



Emily Riehl

Johns Hopkins University

Formalizing ∞ -category theory in the RZK proof assistant

joint with Nikolai Kudasov and Jonathan Weinberger



Interactions of Proof Assistants and Mathematics

Plan



1. Computer formalization of mathematics
2. In search of foundations for higher structures
3. Simplicial type theory and the **RZK** proof assistant
4. Synthetic ∞ -category theory
5. A formalized proof of the ∞ -categorical Yoneda lemma



1

Computer formalization of mathematics

Motivation



CAHIERS DE TOPOLOGIE
ET GÉOMÉTRIE DIFFÉRENTIELLE
CATÉGORIQUES

VOL. XXXII-1 (1991)

∞ -GROUPOIDS AND HOMOTOPY TYPES

by M.M. KAPRANOV and V.A. VOEVODSKY

RÉSUMÉ. Nous présentons une description de la catégorie homotopique des CW-complexes en termes des ∞ -groupoïdes. La possibilité d'une telle description a été suggérée par A. Grothendieck dans son mémoire "A la poursuite des champs".

It is well-known [GZ] that CW-complexes X such that $n_i(X,x) = 0$ for all $i \geq 2$, $x \in X$, are described, at the homotopy level, by groupoids. A. Grothendieck suggested, in his unpublished memoir [Gr], that this connection should have a higher-dimensional generalisation involving polycategories. viz. polycategorical analogues of groupoids. It is the purpose of this paper to establish such a generalisation.

- 15 statements =
 - + 4 theorems
 - + 9 propositions
 - + 1 lemma
 - + 1 corollary
- 5 short “obvious” proofs + 3 proofs

- Carlos Simpson’s “Homotopy types of strict 3-groupoids” (1998) shows that the 3-type of S^2 can’t be realized by a strict 3-groupoid — contradicting the last corollary.
- But no explicit mistake was found. Voevodsky: “I was sure that we were right until the fall of 2013 (!!)"



MATHEMATICS

The Origins and Motivations of Univalent Foundations

*A Personal Mission to Develop Computer Proof
Verification to Avoid Mathematical Mistakes*

By Vladimir Voevodsky • Published 2014

“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.”

Computer formalized mathematics



Formalized mathematics, in tandem with other forms of computerized mathematics¹, provides better management of mathematical knowledge, an opportunity to carry out ever more complex and larger projects, and hitherto unseen levels of precision.

— Andrej Bauer, “The dawn of formalized mathematics,”
delivered at the 8th European Congress of Mathematics

Recent successes include:

- the [Kepler conjecture](#), resolving a 1611 conjecture, 2003–2014, [HOL LIGHT](#)
- the [Feit-Thompson Odd Order Theorem](#), a foundational result in the classification of finite simple groups, 2006–2012, [Coq](#)
- the [liquid tensor experiment](#), formalizing condensed mathematics, 2020–2022, [LEAN](#)
- the [Brunerie number](#), computing $\pi_4 S^3 \cong \mathbb{Z}/2\mathbb{Z}$, 2015–2022, [CUBICAL AGDA](#)

¹Jacques Carette, William M. Farmer, Michael Kohlhase, and Florian Rabe. Big math and the one-brain barrier — the tetrapod model of mathematical knowledge. *Mathematical Intelligencer*, 43(1):78–87, 2021.



2

In search of foundations for higher structures

Rebuilding the pragmatic foundations for higher structures



I am pretty strongly convinced that there is an ongoing reversal in the collective consciousness of mathematicians: the homotopical picture of the world becomes the basic intuition, and if you want to get a discrete set, then you pass to the set of connected components of a space defined only up to homotopy ... Cantor's problems of the infinite recede to the background: from the very start, our images are so infinite that if you want to make something finite out of them, you must divide them by another infinity.

— Yuri Manin “We do not choose mathematics as our profession, it chooses us: Interview with Yuri Manin” by Mikhail Gelfand

∞ -categories in set theory



Essentially, ∞ -categories are 1-categories in which all the **sets** have been replaced by **∞ -groupoids** aka **homotopy types**:

sets :: ∞ -groupoids
categories :: ∞ -categories

Where

- categories have sets of objects, ∞ -categories have ∞ -groupoids of objects, and
- categories have hom-sets, ∞ -categories have ∞ -groupoidal mapping spaces.

