

Formalizing Higher Categories

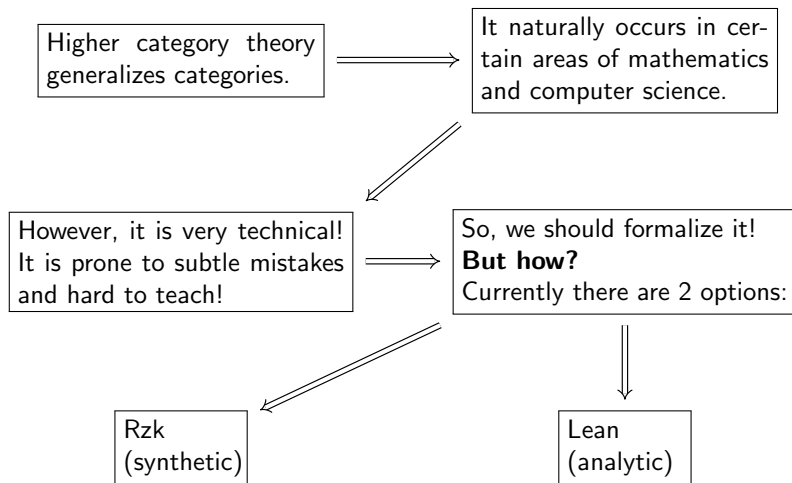
Nima Rasekh

Universität Greifswald



13.03.2025

Overview of the Talk



Category Theory: The Mathematics of Mathematics

Category theory unifies and axiomatizes different mathematics:

$$\left. \begin{array}{l} \text{Monoids} \\ \text{Topologies} \\ \text{Sets} \\ \text{Geometries} \\ \text{Types} \end{array} \right\} \Rightarrow \text{Category} = \left\{ \begin{array}{l} \text{Objects } (\text{Set, type, monoid, topology, ...}) \\ \text{Morphisms } (\text{Function, homomorphism, continuity, ...}) \\ \text{Composition } (g \circ f: X \rightarrow Y \rightarrow Z) \\ \text{Associativity } (h \circ (g \circ f) = (h \circ g) \circ f) \\ \text{Unitality } (\text{id}_X \circ f = f = \text{id}_Y \circ f) \end{array} \right.$$

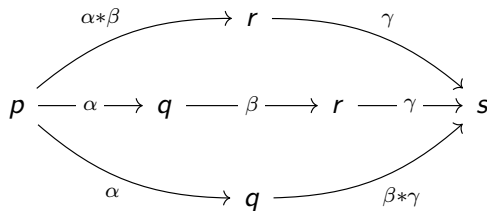
\Rightarrow **Structure & compare** mathematics!

Non-Examples

Some compositions cannot give us a category.

- Composition of loops in a topological space.
- Transitivity of equality for intensional equality types.

$$p, q, r, s : X, \alpha : p =_X q, \beta : p =_X r, \gamma : r =_X s$$



$$\alpha * (\beta * \gamma), (\alpha * \beta) * \gamma : p =_X s$$

Higher Category Theory: Math. of Higher Math.

Higher category theory axiomatizes higher mathematics:

$$\left. \begin{array}{l} \text{Points} \\ \text{Homotopy types} \\ \text{Higher algebra} \\ \text{Derived geometry} \end{array} \right\} \Rightarrow \text{Higher Category} = \left\{ \begin{array}{l} \text{Objects (Point, Homotopy type, ring spectrum, ...)} \\ \text{Morphisms (Paths, Continuous map, Ring map, ...)} \\ \text{Higher morphisms (Homotopies, ...)} \\ \text{Composition } (g \circ f : X \rightarrow Y \rightarrow Z) \\ \text{Associativity } (h \circ (g \circ f) \stackrel{\alpha}{\simeq} (h \circ g) \circ f) \\ \text{Unitality } (\text{id}_Y \circ f \stackrel{\lambda}{\simeq} f \stackrel{\rho}{\simeq} f \circ \text{id}_X) \\ \text{Higher coherences } ((\text{id} \times \lambda) \circ \alpha \simeq (\rho \times \text{id}), \dots) \end{array} \right.$$

\Rightarrow **Structure & compare** coherent mathematics!

