The UniMath Rocq Library

Niels van der Weide

UniMath

- UniMath is a library of formalized mathematics using the Rocq proof assistant
- It is based on **homotopy type theory**
- There are many results in UniMath, especially in the area of **category theory** and **bicategory theory**

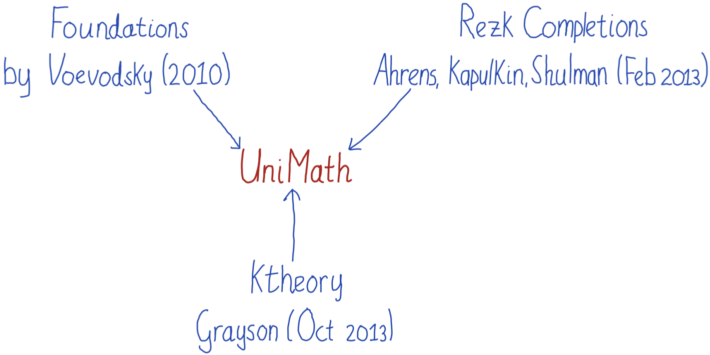
Link:

`https://github.com/UniMath/UniMath`

Table of Contents

Table of Contents

The Founders of UniMath



UniMath was founded in 2014

The UniMath Coordination Committee

The current coordination committee:

- Benedikt Ahrens
- Daniel Grayson
- Arnoud van der Leer
- Michael Lindgren
- Peter LeFanu Lumsdaine
- Ralph Matthes
- Niels van der Weide

We are responsible for maintenance and we review pull requests.

The UniMath Schools

12-2017
Birmingham

4-2019
Birmingham

7-2019
Columbus

7-2022
Cortona

7-2024
Minneapolis

The UniMath Schools

when I learned UniMath →

12-2017
Birmingham

4-2019
Birmingham

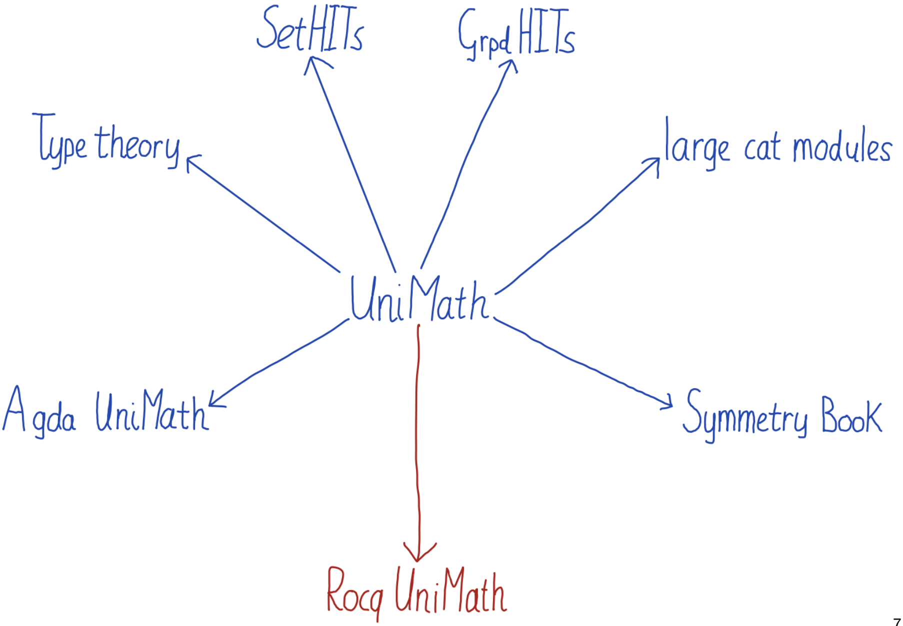
7-2019
Columbus

7-2022
Cortona

7-2024
Minneapolis

Table of Contents

The UniMath Organisation



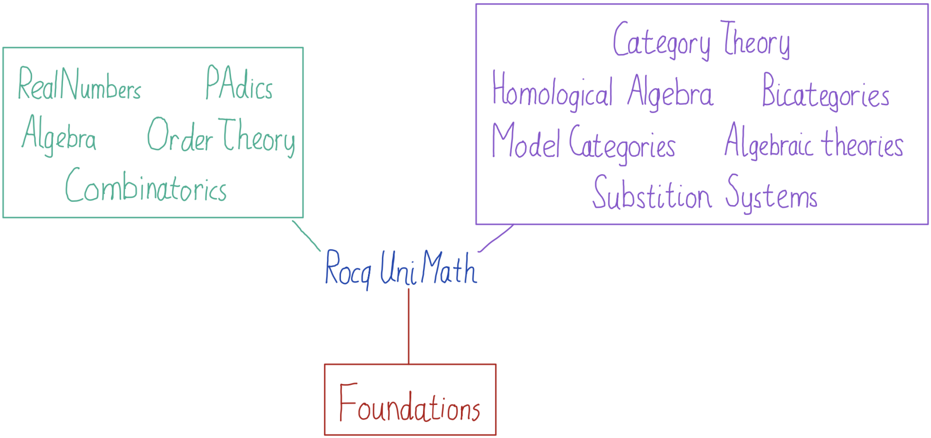
The UniMath Organisation

There are various repositories in the UniMath organisation.

