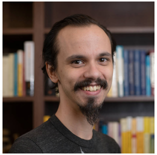
Emily Riehl

Johns Hopkins University

Prospects for Computer Formalization of Infinite-Dimensional Category Theory

joint with Mario Carneiro, Nikolai Kudasov, Dominic Verity, and Jonathan Weinberger




Formalisation of mathematics
with interactive theorem provers

Abstract

A peculiarity of the ∞-categories literature is that proofs are often written without reference to a concrete definition of an ∞-category, a practice that creates an impediment to formalization. We describe three broad strategies that would make ∞-category theory formalizable, which may be described as

(i) analytic, (ii) axiomatic, and (iii) synthetic.

We then highlight two parallel ongoing collaborative efforts to formalize ∞-category theory in two different proof assistants:

- the axiomatic theory in `Lean` and
- the synthetic theory in `Rzk`.

We show some sample formalized proofs to highlight the advantages and drawbacks of each approach and explain how you could contribute to this effort. This involves joint work with Mario Carneiro, Nikolai Kudasov, Dominic Verity, Jonathan Weinberger, and many others.

Plan

1. Prospects for formalizing the ∞-categories literature

2. Formalizing axiomatic ∞-category theory via ∞-cosmoi in `Lean`

3. Formalizing synthetic ∞-category theory in simplicial HoTT in `Rzk`

1

Prospects for formalizing the ∞-categories literature

Avoiding a precise definition of ∞-categories

The precursor to Jacob Lurie's *Higher Topos Theory* is a 2003 preprint On ∞-Topoi, which avoids using a precise definition of ∞-categories:

> *We will begin in §1 with an informal review of the theory of ∞-categories. There are many approaches to the foundation of this subject, each having its own particular merits and demerits. Rather than single out one of those foundations here, we shall attempt to explain the ideas involved and how to work with them. The hope is that this will render this paper readable to a wider audience, while experts will be able to fill in the details missing from our exposition in whatever framework they happen to prefer.*

Perlocutions of this form are quite common in the field.

Very roughly, an ∞-category is a weak infinite-dimensional category.

In the parlance of the field, selecting a set-theoretic definition of this notion is referred to as "choosing a model."

The idea of an ∞-category

Lean defines an ordinary 1-category as follows:

```
class Quiver (V : Type u) where
  /-- The type of edges/arrows/morphisms between a given source and target. -/
  Hom : V → V → Sort v
```

```
class CategoryStruct (obj : Type u) extends Quiver.{v + 1} obj : Type max u (v + 1) where
  /-- The identity morphism on an object. -/
  id : ∀ X : obj, Hom X X
  /-- Composition of morphisms in a category, written `f ≫ g`. -/
  comp : ∀ {X Y Z : obj}, (X ⟶ Y) → (Y ⟶ Z) → (X ⟶ Z)
```

```
class Category (obj : Type u) extends CategoryStruct.{v} obj : Type max u (v + 1) where
  /-- Identity morphisms are left identities for composition. -/
  id_comp : ∀ {X Y : obj} (f : X ⟶ Y), 𝟙 X ≫ f = f := by aesop_cat
  /-- Identity morphisms are right identities for composition. -/
  comp_id : ∀ {X Y : obj} (f : X ⟶ Y), f ≫ 𝟙 Y = f := by aesop_cat
  /-- Composition in a category is associative. -/
  assoc : ∀ {W X Y Z : obj} (f : W ⟶ X) (g : X ⟶ Y) (h : Y ⟶ Z), (f ≫ g) ≫ h = f ≫ g ≫ h := by
    aesop_cat
```

The idea of an ∞-category is just to

- replace all the types by ∞-groupoids aka homotopy types aka anima, i.e., the information of a topological space encoded by its homotopy groups
- and suitably weaken all the structures and axioms.

