Univalent categories and the Rezk completion

Benedikt Ahrens

joint work with Krzysztof Kapulkin and Michael Shulman

Institut de **R**echerche en **I**nformatique de **T**oulouse

ACADIE

2013–11–21

3 kinds of sameness for categories

Equality	$\mathcal{C} = \mathcal{D}$
Isomorphism	$\mathcal{C} \cong \mathcal{D}$
Equivalence	$\mathcal{C} \simeq \mathcal{D}$

- most properties of categories invariant under **equivalence**
- we can only substitute **equals for equals**
- in set-theoretic foundations these notions are worlds apart

In this talk:

Define categories in the **Univalent Foundations** for which all three coincide

Outline

1. Introduction to Univalent Foundations
   - Type theory and its homotopy interpretation
   - Logic in type theory: homotopy levels
   - The Univalence Axiom
2. Category Theory in Univalent Foundations
   - Categories: basic definitions
   - Univalent categories: definition & some properties
   - The Rezk completion

Table of Contents

1. Introduction to Univalent Foundations
   - Type theory and its homotopy interpretation
   - Logic in type theory: homotopy levels
   - The Univalence Axiom
2. Category Theory in Univalent Foundations
   - Categories: basic definitions
   - Univalent categories: definition & some properties
   - The Rezk completion

Univalent Foundations

What are the Univalent Foundations?

- Intensional Martin-Löf Type Theory

$\rightsquigarrow$ *Types as Spaces* interpretation, i.e. Homotopy Type Theory

$+$ **Univalence Axiom**

The 4 kinds of judgments of type theory

Contexts & judgements

Γ	sequence of variable declarations $x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(\vec{x}_i)$
$\Gamma \vdash A$	A is well–formed **type** in context Γ
$\Gamma \vdash a : A$	**term** a is of type A in context Γ
$\Gamma \vdash A \equiv B$	types A and B are **convertible**
$\Gamma \vdash a \equiv b : A$	a is convertible to b in type A

In particular: dependent type B over A

$$x : A \vdash B(x)$$

Conventions for contexts and judgments

Reasoning in type theory

- means deducing judgments from judgments,
- according to **inference rules**.

Conventions: We

- omit leading Γ in $\Gamma, (x : A) \vdash B(x)$
- omit leading $\vdash$ when context is empty
- handle context casually: “for (any) $x : A \ldots$”
- say “if … then …” for describing inference rules

How to do mathematics in type theory?

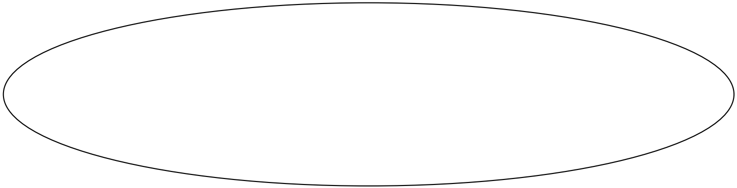
Math. activity	how to do it in type theory
define a class of objects	give a name to a valid type
define a property	give a name to a valid type
define a specific object	give a name to a valid term
construct an object	construct a term of the defining type
prove a property	construct a term of the defining type
	(all relative to some context)

What do types represent?

- Traditionally, types were considered to represent **sets.**
- In Homotopy Type Theory, types are modelled by **spaces**:

What do types represent?

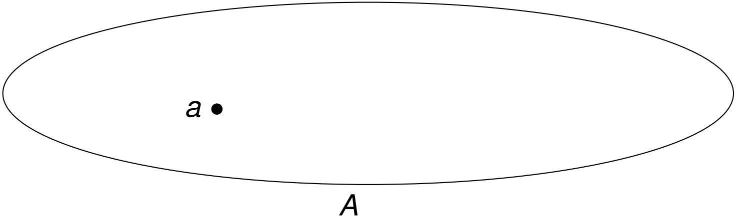
- Traditionally, types were considered to represent **sets.**
- In Homotopy Type Theory, types are modelled by **spaces**:
  - $\emptyset \vdash A$



A

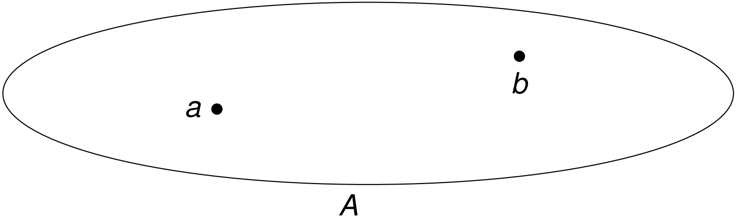
What do types represent?

- Traditionally, types were considered to represent **sets.**
- In Homotopy Type Theory, types are modelled by **spaces**:
  - $\emptyset \vdash A$
  - $\emptyset \vdash a : A$



What do types represent?