- SetHITs and GrpdHITs: study of higher inductive types
- TypeTheory: semantics of type theory in homotopy type theory
- Symmetry book: studies symmetry of mathematical objects in homotopy type theory
- Large Cat Modules: study of higher order abstract syntax

Agda UniMath, which was inspired by the Symmetry Book, is another library of univalent mathematics, but written in Agda.

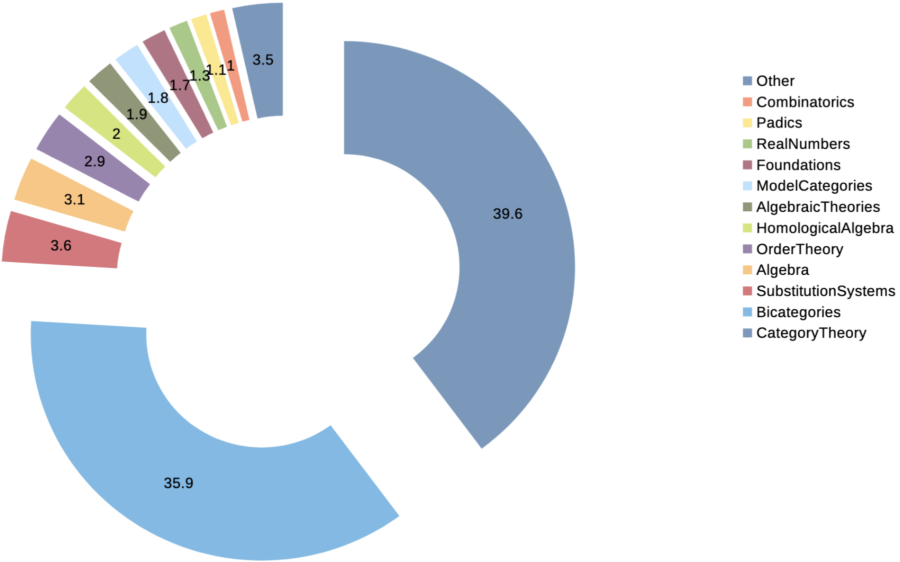
The Rocq UniMath Repository



The Rocq UniMath Repository

- There are some developments of more traditional areas of mathematics in UniMath: real numbers and p-adic numbers
- Main focus: **(higher) category theory** and applications
- Algebraic Theories: formalization of "Classical lambda calculus in modern dress"
- Substitution Systems: categorical study of syntax

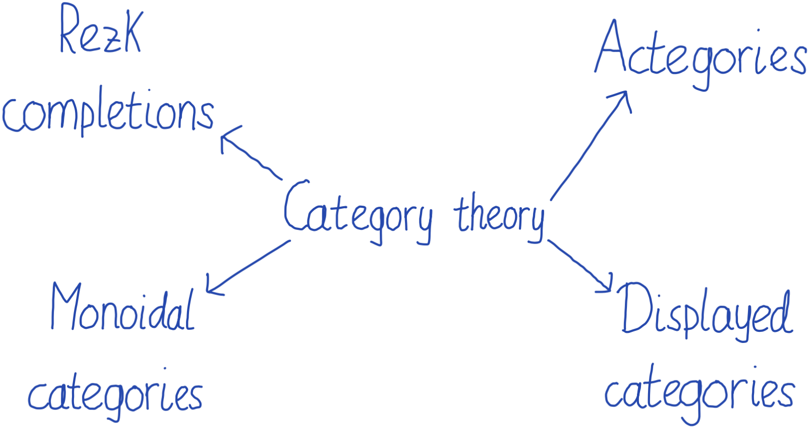
What is in UniMath?



Lines of Code

Directory	LOC
Category Theory	272827
Bicategories	247502
Substitution Systems	25234
Algebra	21917
Order Theory	20297
Homological Algebra	13848
Algebraic Theories	13116
Model Categories	12928
Foundations	11987
Real Numbers	9483
PAdics	7906
Combinatorics	6904

Category Theory



Biategory Theory

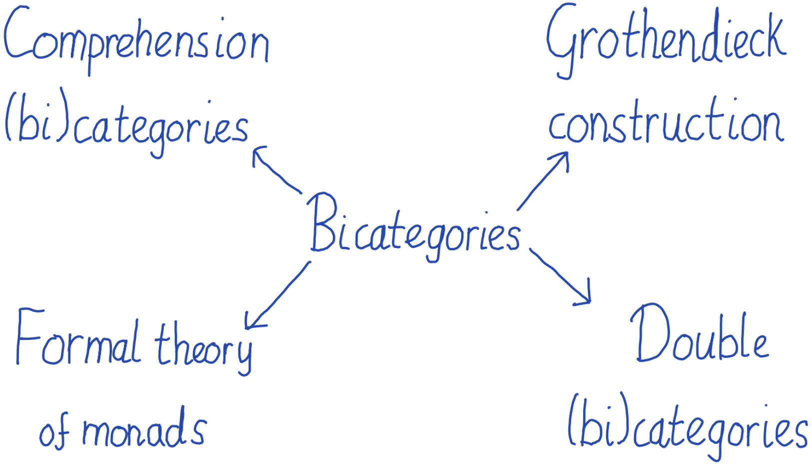


Table of Contents

Homotopy Type Theory

- **Homotopy type theory** is a foundations for mathematics
- Key feature: the **univalence axiom**, which expresses that identity corresponds to isomorphism for types
- Successful applications: synthetic homotopy theory, univalent category theory
- Homotopy type theory is available in various proof assistants