Naively trying to define Higher Categories

Naively we could do the following:

- Objects: X, Y, Z, W, \dots
- Morphisms: $f: X \rightarrow Y, g: Y \rightarrow Z, h: Y \rightarrow Z, \dots$
- Composition: $f \circ g: X \rightarrow Z$
- 2-morphisms: $\alpha, \beta, \gamma, \dots$ between 1-morphisms
- Associativity: $\alpha: f \circ (g \circ h) \xrightarrow{\cong} (f \circ g) \circ h$
- 3-morphisms ...
- ...

This will never end ... we need to bundle it up!

A more systematic approach: Simplices

Definition (Category of Simplices)

Let Δ be the category of finite non-empty linearly ordered sets and order-preserving maps. ($[n] = \{0 < 1 < \dots < n\}$)

$$[0] \begin{array}{c} \xrightarrow{\delta^0} \\ \xleftarrow{\delta^1} \end{array} [1] \begin{array}{c} \xrightarrow{\delta^0} \\ \xleftarrow{\delta^1} \\ \xleftarrow{\delta^2} \end{array} [2] \begin{array}{c} \xrightarrow{\delta^0} \\ \xleftarrow{\delta^1} \\ \xleftarrow{\delta^2} \\ \xleftarrow{\delta^3} \end{array} [3] \begin{array}{c} \xrightarrow{\delta^0} \\ \xleftarrow{\delta^1} \\ \xleftarrow{\delta^2} \\ \xleftarrow{\delta^3} \\ \xleftarrow{\delta^4} \end{array} \dots$$

Example

- $\delta^0 : [0] \rightarrow [1] = \{0 < 1\}$, $\delta^0(0) = 1$
- $\delta^1 : [0] \rightarrow [1] = \{0 < 1\}$, $\delta^1(0) = 0$
- $\delta^0 : [1] \rightarrow [2] = \{0 < 1 < 2\}$, $\delta^0(0) = 1, \delta^0(1) = 2$

Simplicial Objects

Definition (Simplicial Object)


A simplicial object in a category \mathcal{C} is a functor $X: \Delta^{op} \rightarrow \mathcal{C}$.

$$X_0 \begin{array}{c} \xleftarrow{d_0} \\ \xrightarrow{d_1} \end{array} X_1 \begin{array}{c} \xleftarrow{d_0} \\ \xrightarrow{d_2} \end{array} X_2 \begin{array}{c} \xleftarrow{d_0} \\ \xrightarrow{d_3} \end{array} X_3 \begin{array}{c} \xleftarrow{d_0} \\ \xrightarrow{d_4} \end{array} \dots$$

Simplicial sets (simplicial objects in sets) are supposed to be higher categories! But how?

Why are these higher categories?

- X_0 are the objects
- X_1 are the morphisms
- $d_0, d_1: X_1 \rightarrow X_0$ are source and target
- $s_0: X_0 \rightarrow X_1$ is the identity
- X_2 is the set of “composable pairs of morphisms”
- So, composition is given by:

$$\{(f, g) \in X_1 \times X_1 : d_0 f = d_1 g\} \xleftarrow{(d_0, d_2)} X_2 \xrightarrow{d_1} X_1$$


- Associativity comes via similar sections into X_3 .
- More generally X_n is the set of n -composable morphisms.

Quasi-Categories vs. Complete Segal Spaces

Definition (Quasi-Category: First try!)

A **quasi-category (QCat)** is a simplicial set X with *inner horn lifting condition*, meaning maps like

$$(d_0, d_2): X_2 \rightarrow \{(f, g) \in X_1 \times X_1 : d_0 f = d_1 g\}$$

admit a section.

Definition (Complete Segal Space)

A **complete Segal space (CSS)** is a simplicial topological space X with the *Segal* and *completeness* condition.

Remark

These two are suitably equivalent: One with more lifting data, one with more structure.