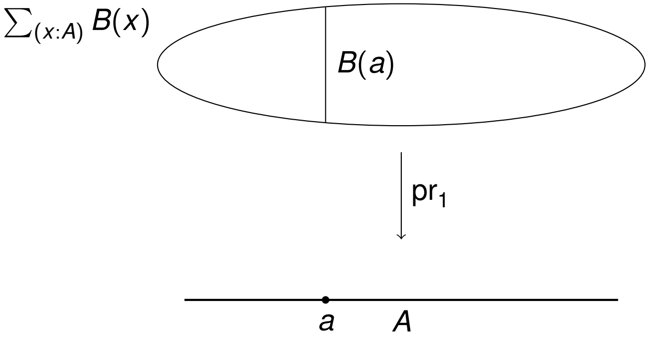
- Traditionally, types were considered to represent **sets.**
- In Homotopy Type Theory, types are modelled by **spaces**:
  - $\emptyset \vdash A$
  - $\emptyset \vdash a : A$
  - $\emptyset \vdash b : A$



Interpretation of dependent type

Interpret the type family $x : A \vdash B(x)$

as a **fibration**, ie. as the projection from the total space $\sum_{(x:A)} B(x)$ to the indexing space A



Introducing new concept = introducing new type

A type is specified by 4 inference rules:

1. **Type former:** declaring a new type
2. **Term former:** way to construct terms of this type
3. **Elimination:** way to use terms of type 1 to construct other terms
4. **Computation:** what if 2 followed by 3

Example (Function types)

1. if A and B are types, then $A \to B$ is a type
2. if $\quad \Gamma, (x : A) \vdash b(x) : B \quad$ then $\quad \Gamma \vdash \lambda x.b(x) : A \to B$
3. if $f : A \to B$ and $a : A$, then $f@a : B$
4. $\lambda x.b(x)@a \quad \equiv \quad b[x := a]$

Example: dependent sum

Dependent sum $\sum_{x:A} B(x)$

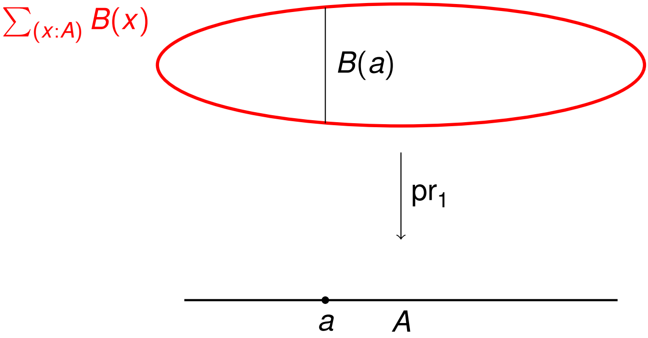
Corresponds to the total space $\sum_{(x:A)} B(x)$ of a fibration:

1. if $\Gamma, (x : A) \vdash B(x)$ then $\Gamma \vdash \sum_{x:A} B(x)$ is a type
2. if $a : A$ and $b : B(a)$ then $(a, b) : \sum_{x:A} B(x)$
3. if $p : \sum_{x:A} B(x)$ then $\mathit{fst}(p) : A$ and $\mathit{snd}(p) : B(\mathit{fst}(p))$
4. $\mathit{fst}(a, b) \equiv a$ and $\mathit{snd}(a, b) \equiv b$

Remark

If B does not depend on x in 1 , we obtain $A \times B$.

Interpretation of Σ-types



Example: Dependent Product

1. if $\Gamma, x : A \vdash B(x)$ then $\Gamma \vdash \prod_{x:A} B(x)$
2. if $x : A \vdash b(x) : B(x)$ then $\vdash \lambda x.b(x) : \prod_{x:A} B(x)$
3. if $f : \prod_{x:A} B(x)$ and $a : A$ then $f@a : B(a)$
4. $\lambda x.b(x)@a \quad \equiv \quad b[x := a]$

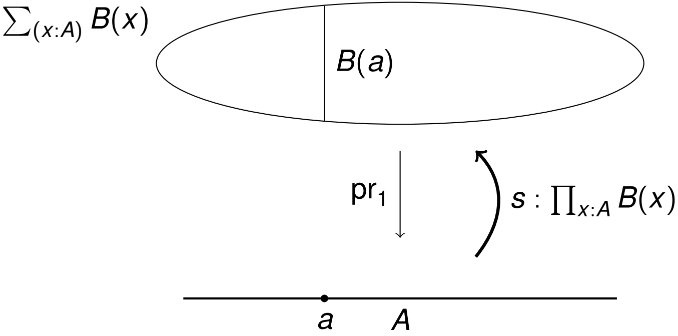
Remark

- If B does not depend on x in 1, we obtain $A \to B$.
- We do not distinguish between constructing
  - a term $f : \prod_{x:A} B(x)$
  - a term $b(x) : B(x)$ in context $x : A$

