic type theory (Martin-Löf 1975), which I began to develop solely with the philosophical motive of clarifying the syntax and semantics of intuitionistic mathematics, may equally well be viewed as a programming language.

↳ Proof assistants!

Curry–Howard–Lambek correspondence:

- the proof of $P \to (Q \to P)$ corresponds to the program that takes two inputs and returns only the first; in λ-calculus notation $\lambda p.\lambda q.p$.
- the inference rule $\dfrac{P \to R \qquad Q \to R}{P \vee Q \to R}$ corresponds to a case analysis program.

In the concluding remarks of that paper, Martin-Löf writes:

[the] uninhibited use [of excluded middle] would lead to programs that you did not know how to execute.

Indeed, we can add excluded middle, $P \vee \neg P$, as an axiom to MLTT, but how would we compute with it?

Why cubical type theory?

- By Voevodsky's simplicial sets model, we know that we can consistently add his univalence axiom to MLTT.

  But just like with excluded middle, that does not give it a computational interpretation.

  In particular, proofs that use univalence may get "stuck" on it when we try to compute with them.



Thierry Coquand

Why cubical type theory?

- By Voevodsky's simplicial sets model, we know that we can consistently add his univalence axiom to MLTT.

  But just like with excluded middle, that does not give it a computational interpretation.

  In particular, proofs that use univalence may get "stuck" on it when we try to compute with them.

- Coquand et al. created **cubical type theory** to give a computational meaning to the univalence axiom: in cubical type theory there is a program that implements univalence.

- Cubical type theory was inspired by a **constructive** model in cubical (as opposed to simplicial) sets, so the (constructive) semantics came first.



Thierry Coquand

HoTT and constructive/classical mathematics

By default, HoTT is constructive in the sense that excluded middle or the axiom of the choice are not baked in. However, as detailed on the next slide,

HoTT is compatible with classical axioms (provided they are phrased carefully).

Question: So why is there often a focus on constructive mathematics in HoTT?

HoTT and constructive/classical mathematics

By default, HoTT is constructive in the sense that excluded middle or the axiom of the choice are not baked in. However, as detailed on the next slide,

HoTT is compatible with classical axioms (provided they are phrased carefully).

Question: So why is there often a focus on constructive mathematics in HoTT?

Some answers:

1. For historical/social reasons: type theory originated as a system for constructive mathematics.
2. Because of the connection to programming, an experience strenghtened / (best?) felt by formalizing mathematics (in Agda).
3. Because of the synthetic mathematics programme: even if you "believe in" excluded middle and the axiom of choice, these principles fail in some toposes.
   This is not a matter of philosophy/taste but of mathematics!

Voevodsky's univalent foundations

Voevodsky's univalent foundations



The simplicial sets model of HoTT is very relevant and important, but I focus on the novelties of Voevodsky's *Foundations* Coq library here.

The significance lied in his **new and elegant definitions**, among them:

- n-(truncated) types (called "hlevels" by Voevodsky), with **contractible** types as a fundamental instance;
- the **fiber** of a function;
- a well behaved notion of **equivalence**;
- the **univalence axiom** which connects equivalences with Martin-Löf's identity types.

n-types

Voevodsky realized that MLTT's identity types can be used to **define** (higher) groupoids *inside type theory*.

For $n \in \{-2, -1, 0, \ldots\}$, we can define a hierarchy of n-types (a.k.a. n-truncated types)

$$-2\text{-types} \subset -1\text{-types} \subset 0\text{-types} \subset \ldots \qquad \text{with}$$

- 0-types being ordinary sets ("types à la Axiom K");
- 1-types being ordinary groupoids ("types à la Hofmann–Streicher");

---

n-types

Voevodsky realized that MLTT's identity types can be used to **define** (higher) groupoids *inside type theory*.