HoTT Libraries

	Rocq	Agda	Lean2
Cubical		1lab Cubical	
HoTT	RocqHoTT UniMath	Agda HoTT Type topology Agda UniMath	HoTTLean

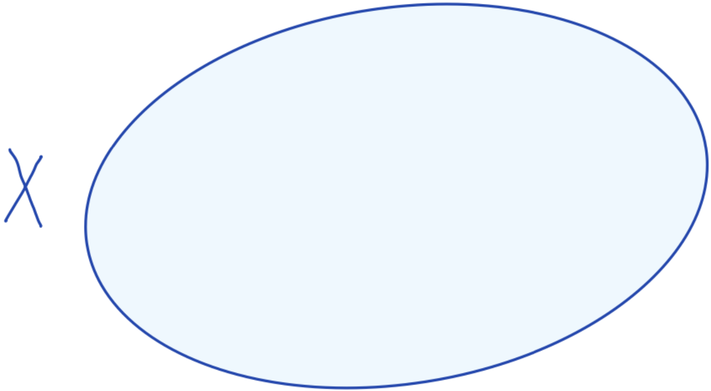
HoTT Libraries

Cubical		1lab Cubical	
HoTT	RocqHoTT UniMath	Agda HoTT Type topology Agda UniMath	HoTTLean
	Rocq	Agda	Lean2

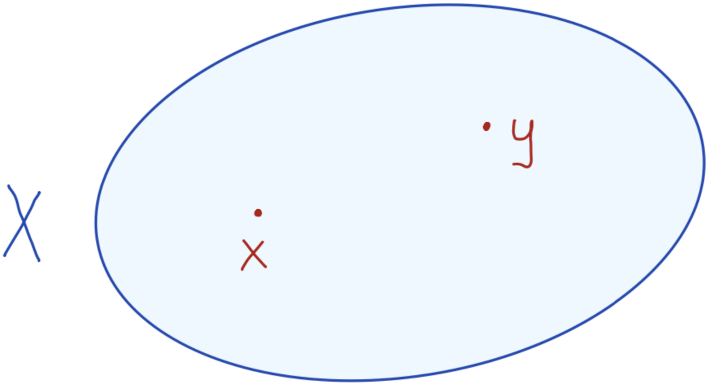
Types as Spaces

Type Theory	Homotopy Theory
Types X	Spaces X
Terms $x : X$	Points $x \in X$
$p : x =_X y$	Paths p from x to y
$h : p =_{x=y} q$	Homotopy h from p to q