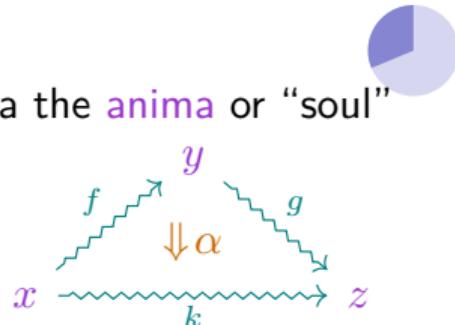
While the axioms that turn a directed graph into a category are expressed in the language of set theory — a category has a composition function satisfying axioms expressed in first-order logic with equality — composition in an ∞ -category, as a morphism between ∞ -groupoids, isn't a “function” in the traditional sense (since homotopy types do not have underlying sets of points).

This is why ∞ -categories are so difficult to model within set theory.

Composing paths

In the total singular complex aka the fundamental ∞ -groupoid aka the **anima** or “soul”

of a space X , composites of paths are witnessed by higher paths:



Theorem. The space of composites of two paths f and g in X is contractible.

Proof: The space of composites of paths f and g in X is defined by the pullback:

$$\begin{array}{ccccc} S^{n-1} & \longrightarrow & \text{Comp}(f, g) & \hookrightarrow & \text{Map}(\Delta, X) \\ \downarrow & \nearrow & \downarrow & & \downarrow \text{restrict} \\ D^n & \xrightarrow{\quad} & * & \xrightarrow{f \wedge g} & \text{Map}(\Lambda, X) \end{array} \quad \Leftrightarrow \quad \begin{array}{ccc} S^{n-1} \times \Delta \cup_{S^{n-1} \times \Lambda} D^n \times \Lambda & \longrightarrow & X \\ \downarrow \pi & & \swarrow \\ D^n \times \Delta & & \end{array}$$

A space is **contractible** just when any sphere S^{n-1} can be filled to a disk D^n for $n \geq 0$.
The extension exists since the inclusion admits a continuous deformation retract. \square

Could ∞ -category theory be taught to undergraduates?

As far as we know, there are no existing formalizations of ∞ -category theory in any proof assistant library such as LEAN-MATHLIB, AGDA-UNIMATH, Coq-HoTT, ...



Why not?

Could ∞ -Category Theory Be Taught to Undergraduates?



Emily Riehl

1. The Algebra of Paths
It is natural to probe a suitably nice topological space X by means of its paths, the continuous functions from the standard interval $[0, 1] \subset \mathbb{R}$ to X . But what structure do the paths in X form?

To start, the paths form the edges of a directed graph whose vertices are the points of X : a path $p : J \rightarrow X$ defines an arrow from the point $p(0)$ to the point $p(1)$. Moreover,

Emily Riehl is a professor of mathematics at Johns Hopkins University. Her email address is eriehl@jhu.edu.

Communicated by Notices Associate Editor Steven S. Seiden.

For permission to reprint this article, please contact:

reprint-permissions@ams.org

DOI: <https://doi.org/10.1090/noti2002>

The traditional foundations of mathematics are not really suitable for “higher mathematics” such as ∞ -category theory, where the basic objects are built out of higher-dimensional types instead of mere sets. However, there are proposals for new foundations for mathematics that are closer to mathematician’s core intuitions, based on Martin-Löf’s dependent type theory such as

- homotopy type theory,
- higher observational type theory, and the
- simplicial type theory, that we use here.

this graph is reflexive, with the constant path refl_x at each point $x \in X$ defining the self-looped end-node of x .

Can this reflexive directed graph be given the structure of a category? To do so, it is natural to define the composite of a path p from x to y and a path q from y to z by gluing together these continuous maps—i.e., by concatenating the paths—and then by reparametrizing via the homeomorphism $J \cong J \cup_{y \in J} J$ that traverses each path at double speed:

$$\begin{array}{c} I \xrightarrow{\quad \cong \quad} I \cup_{y \in J} I \xrightarrow{\quad p \circ q \quad} X \\ \downarrow \text{pq} \end{array} \quad \{1.1\}$$

But the composition operation \circ fails to be associative or unital. In general, given a path r from x to w ,



∞ -categories in homotopy type theory

The identity type family gives each type the structure of an ∞ -groupoid: each type A has a family of identity types over $x, y : A$ whose terms $p : x =_A y$ are called **paths**. In a “directed” extension of homotopy type theory introduced in