Examples

Example (Free Morphism)

We have $\Delta[1]: \Delta^{op} \rightarrow \text{Set}$ with $\Delta[1]_n = \text{Hom}([n], [1])$

$$\{x, y\} \begin{array}{c} \xleftarrow{d_0} \\ \xrightarrow{d_1} \end{array} \{\text{id}_x, f, \text{id}_y\} \begin{array}{c} \xleftarrow{d_0} \\ \xrightarrow{d_2} \end{array} \{(\text{id}_x, \text{id}_x), (\text{id}_x, f), (f, \text{id}_y), (\text{id}_y, \text{id}_y)\} \begin{array}{c} \xleftarrow{d_0} \\ \xrightarrow{d_3} \end{array} \dots$$

so it is the quasi-category $f: x \rightarrow y$ (one non-trivial arrow).

Example (Chain of Morphisms)

More generally, $\Delta[n]: \Delta^{op} \rightarrow \text{Set}$ with $\Delta[n]_m = \text{Hom}([m], [n])$ is the quasi-category $x_0 \rightarrow x_1 \rightarrow \dots \rightarrow x_{n+1}$.

Remark

A map $\Delta[n] \rightarrow X$ precisely corresponds to an element in X_n !

Quasi-Categories reformulated

Definition (Quasi-Categories: More precise)

A *quasi-category* is a simplicial set $X: \Delta^{op} \rightarrow \mathbf{Set}$, such that for all $n \geq 0$ and $0 < i < n$, the following diagram admits a lift

$$\begin{array}{ccc} \Lambda[n]_i & \longrightarrow & X \\ \downarrow & \nearrow \text{dashed} & \\ \Delta[n] & & \end{array},$$

which is equivalent to saying the map

$$X_n \cong \mathrm{Hom}(\Delta[n], X) \rightarrow \mathrm{Hom}(\Lambda[n]_i, X)$$

is surjective.

Example

For $\Lambda[2]_1$, we have

$$\mathrm{Hom}(\Lambda[2]_1, X) \cong \{(f, g) \in X_1 \times X_1 : d_0 f = d_1 g\}$$

Higher Categories are hard!

① Homotopy Hypothesis:

- Published proof by Voevodsky and Kapranov in 1991
- Counter example by Simpson in 1998
- No concrete mistake found until 2013

“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.”¹

② Defining Higher Categories:

- No explicit characterization of higher categorical coherences due to lifting conditions beyond **level 3**.
- Even **level 3** too complicated to keep track of.
- Hence explicit higher category theory is almost exclusively done at **level 2**.

¹<https://www.ias.edu/ideas/2014/voevodsky-origins>

Summary:

- ① Higher categories are out there.
- ② They are defined as a collection of intricately linked sets (or topologies) with various lifting conditions
- ③ They are difficult to work with.
- ④ So we want to formalize them!

Have a Break! Have a KitKat!

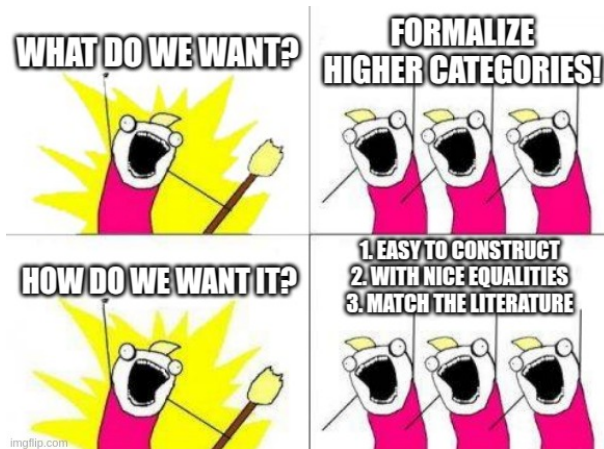


Questions?

Comments?

Suggestions?

What do we want?



Unfortunately, we can't have all of these!

How can we formalize Higher Categories?