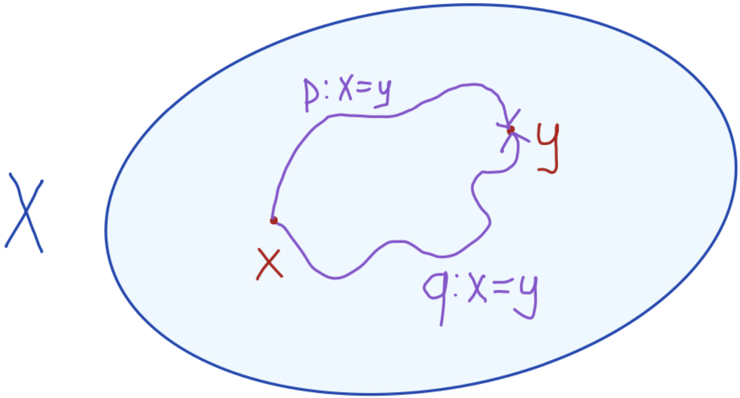
Types as Spaces



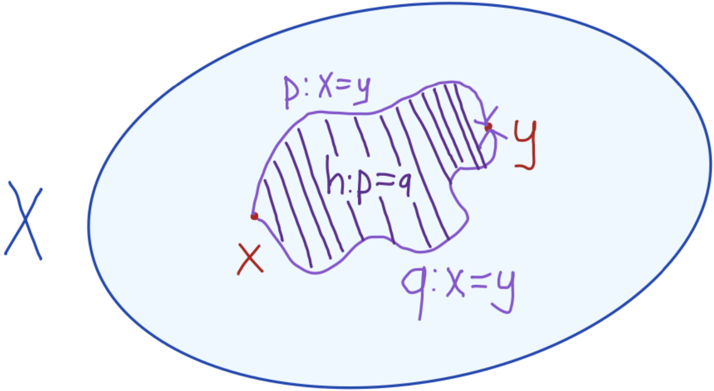
Types as Spaces



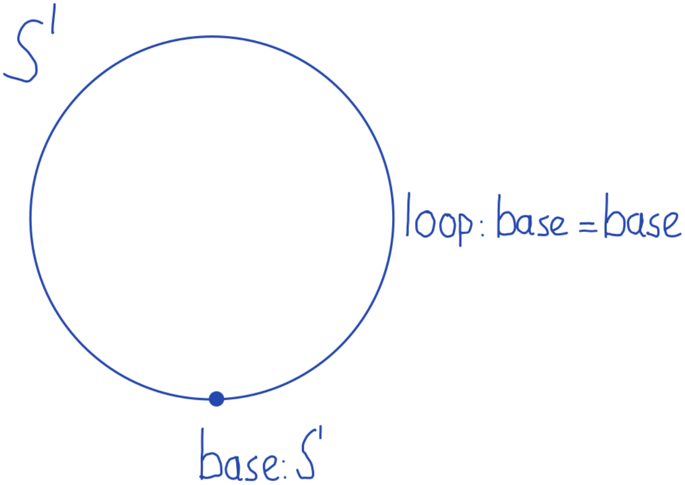
Types as Spaces



Types as Spaces



Example: The Circle



Some Observations

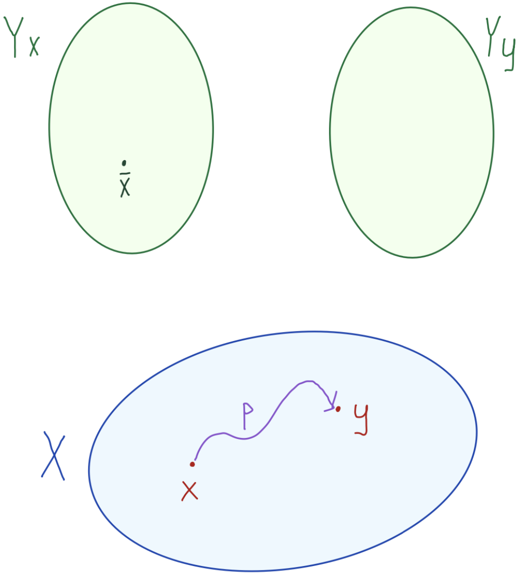
Identity is proof relevant in homotopy type theory

- Specifically, we could have $p, q : x = y$ such that $p \neq q$!
- Proofs of identity can carry more information.
- For instance, proofs $p : G = G'$ between groups G and G' are the same as isomorphism

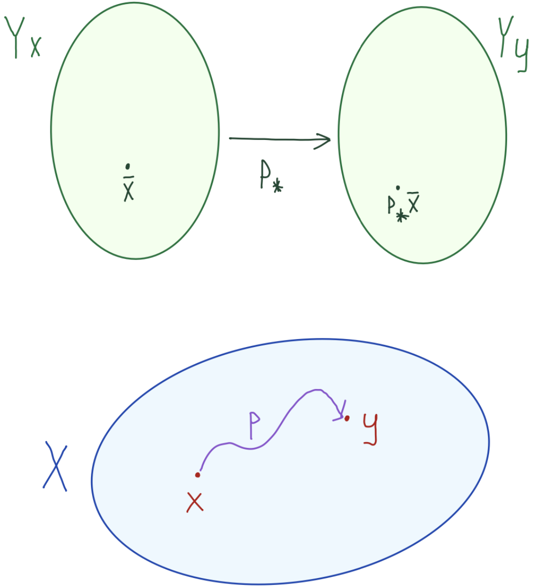
Dependent Types and Transport

Type Theory	Homotopy Theory
Types X	Spaces X
Terms $x : X$	Points $x \in X$
$p : x =_X y$	Paths p from x to y
$h : p =_{x=y} q$	Homotopy h from p to q
Dependent types	**Fibrations**

Dependent Types



Transport



The Univalence Axiom

Key feature of homotopy type theory: **the univalence axiom**, which characterizes when types are identified.

The Univalence Axiom

Key feature of homotopy type theory: **the univalence axiom**, which characterizes when types are identified.

Definition

A function $f : X \to Y$ is called an **equivalence** if for all y there is a unique x with $f(x) = y$.

The Univalence Axiom

Key feature of homotopy type theory: **the univalence axiom**, which characterizes when types are identified.

Definition

A function $f : X \to Y$ is called an **equivalence** if for all y there is a unique x with $f(x) = y$.

Proposition

For all types X and Y we have a map $\mathrm{idtoequiv}_{X,Y}$ *sending identities $X = Y$ to equivalences of types $X \simeq Y$.*

The Univalence Axiom

Key feature of homotopy type theory: **the univalence axiom**, which characterizes when types are identified.

Definition

A function $f : X \to Y$ is called an **equivalence** if for all y there is a unique x with $f(x) = y$.