Interpretation of dependent product

Interpret the dependent product $\prod_{x:A} B(x)$

as the space of **sections** from A to the total space $\sum_{(x:A)} B(x)$



Martin-Löf TT and its Homotopy Interpretation

Type theory	Notation	Interpretation
Inhabitant	$a : A$	a is a point in space A
Dependent type	$x : A \vdash B(x)$	fibration $\sum_{(x:A)} B(x) \to A$
Sigma type	$\sum_{x:A} B(x)$	total space of a fibration
Product type	$\prod_{x:A} B(x)$	space of sections of a fibration
Coproduct type	$A + B$	disjoint union
Identity type	$\mathrm{Id}_A(a, b)$	space of **paths** $p : a \rightsquigarrow b$

- other types as needed (type $\mathbf{N}$ of naturals, empty type)

Rules of the identity type

1. $(x, y) : A \times A \vdash \mathrm{Id}_A(x, y)$
2. if $\quad a : A \quad$ then $\quad \mathrm{refl}(a) : \mathrm{Id}_A(a, a)$
3. $$\frac{(x, y : A)(p : \mathrm{Id}(x, y)) \vdash C(x, y, p) \qquad x : A \vdash c(x) : C(x, x, \mathrm{refl}(x}{(x, y : A), (p : \mathrm{Id}(x, y)) \vdash J(c, x, y, p) : C(x, y, p)}$$
4. $J(c, a, a, \mathrm{refl}(a)) \equiv c(a)$

The Identity elimination rule 3 says:

To define a function of type

$$\prod_{(x,y:A)} \prod_{(p:\mathrm{Id}(x,y))} C(x, y, p)$$

it suffices to specify its image on $(x, x, \mathrm{refl}(x))$.

Leibniz principle

Using ③ , one can construct:

Leibniz principle

Given a dependent type $x : A \vdash C(x)$ and $a, b : A$, a function

$$\mathsf{subst}_{a,b} : \mathsf{Id}(a, b) \to \big(C(a) \to C(b)\big)$$
$$\mathsf{subst}_{a,a}(\mathsf{refl}(a)) := (t \mapsto t)$$

Leibniz principle says:

If there is $p : \mathsf{Id}_A(a, b)$, then no type $x : A \vdash C(x)$ can "distinguish" a and b.

Set-theoretic interpretation of Id type

Using ❸ , one can construct terms of the following types:

"Setoid" structure

$$\begin{aligned} \mathrm{refl}(x) &: \mathrm{Id}_A(x,x) \\ (\_)^{-1} &: \mathrm{Id}_A(x,y) \to \mathrm{Id}_A(y,x) \\ \_ \star \_ &: \mathrm{Id}_A(x,y) \to \mathrm{Id}_A(y,z) \to \mathrm{Id}_A(x,z) \end{aligned}$$

Set-theoretic interpretation of $p : \mathrm{Id}_A(a,b)$

- a and b are interpreted as being equal in A
- justified by Leibniz principle and "setoid" structure
- in this model **only existence of** $p : \mathrm{Id}(a,b)$ **matters**

But terms of Id type have an interesting structure of their own!

Id types are not trivial

Higher identity types

There is also an identity type for each pair of identity terms

$$p, q : \mathrm{Id}_A(x, y) \vdash \mathrm{Id}_{\mathrm{Id}(x,y)}(p, q)$$

But what higher identity terms can we construct?

Theorem (Hofmann & Streicher '95)

*Given a type A, one can **not** construct a term of type*

$$\prod_{x:A, p:\mathrm{Id}(x,x)} \mathrm{Id}_{\mathrm{Id}(x,x)}(p, \mathrm{refl}(x))$$

The higher groupoid structure of Id types

Higher Groupoid laws hold: one can construct terms of type

- $\mathsf{Id}_{\mathsf{Id}(x,x)}(p \star p^{-1}, \mathsf{refl}(x))$ $\qquad$ $\mathsf{Id}_{\mathsf{Id}(x,x)}(p^{-1} \star p, \mathsf{refl}(x))$
- $\mathsf{Id}_{\mathsf{Id}(x,y)}(p \star \mathsf{refl}(y), p)$ $\qquad$ $\mathsf{Id}_{\mathsf{Id}(x,y)}(\mathsf{refl}(x) \star p, p)$
- associativity up to higher Id term

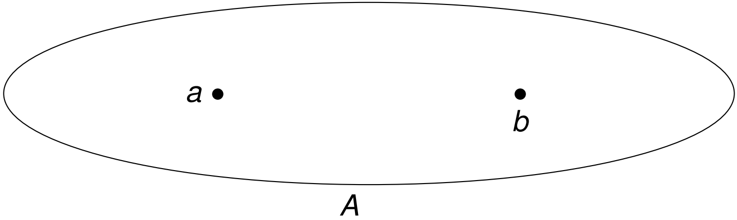
In general, Id terms of height n satisfy groupoid laws wrt Id terms of height $n+1$:

Theorem (Lumsdaine, Garner & van den Berg)

The terms belonging to the iterated identity types of any type A form an ∞-groupoid.