Emily Riehl and Michael Shulman, [A type theory for synthetic \$\infty\$ -categories](#),
Higher Structures 1(1):116–193, 2017

each type A also has a family of hom types $\text{Hom}_A(x, y)$ over $x, y : A$ whose terms $f : \text{Hom}_A(x, y)$ are called **arrows**.

defn (Riehl–Shulman after Joyal and Rezk). A type A is an ∞ -category if:

- Every pair of arrows $f : \text{Hom}_A(x, y)$ and $g : \text{Hom}_A(y, z)$ has a unique composite, defining a term $g \circ f : \text{Hom}_A(x, z)$.
- Paths in A are equivalent to **isomorphisms** in A .

With more of the work being done by the foundation system, perhaps someday ∞ -category theory will be easy enough to teach to undergraduates?



3

Simplicial type theory and the RZK proof assistant



Shapes in the theory of the directed interval

Our types may depend on other types and also on **shapes** $\Phi \subset \mathcal{Z}^n$, which are polytopes embedded in a directed cube defined in a language

$$\top, \perp, \wedge, \vee, \equiv \quad \text{and} \quad 0, 1, \leq$$

satisfying **intuitionistic logic** and **strict interval** axioms.

$$\text{e.g.,} \quad \Delta^n := \{(t_1, \dots, t_n) : \mathcal{Z}^n \mid t_n \leq \dots \leq t_1\}$$

$$\Delta^1 := \mathcal{Z}$$

$$\Delta^2 := \{(t_1, t_2) : \mathcal{Z}^2 \mid t_2 \leq t_1\} := \left\{ \begin{array}{c} (1,1) \\ \diagup | \diagdown \\ (t,t) \\ \hline (0,0) \quad (1,0) \\ \hline (t,0) \end{array} \right.$$

$$\partial\Delta^2 := \{(t_1, t_2) : \mathcal{Z}^2 \mid (t_2 \leq t_1) \wedge ((0 = t_2) \vee (t_2 = t_1) \vee (t_1 = 1))\}$$

$$\Lambda_1^2 := \{(t_1, t_2) : \mathcal{Z}^2 \mid (t_2 \leq t_1) \wedge ((0 = t_2) \vee (t_1 = 1))\}$$

Because $\phi \wedge \psi$ implies ϕ , there are **shape inclusions** such as $\Lambda_1^2 \subset \partial\Delta^2 \subset \Delta^2$.



Extension types

Formation rule for extension types

$$\frac{\Phi \subset \Psi \text{ shape} \quad A \text{ type} \quad a : \Phi \rightarrow A}{\left\langle \begin{array}{c} \Phi \xrightarrow{a} A \\ \Downarrow \\ \Psi \end{array} \right\rangle \text{ type}}$$

A term $f : \left\langle \begin{array}{c} \Phi \xrightarrow{a} A \\ \Downarrow \\ \Psi \end{array} \right\rangle$ defines

$$f : \Psi \rightarrow A \text{ so that } f(t) \equiv a(t) \text{ for } t : \Phi.$$

The simplicial type theory allows us to *prove* equivalences between extension types along composites or products of shape inclusions.

An experimental proof assistant Rzk for ∞ -category theory



rzk

MkDocs documentation Haddock documentation Build with GHCJS and Deploy to GitHub Pages passing

An experimental proof assistant for synthetic ∞ -categories.

rzk: an experimental proof assistant for synthetic ∞ -categories

Search docs

GENERAL

- About
- RZK-L LANGUAGE
- Introduction
- Rendering Diagrams
- Examples
- Weak type direction elimination

TOOLS

- Continuous Verification

RELATED PROJECTS

- shoTT
- simple-topes

If a term is extracted as a part of a larger shape, generally, the whole shape will be shown (in gray). If a term is extracted as a part of a larger shape, generally, the whole shape will be shown (in gray).

Built with MkDocs using a theme provided by Read the Docs.