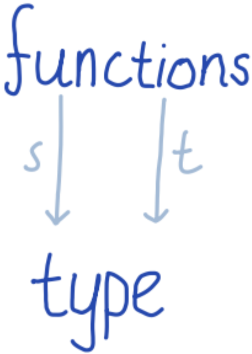
Proposition

For all types X and Y we have a map $\mathrm{idtoequiv}_{X,Y}$ *sending identities $X = Y$ to equivalences of types $X \simeq Y$.*

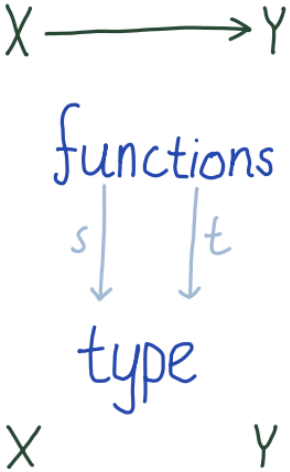
Axiom (The Univalence Axiom)

For all X and Y the map $\mathrm{idtoequiv}_{X,Y}$ *is an equivalence.*

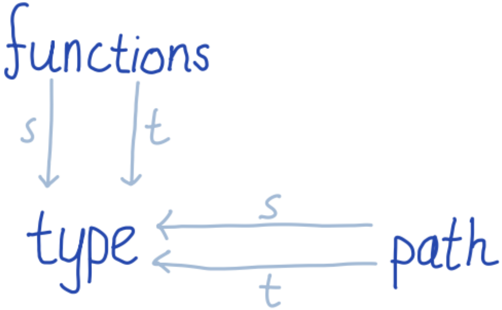
The Univalence Axiom



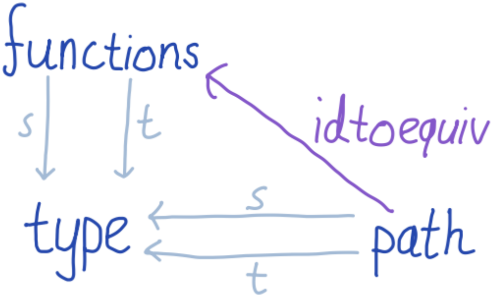
The Univalence Axiom



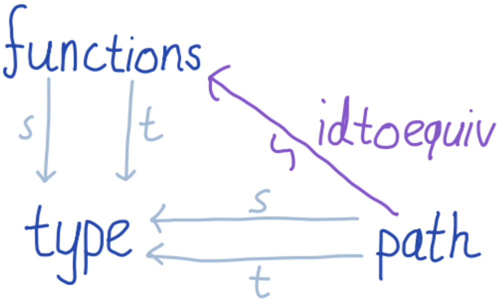
The Univalence Axiom



The Univalence Axiom



The Univalence Axiom



Consequences of the Univalence Axiom

The univalence axiom implies various **structure identity principles** (SIP)

- Identity of groups is the same as isomorphism
- Identity of rings is the same as isomorphism
- Identity of modules is the same as isomorphism

Later we discuss structure identity principles for categories

Homotopy Levels

Since identity is proof relevant, we can classify types by the "complexity" of their identity types. This leads to the notion of **homotopy level (h-level)**.

Homotopy Levels

Since identity is proof relevant, we can classify types by the "complexity" of their identity types. This leads to the notion of **homotopy level (h-level)**.

Definition

We say

- A type X is an **hSet** if for all $x, y : X$ and $p, q : x = y$ we have $p = q$.

Homotopy Levels

Since identity is proof relevant, we can classify types by the "complexity" of their identity types. This leads to the notion of **homotopy level (h-level)**.

Definition

We say

- A type X is an **hSet** if for all $x, y : X$ and $p, q : x = y$ we have $p = q$.
- A type X is a **1-type** if for all $x, y : X$ the type $x = y$ is a set.

Homotopy Levels

Since identity is proof relevant, we can classify types by the "complexity" of their identity types. This leads to the notion of **homotopy level (h-level)**.

Definition

We say

- A type X is an **hSet** if for all $x, y : X$ and $p, q : x = y$ we have $p = q$.
- A type X is a **1-type** if for all $x, y : X$ the type $x = y$ is a set. Specifically, for all points $x, y : X$, paths $p, q : x = y$, and homotopies $h_1, h_2 : p = q$, we have $h_1 = h_2$.

Homotopy Levels

Since identity is proof relevant, we can classify types by the "complexity" of their identity types. This leads to the notion of **homotopy level (h-level)**.

Definition

We say

- A type X is an **hSet** if for all $x, y : X$ and $p, q : x = y$ we have $p = q$.
- A type X is a **1-type** if for all $x, y : X$ the type $x = y$ is a set. Specifically, for all points $x, y : X$, paths $p, q : x = y$, and homotopies $h_1, h_2 : p = q$, we have $h_1 = h_2$.
- A type X is a **2-type** if for all $x, y : X$ the type $x = y$ is a 1-type.

Homotopy Levels

Since identity is proof relevant, we can classify types by the "complexity" of their identity types. This leads to the notion of **homotopy level (h-level)**.