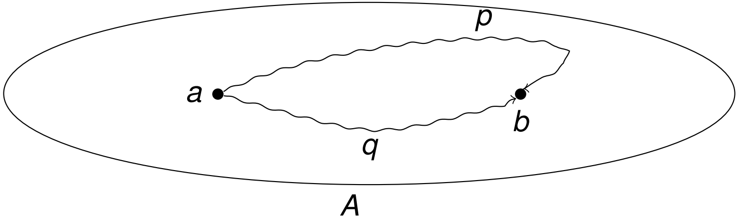
Interpretation: identity type as path space

- For two terms $a\ b : A$ of a type A, there is a type $\mathrm{Id}(a, b)$



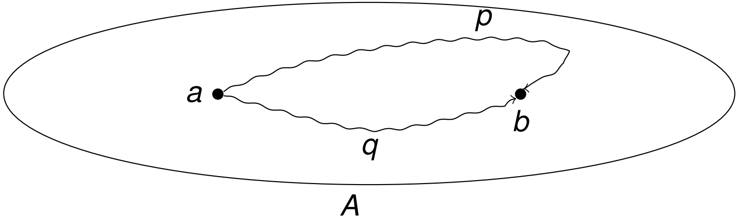
Interpretation: identity type as path space

- For two terms $a\ b : A$ of a type A, there is a type $\mathrm{Id}(a, b)$
- terms $p, q : \mathrm{Id}(a, b)$ are interpreted as paths $p, q : a \rightsquigarrow b$



Interpretation: identity type as path space

- For two terms $a\ b : A$ of a type A, there is a type $\mathrm{Id}(a, b)$
- terms $p, q : \mathrm{Id}(a, b)$ are interpreted as paths $p, q : a \rightsquigarrow b$



Mixing syntax and semantics

- Call a term $p : \mathrm{Id}(a, b)$ a “path from a to b”, write $p : a \rightsquigarrow b$
- Say a and b are **homotopic** if there is a path $p : a \rightsquigarrow b$.

The homotopy interpretation of identity types

Interpretation of the operations on paths:

Type theory	Interpretation	Notation
refl	constant path on a	$\mathsf{refl}(a)$
inverse	path reversal	p^{-1}
concat	path concatenation	$p \star q$
higher identity type	paths between paths	$p \rightsquigarrow q$
	"continuous deformations"	

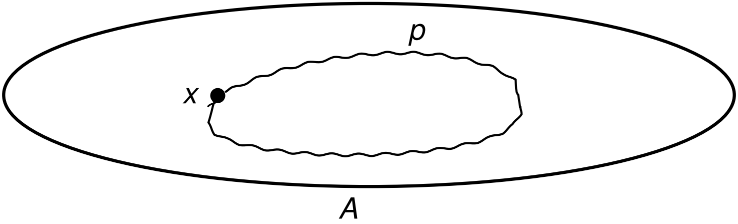
Non-trivial loop spaces

Interpretation of Hofmann & Streicher's theorem

Given a type A, one can **not** construct a term of type

$$\prod_{x:A,p:\mathrm{Id}(x,x)} \mathrm{Id}_{\mathrm{Id}(x,x)}(p, \mathrm{refl}(x))$$

Ie. it is (equi-)consistent to have a type A



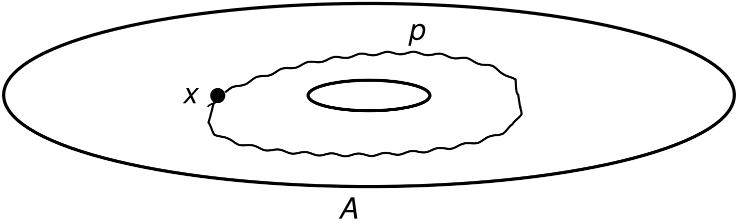
Non-trivial loop spaces

Interpretation of Hofmann & Streicher's theorem

Given a type A, one can **not** construct a term of type

$$\prod_{x:A,p:\mathrm{Id}(x,x)} \mathrm{Id}_{\mathrm{Id}(x,x)}(p, \mathrm{refl}(x))$$

Ie. it is (equi-)consistent to have a type A with non-trivial path spaces.



Summary: homotopy is **not** equality

Homotopy is not like (set-theoretic) equality

- paths, unlike equality proofs, are mathematical objects
- we care about **how** two points are homotopic

However, homotopy has some properties of equality:

Homotopy is a **proof-relevant** equality in type theory

- the substitution principle
- higher groupoidal operations: refl, inverse, concatenation

- we use vocabulary of equality (“equal”, “unique”)
- but are aware of the differences with set-theoretic equality

A model of MLTT in simplicial sets

Types-as-spaces intuition is made precise by a model of MLTT:

- The category sSET of simplicial sets is Quillen-equivalent to the category TOP of topological spaces.
- There is a model of MLTT in simplicial sets [Voevodsky].
- This model satisfies an additional property: **univalence**
- This suggests adding univalence as an additional axiom (UA) to MLTT.

Remark

Traditional set-theoretic models of MLTT do not satisfy univalence and thus are not models of MLTT + UA.