(R817, Definition 5.2)
The type of commutative triangles in A

#def! triangle

t : 0D
x : A
y : A
z : A
e : hom A * x
f : hom A * y
g : hom A * z
h : hom A * x
i : hom A * y
j : hom A * z
k : hom A * x
l : hom A * y
m : hom A * z
n : hom A * x
o : hom A * y
p : hom A * z
q : hom A * x
r : hom A * y
s : hom A * z
t : hom A * x
u : hom A * y
v : hom A * z
w : hom A * x
x : hom A * y
y : hom A * z
z : hom A * x
a : hom A * y
b : hom A * z
c : hom A * x
d : hom A * y
e : hom A * z
f : hom A * x
g : hom A * y
h : hom A * z
i : hom A * x
j : hom A * y
k : hom A * z
l : hom A * x
m : hom A * y
n : hom A * z
o : hom A * x
p : hom A * y
q : hom A * z
r : hom A * x
s : hom A * y
t : hom A * z
u : hom A * x
v : hom A * y
w : hom A * z
x : hom A * x
y : hom A * y
z : hom A * z

The right edge is exactly g, and the diagonal is exactly h



Visualising Terms of Simplicial Types

Terms (with non-trivial labels) are visualised with red color (you can see a detailed label on hover). Recognised parameter part (e.g. fixed endpoints, edges, faces with clear labels) are visualised with purple color. When a term is constructed by taking a part of another shape, the rest of the larger shape is coloured using gray color.

We can visualise terms that fill a shape:

#def! square

t : 0D
x : A
y : A
z : A
h : hom A * x
i : hom A * y
j : hom A * z
k : hom A * x
l : hom A * y
m : hom A * z
n : hom A * x
o : hom A * y
p : hom A * z
q : hom A * x
r : hom A * y
s : hom A * z
t : hom A * x
u : hom A * y
v : hom A * z
w : hom A * x
x : hom A * y
y : hom A * z
z : hom A * x
a : hom A * y
b : hom A * z
c : hom A * x
d : hom A * y
e : hom A * z
f : hom A * x
g : hom A * y
h : hom A * z
i : hom A * x
j : hom A * y
k : hom A * z
l : hom A * x
m : hom A * y
n : hom A * z
o : hom A * x
p : hom A * y
q : hom A * z
r : hom A * x
s : hom A * y
t : hom A * z
u : hom A * x
v : hom A * y
w : hom A * z
x : hom A * x
y : hom A * y
z : hom A * z



#lang rzk-1

2 -- (R817, Definition 5.1)

3 -- An arrow in A from x to y,

4 #def! arrow

t : 0D
x : A
y : A
z : A
t == 0_2 |-> x,
t == 1_2 |-> y

5 -- An arrow in A from x to y,

6 #def! arr

t : 0D
x : A
y : A
z : A
t == 0_2 |-> x,
t == 1_2 |-> y

7 -- (R817, Equation 6.1)

8 -- An arrow in C from x to y, over the arrow f in A from x to y.

9 #def! dhom

(A : Type)
(C : Type)
(f : hom A * x)
(C : A --> C)
(x : C y)
(w : C y)

t : 0D

10 -- (R817, Definition 5.2)

11 -- The type of commutative triangles in the type family C

12 -- over the arrow f in A from x to y.

13 #def! triangle

(A : Type)
(C : Type)
(f : hom A * x)
(C : A --> C)
(x : C y)

t : 0D

14 -- (R817, Equation 6.1)

15 -- An arrow in C from x to y, over the arrow f in A from x to y.

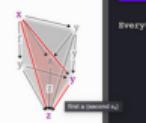
16 #def! arr

(A : Type)
(C : Type)
(f : hom A * x)
(C : A --> C)
(x : C y)

t : 0D

TYPECHECK (CTRL + ENTER)

Everything is ok!



Next

The proof assistant Rzk was written by Nikolai Kudasov:

About this project

This project has started with the idea of bringing Riehl and Shulman's 2017 paper [1] to "life" by implementing a proof assistant based on their type theory with shapes. Currently an early prototype with an [online playground](#) is available. The current implementation is capable of checking various formalisations. Perhaps, the largest formalisations are available in two related projects: <https://github.com/rizruk/shoTT> and <https://github.com/emilyiehlyoneda/shoTT>. The shoTT project (originally a fork of the yoneda project) aims to cover more formalisations in simplicial HoTT and ∞ -categories, while yoneda project aims to compare different formalisations of the Yoneda lemma.