"Analytic" ∞-categories in Lean

A popular model encodes an ∞-category as a quasi-category, which Johan Commelin contributed to `Mathlib`:

```
/-- A simplicial set `S` is a *quasicategory* if it satisfies the following horn-filling condition:
for every `n : ℕ` and `0 < i < n`,
every map of simplicial sets `σ₀ : Λ[n, i] → S` can be extended to a map `σ : Δ[n] → S`.

[Kerodon, 003A] -/
class Quasicategory (S : SSet) : Prop where
  hornFilling' : ∀ ⦃n : ℕ⦄ ⦃i : Fin (n+3)⦄ (σ₀ : Λ[n+2, i] ⟶ S)
    (_h0 : 0 < i) (_hn : i < Fin.last (n+2)),
      ∃ σ : Δ[n+2] ⟶ S, σ₀ = hornInclusion (n+2) i ≫ σ
```

where ∞-groupoids can be similarly "coordinatized" as Kan complexes:

```
/-- A simplicial set `S` is a *Kan complex* if it satisfies the following horn-filling condition:
for every nonzero `n : ℕ` and `0 ≤ i ≤ n`,
every map of simplicial sets `σ₀ : Λ[n, i] → S` can be extended to a map `σ : Δ[n] → S`. -/
class KanComplex (S : SSet.{u}) : Prop where
  hornFilling : ∀ ⦃n : ℕ⦄ ⦃i : Fin (n + 2)⦄ (σ₀ : Λ[n + 1, i] ⟶ S),
    ∃ σ : Δ[n + 1] ⟶ S, σ₀ = hornInclusion (n + 1) i ≫ σ
```

But very few results have been formalized with these technical definitions. Indeed, only last week, Joël Riou discovered that the definition of Kan complexes was wrong!

How are quasi-categories ∞-categories?

Recall the idea of an ∞-category is just to replace all the types in an ordinary 1-category

```
class Quiver (V : Type u) where
  /-- The type of edges/arrows/morphisms between a given source and target. -/
  Hom : V → V → Sort v
```

```
class CategoryStruct (obj : Type u) extends Quiver.{v + 1} obj : Type max u (v + 1) where
  /-- The identity morphism on an object. -/
  id : ∀ X : obj, Hom X X
  /-- Composition of morphisms in a category, written `f ≫ g`. -/
  comp : ∀ {X Y Z : obj}, (X ⟶ Y) → (Y ⟶ Z) → (X ⟶ Z)
```

by ∞-groupoids. In particular,

- the maximal sub Kan complex in a quasi-category S defines the ∞-groupoid of objects,
- a certain pullback of the exponential $\mathrm{sHom}(\Delta[1], S)$ defines the ∞-groupoid of arrows between two objects,
- n-ary composition can be shown to be well-defined up to a contractible ∞-groupoid of choices.

None of this has been formalized in `Mathlib`.

Prospects for formalization?

I can imagine three strategies for formalizing the theory of ∞-categories.

Strategy I. Give precise "analytic" definitions of ∞-categorical notions in some model (e.g., using quasi-categories). Prove theorems using the combinatorics of that model.

Strategy II. Axiomatize the category of ∞-categories (e.g., using the notion of ∞-cosmos or something similar). State and prove theorems about ∞-categories in this axiomatic language. To show that this theory is non-vacuous, prove that some model satisfies the axioms and formalize other examples, as desired.

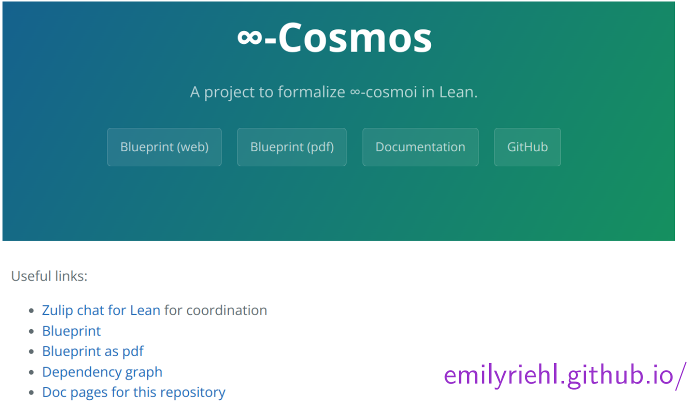
Strategy III. Avoid the technicalities of set-based models by developing the theory of ∞-categories "synthetically," in a domain-specific type theory. Formalization then requires a bespoke proof assistant (e.g., `Rzk`).