Table of Contents

1 Introduction to Univalent Foundations
Type theory and its homotopy interpretation
Logic in type theory: homotopy levels
The Univalence Axiom

2 Category Theory in Univalent Foundations
Categories: basic definitions
Univalent categories: definition & some properties
The Rezk completion

Type theory vs. set theory

Set theory	Type theory
Logic	Types
$\wedge, \vee, \Rightarrow, \neg, \forall, \exists$	$\times, +, \to, \prod, \sum$
Sets	Logic
$\times, +, \to, \prod, \sum$	$\wedge, \vee, \Rightarrow, \neg, \forall, \exists$
$x \in A$ is a proposition	$x : A$ is a typing judgment

Propositions as some types

- In set theory, propositions and sets are separate entities.
- In type theory, **propositions are specific types.**

Definition (Proposition)

A type A is a **proposition** if all its inhabitants are homotopic, ie. if one can construct a term of type

$$\mathrm{isProp}(A) := \prod_{x:A} \prod_{y:A} \mathrm{Id}_A(x, y) \ .$$

Remarks about propositions

- Proving a proposition P means constructing a term $p : P$.
- $p : P$ is called a **proof** of the proposition P.
- "Being a proposition" is a proposition, ie. one can prove

$$\mathrm{isProp}(\mathrm{isProp}(A))$$

- Intuitively, a proposition is either empty or a singleton.
- All operations on types are available for propositions: they correspond to **logical** operations via the **Curry-Howard isomorphism**

Curry-Howard

Logic is **embedded** in type theory via Curry–Howard

- proving $P \Rightarrow Q$ amounts to giving a function $P \rightarrow Q$
- proving $\forall x : A.P(x)$ amounts to constructing a function

$$\lambda x : A.p(x) : \prod_{x:A} P(x)$$

- proving $\exists x : A.P(x)$ amounts to constructing a pair

$$\big(a, p(a)\big) : \sum_{x:A} P(x)$$

! Some more work is actually required for $\exists$, since propositions are not sufficiently closed under $\sum$.

Sets in Univalent Foundations

Definition (Sets)

A type A is a **set** if for any $x, y : A$, the type $\mathrm{Id}(x, y)$ is a proposition:

$$\mathrm{isSet}(A) := \prod_{x\ y:A} \mathrm{isProp}(\mathrm{Id}(x, y))$$

- Points of a set are equal in a unique way, if they are.
- Sets correspond to **discrete spaces**.

About the use of the word “unique”

Definition

We call the point $a : A$ **unique** if any point $x : A$ is **homotopic** to a, ie. if we can construct a term of type

$$\prod_{x:A} \mathrm{Id}(x, a)$$

A type A with a unique point $a : A$ is called “contractible”:

Definition

We call A **contractible** if we can construct a term of type

$$\mathrm{isContr}(A) := \sum_{(a:A)} \prod_{(x:A)} \mathrm{Id}(x, a)$$

Homotopy levels

Homotopy levels: the complete picture

$$\mathrm{isContr}(A) :\equiv \sum_{(a:A)} \prod_{(x:A)} \mathrm{Id}(x, a)$$

$$\mathrm{isProp}(A) :\equiv \prod_{x,y:A} \mathrm{isContr}(\mathrm{Id}(x, y))$$

$$\mathrm{isSet}(A) :\equiv \prod_{x,y:A} \mathrm{isProp}(\mathrm{Id}(x, y))$$

$$\vdots$$

$$\mathrm{isofhlevel}_{n+1}(A) :\equiv \prod_{x,y:A} \mathrm{isofhlevel}_n(\mathrm{Id}(x, y))$$

But we will not need the higher levels.

Table of Contents

1 Introduction to Univalent Foundations
- Type theory and its homotopy interpretation
- Logic in type theory: homotopy levels
- The Univalence Axiom

2 Category Theory in Univalent Foundations
- Categories: basic definitions
- Univalent categories: definition & some properties
- The Rezk completion

Dependent types as maps to a universe

Types are stratified in universes

We suppose

- having a sequence of **universes** $(\mathcal{U}_n)_{n\in\mathbb{N}}$ (à la Russell)
- any type A is a point of some universe $A : \mathcal{U}_n$

Implicit universe polymorphism: omit the index n

A dependent type $x : A \vdash B(x)$

is a map $B : A \to \mathcal{U}$.

Univalence : isomorphic types are equal

The universe $\mathcal{U}$ is a type

- thus can consider $\mathrm{Id}_{\mathcal{U}}(A, B)$
- but no way to construct non-trivial path $A \rightsquigarrow B$

Univalence: paths are isomorphisms

- Idea: any path $p : A \rightsquigarrow B$ corresponds to an isomorphism $f : A \to B$
- impose this correspondance as an axiom
- can construct isomorphism $f : A \to B$ for suitable A and B

Isomorphism of types

Definition (Isomorphism of types)

A function $f : A \to B$ is an **isomorphism of types** if there are

- $$g : B \to A$$
- $$\eta : \prod_{a:A} \mathsf{Id}\Big(g(f(a)), a\Big) \qquad \epsilon : \prod_{b:B} \mathsf{Id}\Big(f(g(b)), b\Big)$$

together with a coherence condition $\tau : \prod_{x:A} \mathsf{Id}\Big(f(\eta x), \epsilon(fx)\Big)$