Internally, rzk uses a version of second-order abstract syntax allowing relatively straightforward handling of binders (such as lambda abstraction). In the future, rzk aims to support dependent type inference relying on E-unification for second-order abstract syntax [2]. Using such representation is motivated by automatic handling of binders and easily automated boilerplate code. The idea is that this should keep the implementation of rzk relatively small and less error-prone than some of the existing approaches to implementation of dependent type checkers.

An important part of rzk is a type layer solver, which is essentially a theorem prover for a part of the type theory. A related project, dedicated just to that part is available at <https://github.com/rizruk/simple-topes>. simple-topes supports user-defined cubes, topes, and type layer axioms. Once stable, simple-topes will be merged into rzk, expanding the proof assistant to the type theory with shapes, allowing formalisations for (variants of) cubical, globular, and other geometric versions of HoTT.

rzk-lang.github.io/rzk

A formalized proof of the ∞ -categorical Yoneda lemma



Our initial aim was to write a formalized proof of the ∞ -categorical Yoneda lemma.

github.com/emilyriehl/yoneda or emilyriehl.github.io/yoneda/

- proof from Emily Riehl & Mike Shulman, [A type theory for synthetic \$\infty\$ -categories](#), Higher Structures 2017.
- formalizations written by [Nikolai Kudasov](#), [Emily Riehl](#), [Jonathan Weinberger](#).
- completed March 12 – April 17, 2023

Our ultimate aim is to compare ∞ -category theory in simplicial type theory with ordinary category theory in traditional foundations. Thus,

- We've included a formalization of the 1-categorical Yoneda lemma in Lean by [Sina Hazratpour](#) as part of an Introduction to Proofs course at Johns Hopkins.
- We wrote a first version of [yoneda-lemma-precategories.lagda.md](#).

More recently, we've professionalized our library, implementing a style guide suggested by [Fredrik Bakke](#), who also contributed some formalizations.

Future work: formalize synthetic ∞ -category theory



Help us formalize other results from synthetic ∞ -category in [Rzk](#)!

We have suggested some formalization goals at

github.com/rzk-lang/sHoTT or rzk-lang.github.io/sHoTT

that should be achievable if you have prior familiarity with homotopy type theory.

It is also possible to formalize standard (book) homotopy type theory in [Rzk](#), and these background results are used as prerequisites for our main project.



4

Synthetic ∞ -category theory

Hom types



In the simplicial type theory, any type A has a family of hom types depending on two terms in $x, y : A$:

$$\text{Hom}_A(x, y) := \left\langle \begin{array}{c} \partial\Delta^1 \xrightarrow{[x,y]} A \\ \Downarrow \\ \Delta^1 \end{array} \right\rangle \text{type}$$

A term $f : \text{Hom}_A(x, y)$ defines an arrow in A from x to y .

We think of the type $\text{Hom}_A(x, y)$ as the mapping space in A from x to y .

A type A also has a family of identity types or path spaces $x = y$ depending on two terms in $x, y : A$, which we will connect to the hom-types momentarily.

Pre- ∞ -categories



defn (Riehl–Shulman after Joyal). A type A is a **pre- ∞ -category** if every pair of arrows $f : \text{Hom}_A(x, y)$ and $g : \text{Hom}_A(y, z)$ has a **unique composite**, i.e.,

$$\left\langle \begin{array}{ccc} \Lambda_1^2 & \xrightarrow{[f,g]} & A \\ \Downarrow & \nearrow & \\ \Delta^2 & & \end{array} \right\rangle \quad \text{is contractible.}^a$$

^aA type C is contractible just when $\sum_{c:C} \prod_{x:C} c = x$.

By contractibility, $\left\langle \begin{array}{ccc} \Lambda_1^2 & \xrightarrow{[f,g]} & A \\ \Downarrow & \nearrow & \\ \Delta^2 & & \end{array} \right\rangle$ has a unique inhabitant $\text{comp}_{f,g} : \Delta^2 \rightarrow A$.

Write $g \circ f : \text{Hom}_A(x, z)$ for its inner face, *the composite* of f and g .

Identity arrows



For any $x : A$, the constant function defines a term

$$\text{id}_x := \lambda t.x : \text{Hom}_A(x, x) := \left\langle \begin{array}{c} \partial\Delta^1 \xrightarrow{[x,x]} A \\ \Downarrow \\ \Delta^1 \end{array} \right\rangle,$$

which we denote by id_x and call the identity arrow.