Roughly three approaches:

	Type Theory	Model	Proof Assistant
Synthetic	sHoTT (simplicial HoTT)	CSS	Rzk
Semi-Synthetic	HoTT (Homotopy TT)	CSS	-
Analytic	TT (Type Theory)	QCat	Lean

The Semi-Synthetic Approach: -

This approach requires “constructing a simplicial object in HoTT”, which is hard, despite serious efforts.²

Remark (Honorable mention)

There are formalizations of

- bicategories³
- double categories⁴

in Coq UniMath.

² N. Kraus, “Internal ∞ -Categorical Models of Dependent Type Theory: Towards 2LTT Eating HoTT,” 2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)

³ Ahrens B, Frumin D, Maggesi M, Veltri N, van der Weide N. Bicategories in univalent foundations. Mathematical Structures in Computer Science. 2021;31(10):1232-1269.

⁴ N. van der Weide, N. Rasekh, B. Ahrens, and P. R. North. 2024. Univalent Double Categories. In Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2024). Association for Computing Machinery, New York, NY, USA, 246–259.

The Synthetic Approach: Rzk

- In Rzk a “type” is a simplicial topological space.
- A higher category is a type with a property.
- Kudasov–Riehl–Weinberger use this approach to prove the ∞ -categorical Yoneda lemma.⁵

Benefits:

- 1 The constructions are manageable.
- 2 Naive proofs for categories work quite well.

For example, compositions of morphisms are **unique**.

Drawbacks:

- 1 Foundation doesn't match existing proofs in literature.
- 2 Very specialized, little support.

⁵

N. Kudasov, E. Riehl, and J. Weinberger. 2024. Formalizing the ∞ -Categorical Yoneda Lemma. In Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2024).

The Analytic Approach: Lean

- A higher category here is a quasi-category.
- Current effort: The ∞ -cosmos project⁶

Benefits:

- 1 Accessible with a lot of support, tactics, ...
- 2 Foundation matches literature.
- 3 Blueprint is resulting in a lot of foundational simplicial machinery

Drawbacks:

- 1 Need to deal with all details.
- 2 Identities are not well-behaved:
Means we have to prove “**by hand**” that every property is equivalence invariant.

⁶<https://emilyriehl.github.io/infinity-cosmos/>

Equivalence Invariance for Categories

“Having a terminal object is equivalence invariant”

```
-- If C is equivalent to D and C has a terminal object, then so does D.
def transp_terminal_along_equiv [Category C] [Category D] (weq: C  $\equiv$  D)
  (has_terminal_C: Limits.HasTerminal C):
  Limits.HasTerminal D := by
    let F := weq.functor
    let G := weq.inverse
    let t_C := Limits.terminal C
    let t_D := F.obj (Limits.terminal C)
    have h :  $\forall Y : D, \text{Unique } (Y \rightarrow t_D)$  := by
      {
        intro Y
        have nat_iso_Y := Adjunction.equivHomsetLeftOfNatIso
          weq.counitIso (X := Y) (Y := t_D)
        have unique_map_to_GY : Unique ((G  $\ggg$  F).obj Y  $\rightarrow$  t_D) := by
          {
            have unique_map_from_GF : Unique (G.obj Y  $\rightarrow$  t_C) := by apply Limits.uniqueToTerminal
            apply (uniqueEquivEquivUnique ((G  $\ggg$  F).obj Y  $\rightarrow$  F.obj (t_C)) (G.obj Y  $\rightarrow$  t_C)).2
            apply weq.fullyFaithfulFunctor.homEquiv.symm
          }
        apply (uniqueEquivEquivUnique (Y  $\rightarrow$  t_D) ((G  $\ggg$  F).obj Y  $\rightarrow$  t_D)).2
        apply nat_iso_Y.symm
      }
    apply (Limits.hasTerminal_of_unique t_D)
```

What are some challenges?

We want a functor category: $\text{Fun}(\mathcal{C}, \mathcal{D})$!