…ie. if we can construct a term of type

$$\mathsf{isIso}(f) := \sum_{(g:B\to A)} \sum_{(\eta:\_)} \sum_{(\epsilon:\_)} \prod_{(x:A)} \mathsf{Id}\Big(f(\eta x), \epsilon(fx)\Big)$$

Isomorphism of types II

Lemma

*For any $f : A \to B$, the type $\mathsf{isIso}(f)$ is a proposition. In particular, the inverse g is **unique**, if it exists.*

Definition (Type of isomorphisms from A to B)

$$\mathsf{Iso}(A, B) := \sum_{f : A \to B} \mathsf{isIso}(f)$$

Example (Leibniz principle)

For any $p : \mathsf{Id}(a, b)$, the substitution function

$$\mathsf{subst}_{a,b}(p) : C(a) \to C(b)$$

is an isomorphism with inverse $\mathsf{subst}_{b,a}(p^{-1})$.

The Univalence Axiom

Definition (From paths to isomorphisms)

$$\begin{aligned} \mathsf{idtoiso}_{A,B} : \mathsf{Id}(A,B) &\to \mathsf{Iso}(A,B) \\ \mathsf{refl}(A) &\mapsto (x \mapsto x, p) \end{aligned}$$

Univalence Axiom

$$\mathsf{univalence} : \prod_{A\ B:\mathcal{U}} \mathsf{isIso}(\mathsf{idtoiso}_{A,B})$$

In particular, Univalence gives a map backwards:

$$\mathsf{isotoid}_{A,B} : \mathsf{Iso}(A,B) \to \mathsf{Id}(A,B)$$

Consequences of Univalence

- Propositional extensionality

$$(P \leftrightarrow Q) \to \mathsf{Id}(P, Q)$$

- Function extensionality:

$$\prod_{x:A} \mathsf{Id}_B(fx, gx) \to \mathsf{Id}_{A \to B}(f, g)$$

  and its dependent variant

- Quotient types exist (cf. later)

Table of Contents

1 Introduction to Univalent Foundations
Type theory and its homotopy interpretation
Logic in type theory: homotopy levels
The Univalence Axiom

2 Category Theory in Univalent Foundations
Categories: basic definitions
Univalent categories: definition & some properties
The Rezk completion

Categories in Univalent Foundations — Take I

A naïve definition of categories

A **category** $\mathcal{C}$ is given by

- a type $\mathcal{C}_0$ of **objects**
- for any $a, b : \mathcal{C}_0$, a type $\mathcal{C}(a, b)$ of **morphisms**
- operations: identity & composition

$$\mathrm{id} : \prod_{a:\mathcal{C}_0} \mathcal{C}(a,a) \qquad (\circ) : \prod_{a,b,c:\mathcal{C}_0} \mathcal{C}(b,c) \times \mathcal{C}(a,b) \to \mathcal{C}(a,c)$$

- axioms: unitality & associativity for any suitable f, g, h:

$$\text{unital} : \prod_{a,b:\mathcal{C}_0, f:\mathcal{C}(a,b)} (\mathrm{id}_b \circ f \rightsquigarrow f) \times (f \circ \mathrm{id}_a \rightsquigarrow f)$$

$$\text{assoc} : \prod_{a,b,c,d,f,g,h} (h \circ g) \circ f \rightsquigarrow h \circ (g \circ f)$$

Coherence for associativity – Mac Lane's pentagon

Problem with above definition: two ways to associate a composition of **four** morphisms from left to right:

$$
\begin{tikzcd}
 & (i \circ h) \circ (g \circ f) \arrow[ddr, squiggly] & \\
((i \circ h) \circ g) \circ f \arrow[ur, squiggly] \arrow[dd, squiggly] & & \\
 & & i \circ (h \circ (g \circ f)) \\
(i \circ (h \circ g)) \circ f \arrow[dr, squiggly] & & \\
 & i \circ ((h \circ g) \circ f) \arrow[uur, squiggly] &
\end{tikzcd}
$$

Coherence for associativity – Mac Lane's pentagon

Problem with above definition: two ways to associate a composition of **four** morphisms from left to right:

$$
\begin{tikzcd}
 & (i \circ h) \circ (g \circ f) \arrow[dr, rightsquigarrow] \arrow[ddd, phantom, "\Downarrow"] & \\
((i \circ h) \circ g) \circ f \arrow[ur, rightsquigarrow] \arrow[d, rightsquigarrow] & & i \circ (h \circ (g \circ f)) \\
(i \circ (h \circ g)) \circ f \arrow[dr, rightsquigarrow] & & \\
 & i \circ ((h \circ g) \circ f) \arrow[uur, rightsquigarrow] &
\end{tikzcd}
$$

Would need to ask for higher coherence $\Rrightarrow$, $\Rrightarrow$ etc

Categories in Univalent Foundations — Take II

Definition (Category in UF)

A **category** $\mathcal{C}$ is given by

- a type $\mathcal{C}_0$ of objects
- for any $a, b : \mathcal{C}_0$, a $\boxed{\textbf{set}}$ $\mathcal{C}(a, b)$ of morphisms
- operations: identity & composition
- axioms: unitality & associativity

For this definition of category, all the postulated paths are trivially coherent.