For any $f : \text{Hom}_A(x, y)$ in a pre- ∞ -category A , the term in the contractible type

$$\lambda(s, t).f(t) : \left\langle \begin{array}{c} \Lambda_1^2 \xrightarrow{[\text{id}_x, f]} A \\ \Downarrow \\ \Delta^2 \end{array} \right\rangle$$

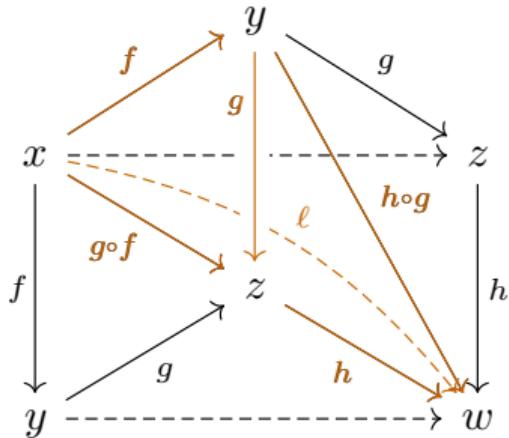
witnesses the unit axiom $f = f \circ \text{id}_x$.

Associativity of composition



Prop. In a pre- ∞ -category A , composition is associative: for any arrows $f : \text{Hom}_A(x, y)$, $g : \text{Hom}_A(y, z)$, and $h : \text{Hom}_A(z, w)$, we have $h \circ (g \circ f) = (h \circ g) \circ f$.

Proof: Consider the composable arrows in the pre- ∞ -category $\Delta^1 \rightarrow A$:



Composing defines a term in the type $\Delta^2 \rightarrow (\Delta^1 \rightarrow A)$ which defines an arrow $\ell : \text{Hom}_A(x, w)$ so that $\ell = h \circ (g \circ f)$ and $\ell = (h \circ g) \circ f$.



Isomorphisms

An arrow $f: \text{Hom}_A(x, y)$ in a pre- ∞ -category is an **isomorphism** if it has a two-sided inverse $g: \text{Hom}_A(y, x)$. However, the type

$$\sum_{g: \text{Hom}_A(y, x)} (g \circ f = \text{id}_x) \times (f \circ g = \text{id}_y)$$

has higher-dimensional structure and is *not* a **proposition**. Instead define

$$\text{is-iso}(f) := \left(\sum_{g: \text{Hom}_A(y, x)} g \circ f = \text{id}_x \right) \times \left(\sum_{h: \text{Hom}_A(y, x)} f \circ h = \text{id}_y \right).$$

For $x, y : A$, the **type of isomorphisms** from x to y is:

$$x \cong_A y := \sum_{f: \text{Hom}_A(x, y)} \text{is-iso}(f).$$



∞ -categories

By path induction, to define a map

$$\text{iso-eq}: (x =_A y) \rightarrow (x \cong_A y)$$

for all $x, y : A$ it suffices to define

$$\text{iso-eq}(\text{refl}_x) := \text{id}_x.$$

defn (Riehl–Shulman after Rezk). A pre- ∞ -category A is ∞ -category iff every isomorphism is an identity, i.e., iff the map

$$\text{iso-eq}: \prod_{x,y:A} (x =_A y) \rightarrow (x \cong_A y)$$

is an equivalence.



∞ -groupoids

Similarly by path induction define

$$\text{arr-eq}: (x =_A y) \rightarrow \text{Hom}_A(x, y)$$

for all $x, y : A$ by $\text{arr-eq}(\text{refl}_x) := \text{id}_x$.

A type A is an ∞ -groupoid iff every arrow is an identity, i.e., iff arr-eq is an equivalence.

Prop. A type is an ∞ -groupoid if and only if it is an ∞ -category and all of its arrows are isomorphisms.

Proof:

$$\begin{array}{ccc} x =_A y & \xrightarrow{\text{arr-eq}} & \text{Hom}_A(x, y) \\ & \searrow \text{iso-eq} & \swarrow \\ & x \cong_A y & \end{array}$$

∞ -categories for undergraduates



defn. An ∞ -groupoid is a type in which arrows are equivalent to identities:

arr-eq: $(x =_A y) \rightarrow \text{Hom}_A(x, y)$ is an equivalence.

defn. An ∞ -category is a type

- which has unique binary composites of arrows:

$$\left\langle \begin{array}{c} \Lambda_1^2 \xrightarrow{[f,g]} A \\ \Downarrow \\ \Delta^2 \end{array} \right\rangle \quad \text{is contractible}$$

- and in which isomorphisms are equivalent to identities:

iso-eq: $(x =_A y) \rightarrow (x \cong_A y)$ is an equivalence.