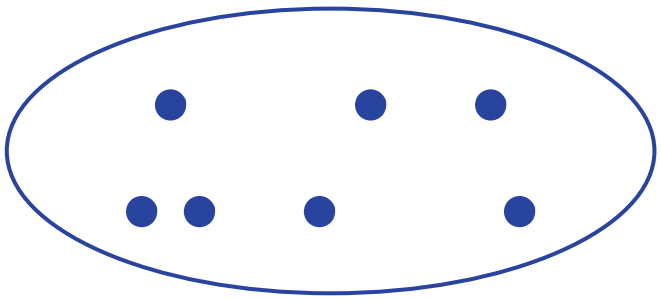
For $n \in \{-2, -1, 0, \ldots\}$, we can define a hierarchy of n-types (a.k.a. n-truncated types)

$$-2\text{-types} \subset -1\text{-types} \subset 0\text{-types} \subset \ldots \qquad \text{with}$$

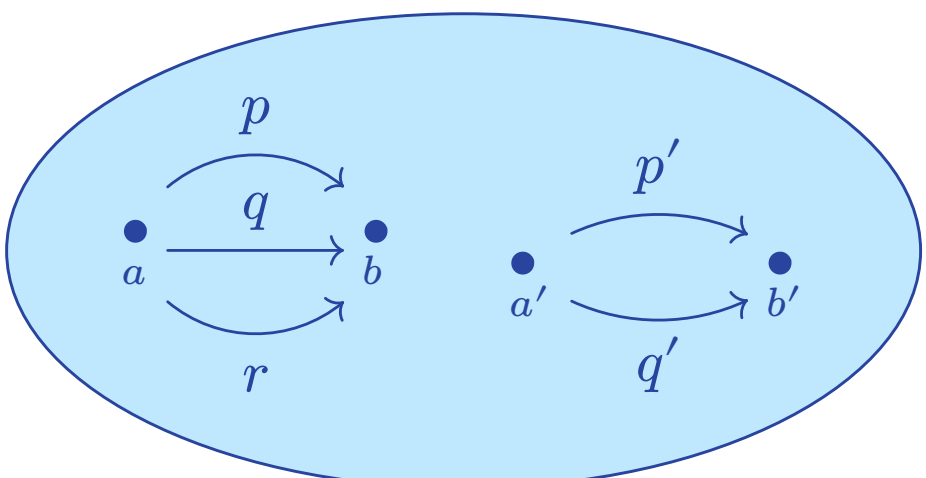
- 0-types being ordinary sets ("types à la Axiom K");
- 1-types being ordinary groupoids ("types à la Hofmann–Streicher");

- -1-types are the **propositions** (or **subsingletons**) — this is where we will **recover logic**;
- -2-types have just a single point.
  These are known as **contractible** — this notion is surprisingly fundamental!

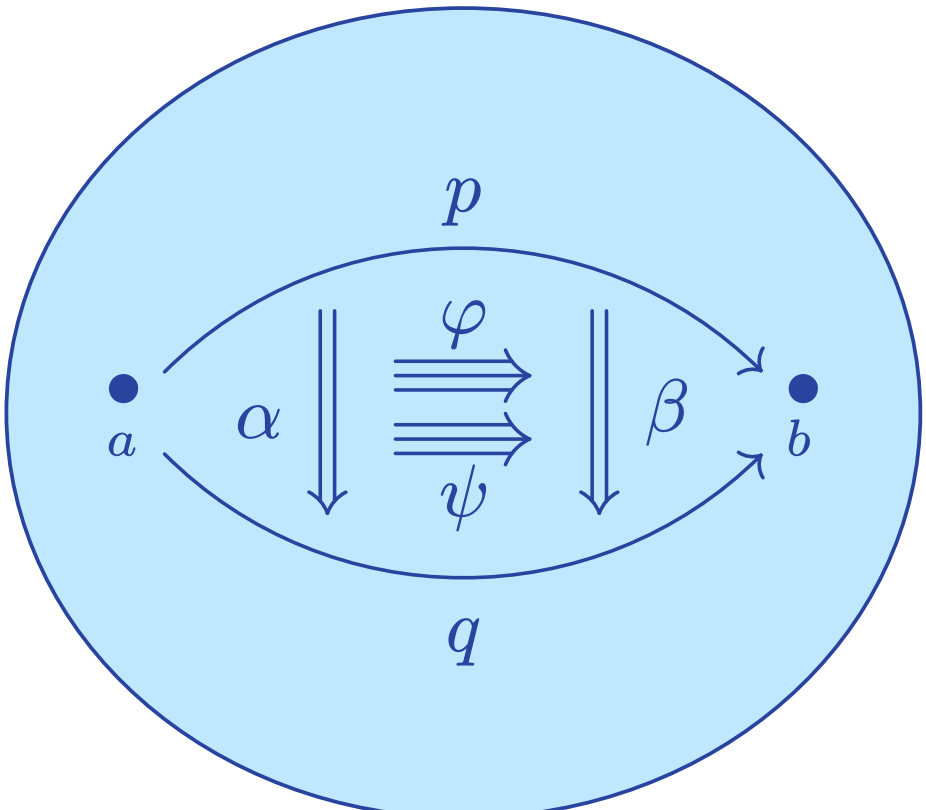
For $n \geq 0$, a good way to think of n-types is as those types whose (iterated) identity types trivialize above level n.