Rzk

```

#def is-local-horn-inclusion-function-type uses (funext)
  ( X : U)
  ( A : X → U)
  ( fiberwise-is-segal-A : (x : X) → is-local-horn-inclusion (A x))
  : is-local-horn-inclusion ((x : X) → A x)
:=
  is-equiv-triple-comp
    ( Δ² → ((x : X) → A x))
    ( ( x : X) → Δ² → A x)
    ( ( x : X) → A → A x)
    ( A → ((x : X) → A x))
    ( \ g x t → g t x) -- first equivalence
    ( second (flip-ext-fun
      ( 2 × 2)
      ( Δ²)
      ( \ t → BOT)
      ( X)
      ( \ t → A)
      ( \ t → recBOT)))
    ( \ h x t → h x t) -- second equivalence
    ( second (equiv-function-equiv-family
      ( funext)
      ( X)
      ( \ x → (Δ² → A x))
      ( \ x → (A → A x))
      ( \ x → (horn-restriction (A x) , fiberwise-is-segal-A x)))
    ( \ h t x → (h x) t) -- third equivalence
    ( second (flip-ext-fun-inv
      ( 2 × 2)
      ( A)
      ( \ t → BOT)
      ( X)
      ( \ t → A)
      ( \ t → recBOT)))

#def is-segal-function-type uses (funext)
  ( X : U)
  ( A : X → U)
  ( fiberwise-is-segal-A : (x : X) → is-segal (A x))
  : is-segal ((x : X) → A x)
:=
  is-segal-is-local-horn-inclusion
    ( ( x : X) → A x)
    ( is-local-horn-inclusion-function-type
      ( X) (A)
      ( \ x → is-local-horn-inclusion-is-segal (A x) (fiberwise-is-segal-A x)))

```

Lean

Need to construct the following lift:

$$\begin{array}{ccc}
 \mathcal{C} \times \Lambda[n]_i & \xrightarrow{\quad} & \mathcal{D} \\
 \downarrow & \nearrow & \\
 \mathcal{C} \times \Delta[n] & &
 \end{array}$$

For $\Lambda[2]_1 \rightarrow \Delta[2]$ this lift means

“natural transformations compose”.

This already is beyond the existing results in Lean!

Every Challenge is an Opportunity

Hidden in this challenge is an opportunity for Lean! Many proofs in quasi-category theory follow this script:

- 1 Want to prove a theorem about a quasi-category \mathcal{C} .
- 2 Describe the assumptions as a map $A \rightarrow \mathcal{C}$
- 3 Describe the desired outcome as a diagram $B \rightarrow \mathcal{C}$
- 4 Prove \mathcal{C} lifts against $A \rightarrow B$ doing **simplicial combinatorics**!

This suggests a *tactic* ...

A Tactic for Simplicial Saturations

Potential tactic in Lean, which would be **genuinely** useful:

- **Input:** A set of inclusions of simplicial sets (such as inner horns) S and a specific map $f: A \rightarrow B$.
- **Output:** A proof that f is in the *saturation*, meaning can be obtained from maps in S via specific rules (composition, retract, pushouts, ...).

Remark

There is a version of this in cubical Agda⁷, but written in Haskell and for undirected (invertible) input.

⁷

M. Doré, E. Cavallo, and A. Mörtberg. Automating Boundary Filling in Cubical Agda. In 9th International Conference on Formal Structures for Computation and Deduction (FSCD 2024).

Summary

- ① Higher category theory has many applications but can be difficult to work with.
- ② Ideally formalization can help fix this.
- ③ However, formalization is either
 - somewhat niche (Rzk)
 - or very technical itself (Lean)
- ④ Things should get easier.

One room for improvement: specialized tactics for simplicial combinatorics.

The End

Thank you for your time!

For more about **higher categories**:

- Ask me in person!
- Website: nimarasekh.github.io
- Email: nima.rasekh@uni-greifswald.de

For more higher categories in **Lean** check out:

- ∞ -cosmos project: <https://emilyriehl.github.io/infinity-cosmos/>

For more higher categories in **Rzk** check out:

- Rzk proof assistant: <https://rzk-lang.github.io/>