5

A formalized proof of the ∞ -categorical Yoneda lemma

Covariant type families

defn (Riehl–Shulman after Joyal). A type family $B(x)$ over $x : A$ is covariant if for every $f : \text{Hom}_A(x, y)$ and $u : B(x)$ there is a unique lift of f with domain u .

The codomain of the unique lift defines a term $f_* u : B(y)$.

Prop. Fix $a : A$. The type family $\text{Hom}_A(a, x)$ over $x : A$ is covariant if and only if A is a pre- ∞ -category.

Prop. When A is a pre- ∞ -category, for any $u : B(x)$, $f : \text{Hom}_A(x, y)$, and $g : \text{Hom}_A(y, z)$, then $g_*(f_* u) = (g \circ f)_* u$ and $(\text{id}_x)_* u = u$.

Prop. For any covariant families $B(x)$ and $C(x)$ over $x : A$, a pre- ∞ -category, any family of maps $\phi : \prod_{x:A} B(x) \rightarrow C(x)$ is natural.

Prop. If $B(x)$ is covariant over $x : A$, a pre- ∞ -category, then each fiber $B(x)$ is an ∞ -groupoid.



The Yoneda lemma

Let $B(x)$ be a covariant family over $x : A$, a pre- ∞ -category, and fix $a : A$.

Yoneda lemma. The maps

$$\text{evid} := \lambda\phi.\phi(a, \text{id}_a) : \left(\prod_{x:A} \text{Hom}_A(a, x) \rightarrow B(x) \right) \rightarrow B(a) \quad \text{and}$$

$$\text{yon} := \lambda u.\lambda x.\lambda f.f_*u : B(a) \rightarrow \left(\prod_{x:A} \text{Hom}_A(a, x) \rightarrow B(x) \right)$$

are inverse equivalences.

Proof: By definition, $\text{evid} \circ \text{yon}(u) := (\text{id}_a)_*u$. By functoriality $(\text{id}_a)_*u = u$, so yon is a section of evid . To see that yon is a retraction of evid , start from the definition $\text{yon} \circ \text{evid}(\phi)(x, f) := f_*\phi(a, \text{id}_a)$. By naturality of ϕ and the identity law for pre- ∞ -categories $f_*\phi(a, \text{id}_a) = \phi(x, f \circ \text{id}_a) = \phi(x, f)$. □

Conclusions and future work



Observations:

- ∞ -category theory is significantly easier to formalize in a foundation system based on homotopy type theory.
- By moving much of the complexity of “higher structures” into the background foundation system, the gap between ∞ -category theory and 1-category narrows substantially.
- A computer proof assistant is a fantastic tool for learning to write proofs in new foundations — indeed, through formalization in RZK we caught an error of circular reasoning in the Riehl–Shulman paper!

Future work:

- We would love help formalizing more results from ∞ -category theory in RZK.
- But the initial version of the simplicial type theory is not sufficiently powerful to prove all results about ∞ -categories, so further extensions of this synthetic framework are needed.

References

- Emily Riehl and Michael Shulman, [A type theory for synthetic \$\infty\$ -categories](#), Higher Structures 1(1):116–193, 2017; [arXiv:1705.07442](#)
- Emily Riehl, [Could \$\infty\$ -category theory be taught to undergraduates?](#), Notices of the AMS 70(5):727–736, May 2023; [arXiv:2302.07855](#)
- César Bardomiano Martínez, [Limits and colimits of synthetic \$\infty\$ -categories](#), [arXiv:2202.12386](#)
- Ulrik Buchholtz, Jonathan Weinberger, [Synthetic fibered \$\(\infty, 1\)\$ -category theory](#), Higher Structures 7(1): 74–165, 2023; [arXiv:2105.01724](#)
- Nikolai Kudasov, Emily Riehl, Jonathan Weinberger, [Formalizing the \$\infty\$ -categorical Yoneda lemma](#), 1–13; [arXiv:2309.08340](#)

Danke!