An illustration of a 0-type.



An illustration of a 1-type.



An illustration of a higher type.

Examples of (higher) types

- A **proposition**, **subsingleton**, or **-1-type** is a type with at most one element.
- A **0-type** or **set** is a type whose identity types all have at most one element, i.e. all its identity types are propositions.
- An **$n+1$-type** is a type whose identity types are all n-types.

Examples of (higher) types

- A **proposition**, **subsingleton**, or **-1-type** is a type with at most one element.
- A **0-type** or **set** is a type whose identity types all have at most one element, i.e. all its identity types are propositions.
- An **$n+1$-type** is a type whose identity types are all n-types.

- The empty type $\mathbb{0}$ and the type $\mathbb{1}$ with exactly one element are examples of propositions, i.e. -1-types.
- We assume to have an inductive type $\mathbb{N}$ of natural numbers. It can be shown to be a 0-type.

  We can construct $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ as usual, and the resulting types are 0-types.
- With **univalence**, we can construct examples of n-types for any n, and even of untruncated types (i.e. types with nontrivial identity types at any level).

Function extensionality (warm up to the univalence axiom)

Question: When should two functions be equal?

Answer: Two functions $f, g : A \to B$ should be equal when they are pointwise equal, i.e. for every point a in A, the values $f(a)$ and $g(a)$ are equal.

Function extensionality (warm up to the univalence axiom)

Question: When should two functions be equal?

Answer: Two functions $f, g : A \to B$ should be equal when they are pointwise equal, i.e. for every point a in A, the values $f(a)$ and $g(a)$ are equal.

- Suppose we have functions $f, g : A \to B$ in MLTT. We can construct a type $f \sim g$ expressing that f and g are pointwise equal.
- We can also consider the identity type $\mathrm{Id}_{A \to B}(f, g)$. How does it compare to $f \sim g$?

Function extensionality (warm up to the univalence axiom)

Question: When should two functions be equal?

Answer: Two functions $f, g : A \to B$ should be equal when they are pointwise equal, i.e. for every point a in A, the values $f(a)$ and $g(a)$ are equal.

- Suppose we have functions $f, g : A \to B$ in MLTT. We can construct a type $f \sim g$ expressing that f and g are pointwise equal.
- We can also consider the identity type $\mathrm{Id}_{A\to B}(f, g)$. How does it compare to $f \sim g$?
- We always have $\mathrm{Id}_{A\to B}(f, g) \longrightarrow f \sim g$, but the converse is not derivable in plain MLTT.
- Therefore, people considered the **function extensionality** axiom:

$$f \sim g \longrightarrow \mathrm{Id}_{A\to B}(f, g) \quad \text{“pointwise equal functions are equal”.}$$

The univalence axiom: universe extensionality

Question: When should two types, living in a universe $\mathcal{U}$, be equal?

Voevodsky's answer: Two such types A and B should be equal when they are **equivalent**.

The univalence axiom: universe extensionality

Question: When should two types, living in a universe $\mathcal{U}$, be equal?