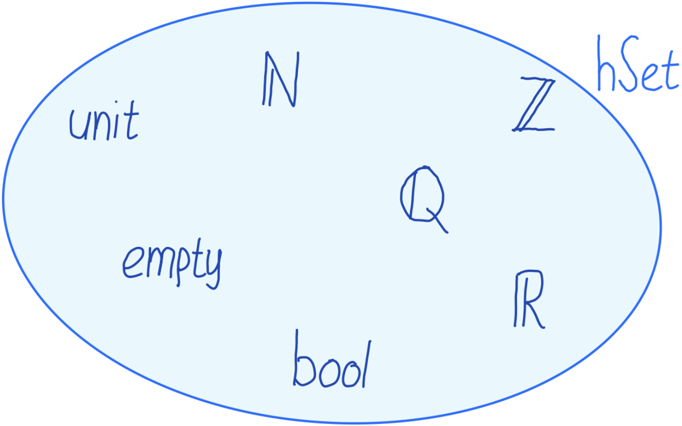
Definition

We say

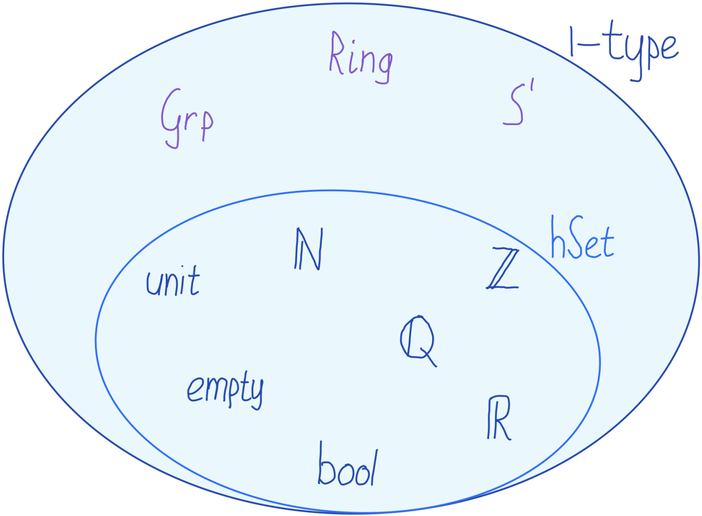
- A type X is an **hSet** if for all $x, y : X$ and $p, q : x = y$ we have $p = q$.
- A type X is a **1-type** if for all $x, y : X$ the type $x = y$ is a set. Specifically, for all points $x, y : X$, paths $p, q : x = y$, and homotopies $h_1, h_2 : p = q$, we have $h_1 = h_2$.
- A type X is a **2-type** if for all $x, y : X$ the type $x = y$ is a 1-type.

and so on

Homotopy Levels



Homotopy Levels



Homotopy Levels

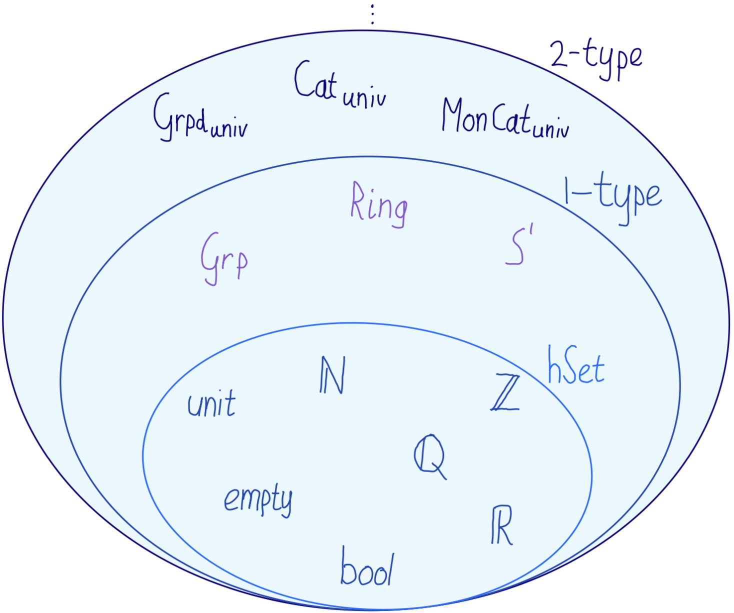


Table of Contents

Categories in Homotopy Type Theory

- In homotopy type theory, we have two notions of category: **setcategories** and **univalent categories**
- This bifurcation reflects two ways of doing category theory: either up to isomorphism or up to adjoint equivalence
- **Setcategories**: category theory up to **isomorphism**
- **Univalent categories**: category theory up to **adjoint equivalence**

Categories in Homotopy Type Theory

Definition

A **category**[1] is given by

- a **type** O of objects
- for all $x, y : O$ a **hSet** $x \to y$ of morphisms

[1] This is called "precategory" in the HoTT book

Categories in Homotopy Type Theory

Definition

A **category**[1] is given by

- a **type** O of objects
- for all $x, y : O$ a **hSet** $x \to y$ of morphisms
- for each $x : O$ an identity morphism $\mathrm{id} : x \to x$
- for each $f : x \to y$ and $g : y \to z$, a composition $f \cdot g : x \to z$

[1] This is called "precategory" in the HoTT book

Categories in Homotopy Type Theory

Definition

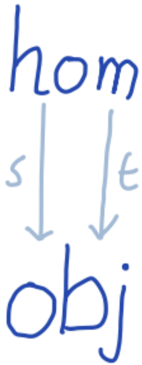
A **category**[1] is given by

- a **type** O of objects
- for all $x, y : O$ a **hSet** $x \to y$ of morphisms
- for each $x : O$ an identity morphism $\mathrm{id} : x \to x$
- for each $f : x \to y$ and $g : y \to z$, a composition $f \cdot g : x \to z$

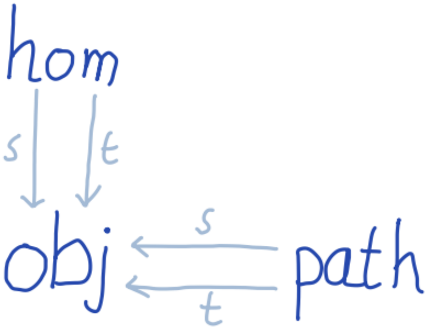
such that the usual identity and associativity laws hold.