2

Formalizing axiomatic ∞-category theory via ∞-cosmoi in `Lean`

An axiomatic theory of ∞-categories in Lean

The ∞-cosmos project — co-led Mario Carneiro, Dominic Verity, and myself — aims to formalize a particular axiomatic theory approach to ∞-category theory Lean's mathematics library `Mathlib`. Pietro Monticone and others helped us set up a blueprint, website, github repository, and Zulip channel to organize the workflow.



emilyriehl.github.io/infinity-cosmos

The idea of the ∞-cosmos project

The aim of the ∞-cosmos project is to leverage the existing 1-category theory, 2-category theory, and enriched category theory libraries in `Lean` to formalize basic ∞-category theory.

This is achieved by developing the theory of ∞-categories more abstractly, using the axiomatic notion of an ∞-cosmos, which is an enriched category whose objects are ∞-categories.

From this we can extract a 2-category whose objects are ∞-categories, whose morphisms are ∞-functors, and whose 2-cells are ∞-natural transformations. The formal theory of ∞-categories (adjunctions, co/limits, Kan extensions) can be defined using this 2-category and some of these notions are in the `Mathlib` already!

Proving that quasi-categories define an ∞-cosmos will be hard, but this tedious verifying of homotopy coherences will only need to be done once rather than in every proof.

Progress

The ∞-cosmos project was launched in September 2024. After adding some background material on enriched category theory, we have formalized the following definition:

1.2.1. Definition (∞-cosmos). An **∞-cosmos** $\mathcal{K}$ is a category that is enriched over quasi-categories,[13] meaning in particular that

- its morphisms $f\colon A \to B$ define the vertices of a quasi-category denoted $\mathsf{Fun}(A,B)$ and referred to as a **functor space**,

that is also equipped with a specified collection of maps that we call **isofibrations** and denote by "$\twoheadrightarrow$" satisfying the following two axioms:

(i) (completeness) The quasi-categorically enriched category $\mathcal{K}$ possesses a terminal object, small products, pullbacks of isofibrations, limits of countable towers of isofibrations, and cotensors with simplicial sets, each of these limit notions satisfying a universal property that is enriched over simplicial sets.[14]

(ii) (isofibrations) The isofibrations contain all isomorphisms and any map whose codomain is the terminal object; are closed under composition, product, pullback, forming inverse limits of towers, and Leibniz cotensors with monomorphisms of simplicial sets; and have the property that if $f\colon A \twoheadrightarrow B$ is an isofibration and X is any object then $\mathsf{Fun}(X,A) \twoheadrightarrow \mathsf{Fun}(X,B)$ is an isofibration of quasi-categories.

A formalized definition of an ∞-cosmos

```
variable (K : Type u) [Category.{v} K] [SimplicialCategory K]
```

```
/-- A `PreInfinityCosmos` is a simplicially enriched category whose hom-spaces are quasi-categories
and whose morphisms come equipped with a special class of isofibrations.-/
class PreInfinityCosmos extends SimplicialCategory K where
  [has_qcat_homs : ∀ {X Y : K}, SSet.Quasicategory (EnrichedCategory.Hom X Y)]
  IsIsofibration : MorphismProperty K
```

```
variable (K : Type u) [Category.{v} K][PreInfinityCosmos.{v} K]
```

```
/-- An `InfinityCosmos` extends a `PreInfinityCosmos` with limit and isofibration axioms..-/
class InfinityCosmos extends PreInfinityCosmos K where
  comp_isIsofibration {A B C : K} (f : A ↠ B) (g : B ↠ C) : IsIsofibration (f.1 ≫ g.1)
  iso_isIsofibration {X Y : K} (e : X ⟶ Y) [IsIso e] : IsIsofibration e
  all_objects_fibrant {X Y : K} (hY : IsConicalTerminal Y) (f : X ⟶ Y) : IsIsofibration f
  [has_products : HasConicalProducts K]
  prod_map_fibrant {γ : Type w} {A B : γ → K} (f : ∀ i, A i ↠ B i) :
    IsIsofibration (Limits.Pi.map (λ i ↦ (f i).1))
  [has_isoFibration_pullbacks {E B A : K} (p : E ↠ B) (f : A ⟶ B)  : HasConicalPullback p.1 f]
  pullback_is_isoFibration {E B A P : K} (p : E ↠ B) (f : A ⟶ B)
    (fst : P ⟶ E) (snd : P ⟶ A) (h : IsPullback fst snd p.1 f) : IsIsofibration snd
  [has_limits_of_towers (F : ℕᵒᵖ ⥤ K) :
    (∀ n : ℕ, IsIsofibration (F.map (homOfLE (Nat.le_succ n)).op)) → HasConicalLimit F]
  has_limits_of_towers_isIsofibration (F : ℕᵒᵖ ⥤ K) (hf) :
    haveI := has_limits_of_towers F hf
    IsIsofibration (limit.π F (.op 0))
  [has_cotensors : HasCotensors K]
  leibniz_cotensor  {U V : SSet} (i : U ⟶ V) [Mono i] {A B : K} (f : A ↠ B) {P : K}
    (fst : P ⟶ U ⋔ A) (snd : P ⟶ V ⋔ B)
    (h : IsPullback fst snd (cotensorCovMap U f.1) (cotensorContraMap i B)) :
    IsIsofibration (h.isLimit.lift <|
      PullbackCone.mk (cotensorContraMap i A) (cotensorCovMap V f.1)
        (cotensor_bifunctoriality i f.1)) --TODO : Prove that these pullbacks exist.
  local_isoFibration {X A B : K} (f : A ↠ B) : Isofibration (toFunMap X f.1)
```