Voevodsky's answer: Two such types A and B should be equal when they are **equivalent**.

- This requires a good notion of equivalence! Fortunately, Voevodsky provided this; he defined what it means for a function $f : A \to B$ to be an equivalence.
  Voevodsky: f is an equivalence when its fibers are contractible types. Alternatively, when it has a left and right inverse.
- We can collect all equivalences from A to B into a type $A \simeq B$.
- A naive formulation of the above answer would be to assume that there is a map $A \simeq B \longrightarrow \mathrm{Id}_{\mathcal{U}}(A, B)$.

The univalence axiom: universe extensionality

Question: When should two types, living in a universe $\mathcal{U}$, be equal?

Voevodsky's answer: Two such types A and B should be equal when they are **equivalent**.

- This requires a good notion of equivalence! Fortunately, Voevodsky provided this; he defined what it means for a function $f : A \to B$ to be an equivalence.
  Voevodsky: f is an equivalence when its fibers are contractible types. Alternatively, when it has a left and right inverse.
- We can collect all equivalences from A to B into a type $A \simeq B$.
- A naive formulation of the above answer would be to assume that there is a map $A \simeq B \longrightarrow \mathrm{Id}_{\mathcal{U}}(A, B)$.
- A more refined formulation, **Voevodsky's univalence axiom**, says that the canonical map $\mathrm{Id}_{\mathcal{U}}(A, B) \longrightarrow A \simeq B$ which can always be defined is itself an equivalence.

Univalence at work: examples of higher types

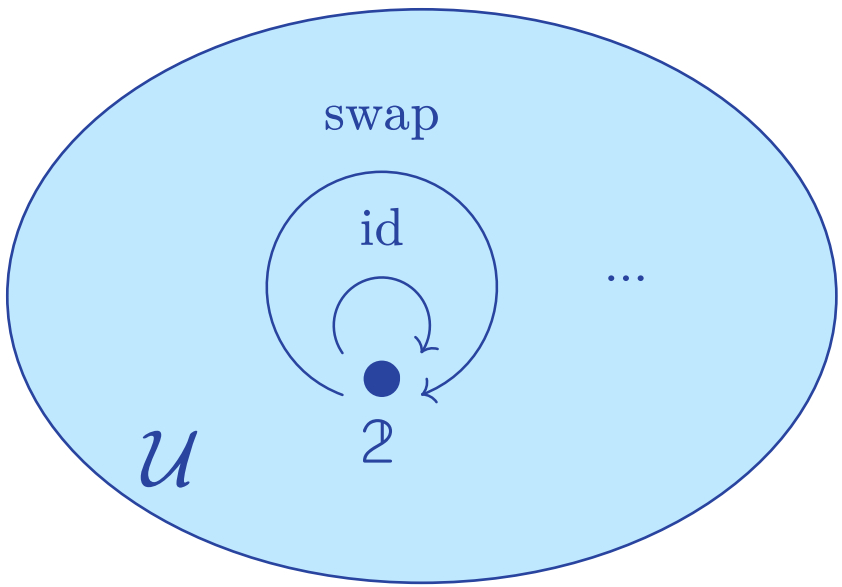
Universes

Let 2 be the type with exactly two elements (0 and 1, say) and let $\mathcal{U}$ be a universe that contains it.

If $\mathcal{U}$ is univalent, then $\mathrm{Id}_{\mathcal{U}}(2, 2)$ and $2 \simeq 2$ are equivalent.

But it's easy to see that there exactly two equivalences from 2 to itself, namely the identity and the map that swaps 0 and 1.

Thus, the identity type $\mathrm{Id}_{\mathcal{U}}(2, 2)$ has exactly two elements, and $\mathcal{U}$ is an example of a higher type.

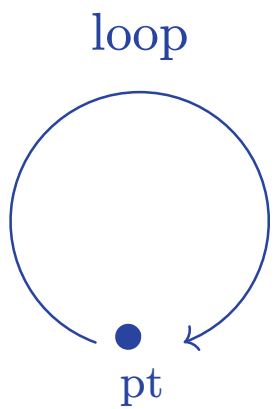


Circle

The circle is an example of a **higher inductive type**.