Isomorphism in a category

Definition (Isomorphism in a category)

A morphism $f : \mathcal{C}(a, b)$ is an **isomorphism** if there are

- $$g : \mathcal{C}(b, a)$$

- $$\eta : g \circ f \rightsquigarrow \mathrm{id}_a \qquad \epsilon : f \circ g \rightsquigarrow \mathrm{id}_b$$

Put differently, we define

$$\mathrm{isIso}(f) := \sum_{g : \mathcal{C}(b,a)} \Big((g \circ f \rightsquigarrow \mathrm{id}_a) \times (f \circ g \rightsquigarrow \mathrm{id}_b) \Big)$$

Isomorphism in a category II

Lemma

For any $f : \mathcal{C}(a, b)$*, the type* $\mathsf{isIso}(f)$ *is a proposition.*

Definition (The type of isomorphisms)

$$\mathsf{Iso}(a, b) :\equiv \sum_{f:\mathcal{C}(a,b)} \mathsf{isIso}(f)$$

What about categories as objects?

Definition (Functor)

A **functor** F from $\mathcal{C}$ to $\mathcal{D}$ is given by

- a map $F_0 : \mathcal{C}_0 \to \mathcal{D}_0$
- for any $a, a' : \mathcal{C}_0$, a map $F_{a,a'} : \mathcal{C}(a, a') \to \mathcal{D}(Fa, Fa')$
- preserving identity and composition

Definition (Isomorphism of categories)

A functor F is an **isomorphism of categories** if

- F_0 is an isomorphism of types and
- $F_{a,a'}$ is an isomorphism of types (a bijection) for any a, a',

$$\text{isIsoOfCats}(F) :\equiv \Big(\ldots\Big) \times \Big(\prod_{a,a' : \mathcal{C}_0} \ldots\Big)$$

Isomorphisms of categories

Lemma

"Being an isomorphism of categories" is a proposition.

Definition (Type of isomorphisms of categories)

$$\mathcal{C} \cong \mathcal{D} := \sum_{F:\mathcal{C}\to\mathcal{D}} \mathsf{isIsoOfCats}(F)$$

Natural transformations

Definition (Natural transformation)

Let $F, G : \mathcal{C} \to \mathcal{D}$ be functors. A **natural transformation** $\alpha : F \to G$ is given by

- for any $a : \mathcal{C}_0$ a morphism $\alpha_a : \mathcal{D}(Fa, Ga)$ s.t.
- for any $f : \mathcal{C}(a, b)$, $Gf \circ \alpha_a \rightsquigarrow \alpha_b \circ Ff$

The type of natural transformations $F \to G$ is a **set.**

Definition (Functor category $\mathcal{D}^{\mathcal{C}}$)

- objects: functors from $\mathcal{C}$ to $\mathcal{D}$
- morphisms from F to G: natural transformations

Equivalence of categories

Definition (Left Adjoint)

A functor $F : \mathcal{C} \to \mathcal{D}$ is a **left adjoint** if there are

- $G : \mathcal{D} \to \mathcal{C}$
- $\eta : 1_{\mathcal{C}} \to GF$
- $\epsilon : FG \to 1_{\mathcal{D}}$
- + higher coherence data.

Equivalence of categories

Definition (Equivalence of categories)

A left adjoint F is an **equivalence of categories** if η and ϵ are isomorphisms.

Lemma

"F is an equivalence" is a proposition.

Definition

$$\mathcal{C} \simeq \mathcal{D} := \sum_{F:\mathcal{C}\to\mathcal{D}} \mathrm{isEquivOfCats}(F)$$

Table of Contents

1. Introduction to Univalent Foundations
   - Type theory and its homotopy interpretation
   - Logic in type theory: homotopy levels
   - The Univalence Axiom
2. Category Theory in Univalent Foundations
   - Categories: basic definitions
   - Univalent categories: definition & some properties
   - The Rezk completion

From paths to isomorphisms

Definition (From paths to isomorphisms, univalent categories)

For objects $a, b : \mathcal{C}_0$ we define

$$\begin{aligned} \mathsf{idtoiso}_{a,b} : (a \rightsquigarrow b) &\to \mathsf{Iso}(a, b) \\ \mathsf{refl}(a) &\mapsto \mathsf{id}_a \end{aligned}$$

We call the category $\mathcal{C}$ **univalent** if, for any objects $a, b : \mathcal{C}_0$,

$$\mathsf{idtoiso}_{a,b} : (a \rightsquigarrow b) \to \mathsf{Iso}(a, b)$$

is an isomorphism of types.