Related contributions to `Mathlib`

One successful aspect of our project is the rapid rate of contributions to `Mathlib`:

- codiscrete categories (Alvaro Belmonte)
- reflexive quivers (Mario Carneiro, Pietro Monticone, Emily Riehl)
- the opposite category of an enriched category (Daniel Carranza)
- a closed monoidal category is enriched in itself (Daniel Carranza, Joël Riou)
- StrictSegal simplicial sets are 2-coskeletal (Mario Carneiro and Joël Riou)
- StrictSegal simplicial sets are quasicategories (Johan Commelin, Emily Riehl, Nick Ward)
- left and right lifting properties (Jack McKoen)
- SSet.hoFunctor, which constructs a category from a simplicial set (Mario Carneiro, Pietro Monticone, Emily Riehl, Joël Riou)
- SimplicialSet (co)skeleton properties (Mario Carneiro, Pietro Monticone, Emily Riehl, Joël Riou)

A key challenge is the extraordinary demands this has placed on Joël Riou as a reviewer.

Challenge: Lean's difficulty with the 1-category of categories

To define the 2-categorical quotient of an ∞-cosmos (WIP), Mario Carneiro and I introduced reflexive quivers

```
/-- A reflexive quiver extends a quiver with a specified arrow `id X : X ⟶ X` for each `X` in its
type of objects. We denote these arrows by `id` since categories can be understood as an extension
of refl quivers.
-/
class ReflQuiver (obj : Type u) extends Quiver.{v} obj : Type max u v where
  /-- The identity morphism on an object. -/
  id : ∀ X : obj, Hom X X
```

and formalized the free category and underlying reflexive quiver adjunction between Cat and ReflQuiv. This is now in `Mathlib`:

```
/--
The adjunction between forming the free category on a reflexive quiver, and forgetting a category
to a reflexive quiver.
-/
nonrec def adj : Cat.freeRefl.{max u v, u} ⊣ ReflQuiv.forget :=
  Adjunction.mkOfUnitCounit {
```

Challenge: Lean's difficulty with the 1-category of categories

In formalizing the free category and underlying reflexive quiver adjunction:

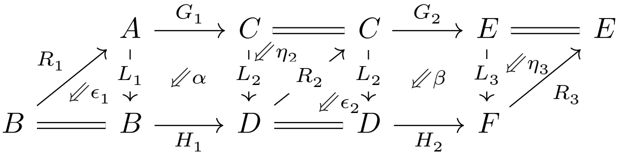
```
left_triangle := by
  ext V
  apply Cat.FreeRefl.lift_unique'
  simp only [id_obj, Cat.free_obj, comp_obj, Cat.freeRefl_obj_α, NatTrans.comp_app,
    forget_obj, whiskerRight_app, associator_hom_app, whiskerLeft_app, id_comp,
    NatTrans.id_app']
  rw [Cat.id_eq_id, Cat.comp_eq_comp]
  simp only [Cat.freeRefl_obj_α, Functor.comp_id]
  rw [← Functor.assoc, ← Cat.freeRefl_naturality, Functor.assoc]
  dsimp [Cat.freeRefl]
  rw [adj.counit.component_eq' (Cat.FreeRefl V)]
  conv =>
    enter [1, 1, 2]
    apply (Quiv.comp_eq_comp (X := Quiv.of _) (Y := Quiv.of _) (Z := Quiv.of _) ..).symm
  rw [Cat.free.map_comp]
  show (_ ⋙ ((Quiv.forget ⋙ Cat.free).map (X := Cat.of _) (Y := Cat.of _)
    (Cat.FreeRefl.quotientFunctor V))) ⋙ _ = _
  rw [Functor.assoc, ← Cat.comp_eq_comp]
  conv => enter [1, 2]; apply Quiv.adj.counit.naturality
  rw [Cat.comp_eq_comp, ← Functor.assoc, ← Cat.comp_eq_comp]
  conv => enter [1, 1]; apply Quiv.adj.left_triangle_components V.toQuiv
  exact Functor.id_comp _
```