While the type $\mathbb{N}$ of natural numbers is inductively generated by two constructors $0 : \mathbb{N}$ and $\mathrm{succ}(n) : \mathbb{N}$ (for $n : \mathbb{N}$), the circle S^1 is inductively generated by

- a single point constructor $\mathrm{pt} : S^1$,
- and a *path* constructor $\mathrm{loop} : \mathrm{Id}_{S^1}(\mathrm{pt}, \mathrm{pt})$.



With univalence, we can show that $\mathrm{Id}_{S^1}(\mathrm{pt}, \mathrm{pt})$ is equivalent to $\mathbb{Z}$, the type of integers.

We can similarly introduce higher spheres (via iterated suspensions, in fact).

Univalence at work: isomorphic structures are equal

Pick your favourite mathematical structure, e.g. boolean algebras, topological spaces, groups, etc. For concreteness, I will pick groups here.

- We can define a type Group of groups.
- For groups, i.e. elements of this type $G, H : \mathrm{Group}$, we can consider $\mathrm{Id}_{\mathrm{Group}}(G, H)$.

Univalence at work: isomorphic structures are equal

Pick your favourite mathematical structure, e.g. boolean algebras, topological spaces, groups, etc. For concreteness, I will pick groups here.

- We can define a type Group of groups.
- For groups, i.e. elements of this type $G, H : \mathrm{Group}$, we can consider $\mathrm{Id}_{\mathrm{Group}}(G, H)$.
- It is a consequence of the univalence axiom that two groups are equal* if and only if the groups G and H are isomorphic.

  [*] i.e. we have an element of the type $\mathrm{Id}_{\mathrm{Group}}(G, H)$

Univalence at work: isomorphic structures are equal

Pick your favourite mathematical structure, e.g. boolean algebras, topological spaces, groups, etc. For concreteness, I will pick groups here.

- We can define a type Group of groups.
- For groups, i.e. elements of this type $G, H : \mathrm{Group}$, we can consider $\mathrm{Id}_{\mathrm{Group}}(G, H)$.
- It is a consequence of the univalence axiom that two groups are equal* if and only if the groups G and H are isomorphic.

  [*] i.e. we have an element of the type $\mathrm{Id}_{\mathrm{Group}}(G, H)$

- If we have an element p of $\mathrm{Id}_{\mathrm{Group}}(G, H)$, then *any* true statement about G must also be true of H, since we can transport the proof of the statement for G along p.

So with univalence, isomorphic groups **really** do behave the same.

Resources

My slides and notes

- These slides are available at https://tdejong.com/talks.html.

  They are based on lectures for a course at the *Proof and Computation* 2026 autumn school. The autumn school slides are available at https://tdejong.com/teaching.html.

- In August 2026, I taught a **5×90 min** introductory course on HoTT/UF at the European Summer School in Logic, Language and Information (ESSLLI).

  You can find the material for that bigger course at https://github.com/tomdjong/ESSLLI-2026