- In a univalent category, isomorphic objects are equal.
- *"$\mathcal{C}$ is univalent"* is a proposition, written $\mathsf{isUniv}(\mathcal{C})$.

Examples of univalent categories

- Set (follows from the Univalence Axiom)
- categories of algebraic structures (groups, rings,...)
  - made precise by the **Structure Identity Principle** (P. Aczel)
- full subcategories of univalent categories
- functor category $\mathcal{D}^{\mathcal{C}}$, if $\mathcal{D}$ is univalent (see below)
- if $\mathcal{C}$ is univalent, then the category of **cones** of shape $F : \mathcal{J} \to \mathcal{C}$ is
  - $\leadsto$ limits (limiting cones) in a univalent category are **unique**

1 kind of sameness for univalent categories

Equality	$\mathcal{C} \rightsquigarrow \mathcal{D}$
Isomorphism	$\mathcal{C} \cong \mathcal{D}$
Equivalence	$\mathcal{C} \simeq \mathcal{D}$

Theorem

*For **univalent** categories $\mathcal{C}$ and $\mathcal{D}$, these three are equivalent as types.*

In particular, we can substitute a univalent category with an equivalent one.

Table of Contents

1 Introduction to Univalent Foundations
Type theory and its homotopy interpretation
Logic in type theory: homotopy levels
The Univalence Axiom

2 Category Theory in Univalent Foundations
Categories: basic definitions
Univalent categories: definition & some properties
The Rezk completion

Rezk completion

- *“Being univalent”* is a proposition

$\rightsquigarrow$ Inclusion from univalent categories to categories

Theorem

The inclusion of univalent categories into categories has a left adjoint (in bicategorical sense),

$$\mathcal{C} \mapsto \widehat{\mathcal{C}}, \qquad \text{the } \textbf{Rezk completion} \text{ of } \mathcal{C} \;.$$

Rezk completion II

Any functor $F : \mathcal{C} \to \mathcal{D}$ with $\mathcal{D}$ univalent factors uniquely:

$$\begin{tikzcd}
\mathcal{C} \arrow[rr, "\eta_{\mathcal{C}}"] \arrow[drr, "\forall F"'] & & \widehat{\mathcal{C}} \arrow[d, dotted, "\exists!"] \\
& & \mathcal{D} \text{ (univalent)}
\end{tikzcd}$$

The functor $\eta_{\mathcal{C}}$ is the unit of the adjunction; it is

- fully faithful and
- essentially surjective.

Construction of the Rezk completion

- $\widehat{\mathcal{C}} :=$ full image subcat. of $\underline{\mathrm{Set}}^{\mathcal{C}^{\mathrm{op}}}$ of **Yoneda embedding**
  - $\widehat{\mathcal{C}}$ is univalent
- let $\eta_{\mathcal{C}} : \mathcal{C} \to \widehat{\mathcal{C}}$ be the **Yoneda embedding** (into $\widehat{\mathcal{C}}$):
  - fully faithful
  - essentially surjective (by definition)
- precomposition $\_ \circ H : \mathcal{C}^{\mathcal{B}} \to \mathcal{C}^{\mathcal{A}}$ is an equivalence—and hence an isomorphism—of categories if
  - H is essentially surjective
  - $\mathcal{C}$ is univalent
- the object function thus is an isomorphism of types

$$\_ \circ H : (\mathcal{C}^{\mathcal{B}})_0 \to (\mathcal{C}^{\mathcal{A}})_0$$

Special case of Rezk completion: Quotienting

Specialise: category $\rightsquigarrow$ groupoid $\rightsquigarrow$ equivalence relation

Theorem

Univalent Foundations admits quotients, i.e. any map $f : S \to R$ such that $s \sim s' \implies f(s) = f(s')$ factors uniquely via $\widehat{S}$:

$$\begin{tikzcd}
S \arrow[r, "\eta_S"] \arrow[dr, "\forall"'] & \widehat{S} \arrow[d, dotted, "\exists!"] \\
& R
\end{tikzcd}$$

- More direct construction of set-level quotients by Voevodsky: "type of equivalence classes"

Mechanization in Coq

Rezk Completion mechanized in Coq+UA+TypeInType

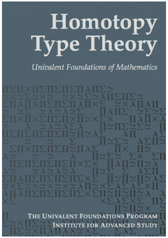
- approx. 4000 lines of code
- based on Voevodsky's library "*Foundations*"

Design choices for the implementation (same for *Foundations*)

- Goal: make maths in UF accessible for mathematicians
  - $\rightsquigarrow$ stick to that part of syntax with clear semantics
- Restriction to basic type constructors ($\prod$, $\sum$,. . .)
- Coercions and notations as in mathematical practice
- No automation: no type classes, no automatic tactics

References

- Univalent Foundations program, *Homotopy Type Theory: Univalent Foundations of Mathematics*, 2013



- Hofmann, M. and Streicher, T., *The groupid interpretation of type theory*, 1996
- Rezk, C., *A model for the homotopy theory of homotopy theory*, 2001
- preprint with same title `arXiv:1303.0584`