[1] This is called "precategory" in the HoTT book

Categories in Homotopy Type Theory



Categories in Homotopy Type Theory



Note: since identity is proof relevant, the identity type of objects could be nontrivial

In the semantics, this notion does not correspond to categories

Correcting the Notion of Category

There are two ways to "correct" the notion of category

- **Setcategories**: identity on objects is trivial
- **Univalent categories**: identity on objects is determined by the morphisms

[2]This is called "strict" in the HoTT book

Correcting the Notion of Category

There are two ways to “correct” the notion of category

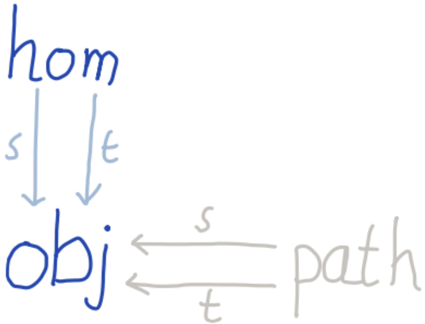
- **Setcategories**: identity on objects is trivial
- **Univalent categories**: identity on objects is determined by the morphisms

Definition

A category is called a **setcategory**[2] if its type of objects is an hSet.

[2] This is called “strict” in the HoTT book

Setcategories



Univalent Categories

Main idea: identity on objects is determined by the morphisms in a univalent category

Univalent Categories

Main idea: identity on objects is determined by the morphisms in a univalent category

Proposition

For all objects x *and* y *in a category* $\mathcal{C}$ *we have a map* $\mathrm{idtoiso}_{x,y} : x = y \to x \cong y$ *sending identities* $p : x = y$ *to isomorphisms* $\mathrm{idtoiso}_{x,y}(p) : x \cong y$.

Univalent Categories

Main idea: identity on objects is determined by the morphisms in a univalent category

Proposition

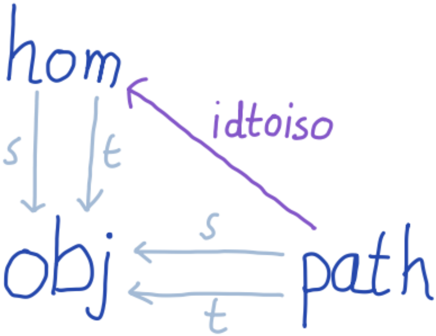
For all objects x *and* y *in a category* $\mathcal{C}$ *we have a map* $\mathsf{idtoiso}_{x,y} : x = y \to x \cong y$ *sending identities* $p : x = y$ *to isomorphisms* $\mathsf{idtoiso}_{x,y}(p) : x \cong y$*.*

Definition

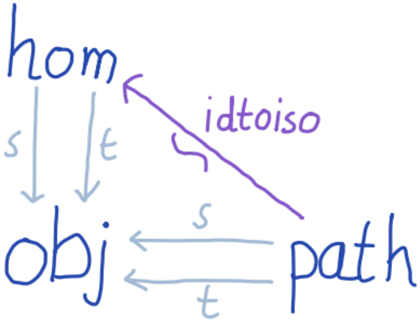
A category $\mathcal{C}$ is called **univalent** if for all $x, y : \mathcal{C}$ the map $\mathsf{idtoiso}_{x,y}$ is an equivalence of types.

So: identity on objects is the same as isomorphism.

Univalent Categories



Univalent Categories



Setcategories versus Univalent Categories

We can distinguish the notions of setcategory and of univalent category via their structure identity principles (SIP).

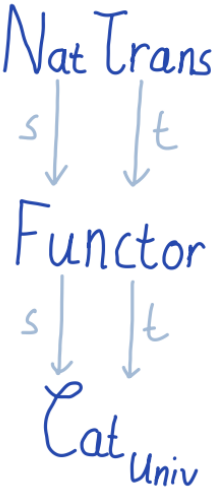
- **SIP for setcategories**: identity of setcategories corresponds to isomorphism
- **SIP for univalent categories**: identity of univalent categories corresponds to adjoint equivalence

Setcategories versus Univalent Categories

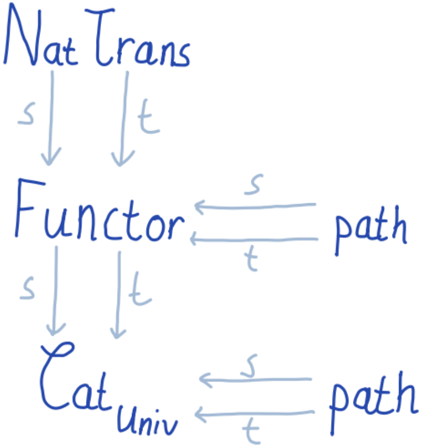
We can distinguish the notions of setcategory and of univalent category via their structure identity principles (SIP).