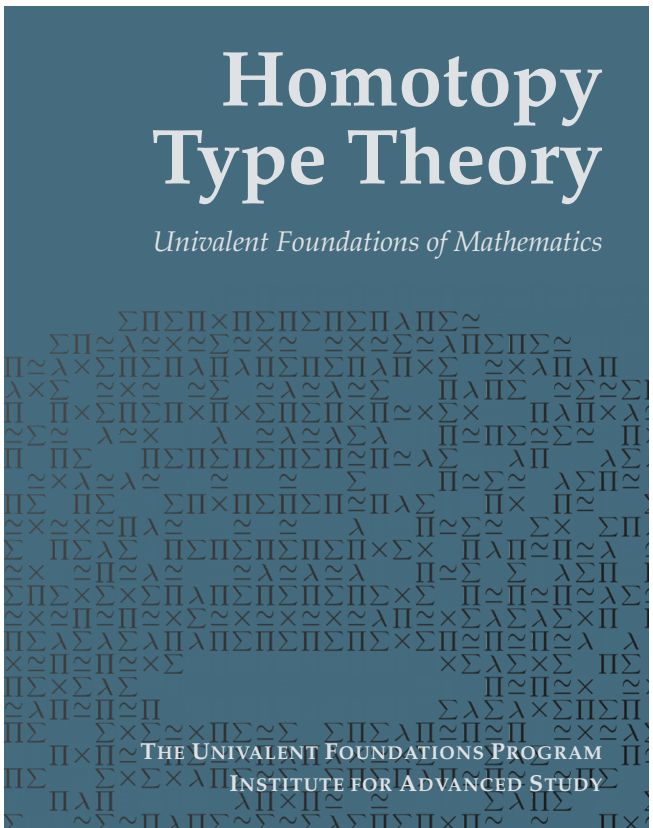
The OG resource

The **HoTT Book** was written during the *Special Year on Univalent Foundations of Mathematics* held in 2012–2013 at the Institute for Advanced Study, organized by Steve Awodey, Thierry Coquand, and Vladimir Voevodsky.

https://homotopytypetheory.org/book/

From the preface:

We did not set out to write a book. The present work has its origins in our collective attempts to develop a new style of "informal type theory" that can be read and understood by a human being, as a complement to a formal proof that can be checked by a machine.



New resources I

Introduction to Univalent Foundations of Mathematics with Agda (2019; last updated: 2025)

https://martinescardo.github.io/HoTT-UF-in-Agda-Lecture-Notes/index.html

Written *in* the proof assistant Agda!



Martín Escardó

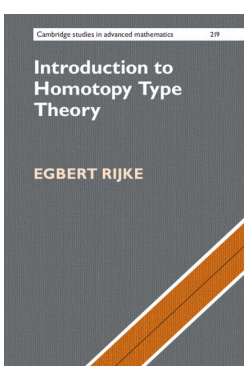
Introduction to Homotopy Type Theory (2022)

arXiv:2212.11082 (free)
Published by Cambridge University Press
doi:10.1017/9781108933568 (£50)

Largely formalized in agda-unimath.

Beginner friendly; modern perspective on HoTT.





Egbert Rijke

New resources II

HoTTEST Summer School 2022

https://hottest-seminar.github.io/HoTTESTSummerSchool.html

- The school featured two tracks: one on **Agda** and one on **HoTT**.
  - Each track had 12 lectures and as many problem sessions, all available on YouTube: https://tinyurl.com/HoTTEST-Summer-School-YT
  - The HoTT track was based on Rijke's book.
- There is an associated GitHub repository: https://github.com/martinescardo/HoTTEST-Summer-School
- You can still join the **Discord** to discuss, ask questions, etc. — https://discord.gg/tkhJ9zCGs9

New resources II

HoTTEST Summer School 2022

https://hottest-seminar.github.io/HoTTESTSummerSchool.html

- The school featured two tracks: one on **Agda** and one on **HoTT**.
  - Each track had 12 lectures and as many problem sessions, all available on YouTube: https://tinyurl.com/HoTTEST-Summer-School-YT
  - The HoTT track was based on Rijke's book.
- There is an associated GitHub repository: https://github.com/martinescardo/HoTTEST-Summer-School
- You can still join the **Discord** to discuss, ask questions, etc. — https://discord.gg/tkhJ9zCGs9

The HoTT Book covers more synthetic homotopy theory and (univalent) category theory than Rijke's book and Escardó's notes, but **no work is a subset of any of the others**.

What about cubical type theory, metatheory, semantics…?

Principles of Dependent Type Theory

Carlo Angiuli
Indiana University
cangiuli@iu.edu

Daniel Gratzer
Aarhus University
gratzer@cs.au.dk

(2026-03-21)



Daniel Gratzer



Carlo Angiuli

https://www.danielgratzer.com/papers/type-theory-book.pdf

To be published by Cambridge University Press

Metatheory and implementation	§3
Semantics	§6 (draft)
Cubical type theory	§5.3 (draft)