Lean was confused by

- what category we're in when objects are type classes
- competing notations for functors
- whiskered commutative diagrams

Challenge: dependent equalities between the 2-cells in a 2-category

On paper, 2-cells in a 2-category compose by pasting:



In `Mathlib`, the 2-cells displayed here belong to dependent types (over their boundary 1-cells and objects) and depending on how the whiskerings are encoded are not obviously composable at all:

e.g., is $R_3 H_2 L_2 \eta_2 G_1 R_1$ composable with $R_3 H_2 \epsilon_2 L_2 G_1 R_1$?

Challenge: dependent equalities between the 2-cells in a 2-category

```
/-- The mates equivalence commutes with vertical composition. -/
theorem mateEquiv_vcomp
    (α : G₁ ⋙ L₂ ⟶ L₁ ⋙ H₁) (β : G₂ ⋙ L₃ ⟶ L₂ ⋙ H₂) :
    (mateEquiv (G := G₁ ⋙ G₂) (H := H₁ ⋙ H₂) adj₁ adj₃) (leftAdjointSquare.vcomp α β) =
      rightAdjointSquare.vcomp (mateEquiv adj₁ adj₂ α) (mateEquiv adj₂ adj₃ β) := by
  unfold leftAdjointSquare.vcomp rightAdjointSquare.vcomp mateEquiv
  ext b
  simp only [comp_obj, Equiv.coe_fn_mk, whiskerLeft_comp, whiskerLeft_twice, whiskerRight_comp,
    assoc, comp_app, whiskerLeft_app, whiskerRight_app, id_obj, Functor.comp_map,
    whiskerRight_twice]
  slice_rhs 1 4 => rw [← assoc, ← assoc, ← unit_naturality (adj₃)]
  rw [L₃.map_comp, R₃.map_comp]
  slice_rhs 2 4 =>
    rw [← R₃.map_comp, ← R₃.map_comp, ← assoc, ← L₃.map_comp, ← G₂.map_comp, ← G₂.map_comp]
    rw [← Functor.comp_map G₂ L₃, β.naturality]
  rw [(L₂ ⋙ H₂).map_comp, R₃.map_comp, R₃.map_comp]
  slice_rhs 4 5 =>
    rw [← R₃.map_comp, Functor.comp_map L₂ _, ← Functor.comp_map _ L₂, ← H₂.map_comp]
    rw [adj₂.counit.naturality]
  simp only [comp_obj, Functor.comp_map, map_comp, id_obj, Functor.id_map, assoc]
  slice_rhs 4 5 =>
    rw [← R₃.map_comp, ← H₂.map_comp, ← Functor.comp_map _ L₂, adj₂.counit.naturality]
  simp only [comp_obj, id_obj, Functor.id_map, map_comp, assoc]
  slice_rhs 3 4 =>
    rw [← R₃.map_comp, ← H₂.map_comp, left_triangle_components]
  simp only [map_id, id_comp]
```

In the 2-category Cat, I formalized a proof that the unit η_2 and counit ϵ_2 cancel, but not via a 2-categorical pasting argument. As a result, this proof does not extend to a general 2-category.