- **SIP for setcategories**: identity of setcategories corresponds to isomorphism
- **SIP for univalent categories**: identity of univalent categories corresponds to adjoint equivalence
- **SIP for functors between univalent categories**: identity of such functors corresponds to natural isomorphism

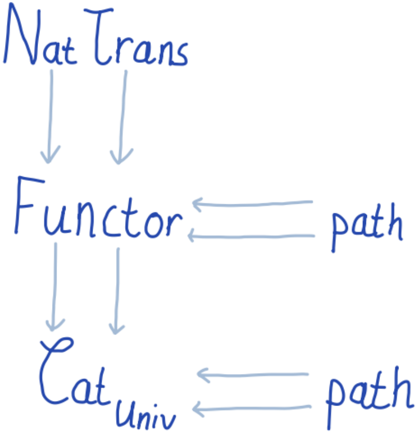
The Univalence Principle for Univalent Categories



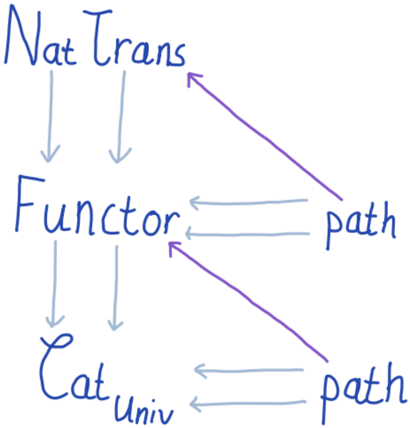
The Univalence Principle for Univalent Categories



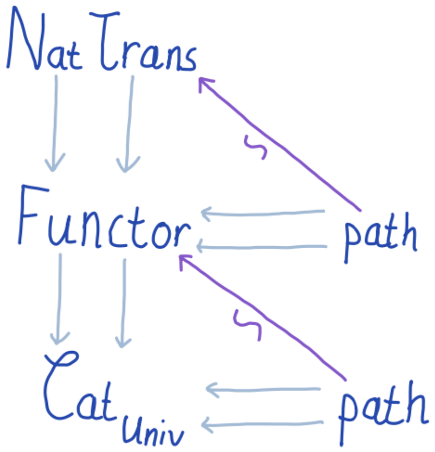
The Univalence Principle for Univalent Categories



The Univalence Principle for Univalent Categories



The Univalence Principle for Univalent Categories



Consequences of the Univalence Principle

- Univalence allows us to treat adjoint equivalences as identities, which allows us to do **equivalence induction**.
- Specifically, to prove a statement

$$\forall(\mathcal{C}_1, \mathcal{C}_2 : \mathsf{Cat}_{\mathsf{univ}})(e : \mathcal{C}_1 \simeq \mathcal{C}_2), P(\mathcal{C}_1, \mathcal{C}_1, e)$$

it suffices to prove

$$\forall(\mathcal{C} : \mathsf{Cat}_{\mathsf{univ}}), P(\mathcal{C}, \mathcal{C}, \mathsf{id})$$

Benefits of the Univalence Principle I

Equivalence induction is useful for various applications, such as **transporting properties/structure along adjoint equivalences**.

Benefits of the Univalence Principle I

Equivalence induction is useful for various applications, such as **transporting properties/structure along adjoint equivalences**.

- For instance, one might want to prove that if $\mathcal{C}_1$ is locally Cartesian closed and $e : \mathcal{C}_1 \simeq \mathcal{C}_2$, then $\mathcal{C}_2$ is locally Cartesian closed.
- A manual proof is quite technical.
- With univalence: trivial

Benefits of the Univalence Principle II

Another application of equivalence induction is **characterizing adjoint equivalences**.

Benefits of the Univalence Principle II

Another application of equivalence induction is **characterizing adjoint equivalences**.

For instance, one might want to prove

- A pseudotransformation is an adjoint equivalence if it is a pointwise adjoint equivalence
- There are similar statements for double categories and comprehension categories

Benefits of the Univalence Principle II

Another application of equivalence induction is **characterizing adjoint equivalences**.

For instance, one might want to prove

- A pseudotransformation is an adjoint equivalence if it is a pointwise adjoint equivalence
- There are similar statements for double categories and comprehension categories

There's not enough time to discuss this in some detail.

The main idea:

- Equivalences of such structures are built up from equivalences of simpler structures.
- Equivalence induction allows us to treat equivalences of simpler structures as identities, which simplifies calculational proofs

Table of Contents

Conclusion

- UniMath is library in Rocq based on **homotopy type theory**, with a particular focus on (higher) category theory
- Homotopy type theory is **advantageous for the formalization of category theory**, and it simplifies various proofs
- Check out

  `https://github.com/UniMath/UniMath`