Challenge: dependent equalities between the 2-cells in a 2-category

```
/-- The mates equivalence commutes with vertical composition. -/
theorem mateEquiv_vcomp (α : g₁ ≫ l₂ ⟶ l₁ ≫ h₁) (β : g₂ ≫ l₃ ⟶ l₂ ≫ h₂) :
    mateEquiv adj₁ adj₃ (leftAdjointSquare.vcomp α β) =
      rightAdjointSquare.vcomp (mateEquiv adj₁ adj₂ α) (mateEquiv adj₂ adj₃ β) := by
  dsimp only [leftAdjointSquare.vcomp, mateEquiv_apply, rightAdjointSquare.vcomp]
  symm
  calc
    _ = 𝟙 _ ⊗≫ r₁ ◁ g₁ ◁ adj₂.unit ▷ g₂ ⊗≫ r₁ ◁ α ▷ r₂ ▷ g₂ ⊗≫
          ((adj₁.counit ▷ (h₁ ≫ r₂ ≫ g₂ ≫ 𝟙 e)) ≫ 𝟙 b ◁ (h₁ ◁ r₂ ◁ g₂ ◁ adj₃.unit)) ⊗≫
            h₁ ◁ r₂ ◁ β ▷ r₃ ⊗≫ h₁ ◁ adj₂.counit ▷ h₂ ▷ r₃ ⊗≫ 𝟙 _ := by
      bicategory
    _ = 𝟙 _ ⊗≫ r₁ ◁ g₁ ◁ adj₂.unit ▷ g₂ ⊗≫
          (r₁ ◁ (α ▷ (r₂ ≫ g₂ ≫ 𝟙 e) ≫ (l₁ ≫ h₁) ◁ r₂ ◁ g₂ ◁ adj₃.unit)) ⊗≫
            ((adj₁.counit ▷ (h₁ ≫ r₂) ▷ (g₂ ≫ l₃) ≫ (𝟙 b ≫ h₁ ≫ r₂) ◁ β) ▷ r₃) ⊗≫
              h₁ ◁ adj₂.counit ▷ h₂ ▷ r₃ ⊗≫ 𝟙 _ := by
      rw [← whisker_exchange]
      bicategory
    _ = 𝟙 _ ⊗≫ r₁ ◁ g₁ ◁ (adj₂.unit ▷ (g₂ ≫ 𝟙 e) ≫ (l₂ ≫ r₂) ◁ g₂ ◁ adj₃.unit) ⊗≫
          (r₁ ◁ (α ▷ (r₂ ≫ g₂ ≫ l₃) ≫ (l₁ ≫ h₁) ◁ r₂ ◁ β) ▷ r₃) ⊗≫
            (adj₁.counit ▷ h₁ ▷ (r₂ ≫ l₂) ≫ (𝟙 b ≫ h₁) ◁ adj₂.counit) ▷ h₂ ▷ r₃ ⊗≫ 𝟙 _ := by
      rw [← whisker_exchange, ← whisker_exchange]
      bicategory
    _ = 𝟙 _ ⊗≫ r₁ ◁ g₁ ◁ g₂ ◁ adj₃.unit ⊗≫
          r₁ ◁ g₁ ◁ (adj₂.unit ▷ (g₂ ≫ l₃) ≫ (l₂ ≫ r₂) ◁ β) ▷ r₃ ⊗≫
            r₁ ◁ (α ▷ (r₂ ≫ l₂) ≫ (l₁ ≫ h₁) ◁ adj₂.counit) ▷ h₂ ▷ r₃ ⊗≫
              adj₁.counit ▷ h₁ ▷ h₂ ▷ r₃ ⊗≫ 𝟙 _ := by
      rw [← whisker_exchange, ← whisker_exchange, ← whisker_exchange]
      bicategory
    _ = 𝟙 _ ⊗≫ r₁ ◁ g₁ ◁ g₂ ◁ adj₃.unit ⊗≫ r₁ ◁ g₁ ◁ β ▷ r₃ ⊗≫
          ((r₁ ≫ g₁) ◁ leftZigzag adj₂.unit adj₂.counit ▷ (h₂ ≫ r₃)) ⊗≫
            r₁ ◁ α ▷ h₂ ▷ r₃ ⊗≫ adj₁.counit ▷ h₁ ▷ h₂ ▷ r₃ ⊗≫ 𝟙 _ := by
      rw [← whisker_exchange, ← whisker_exchange]
      bicategory
    _ = _ := by
      rw [adj₂.left_triangle]
      bicategory
```

After describing this challenge two weeks ago, Yuma Mizuno leveraged his bicategory tactic to formalize the desired generalization.

It would be great to extend this tactic to automate the intermediate steps in this calculation.

Contributors to the ∞-cosmos project

So far formalizations (and preliminary mathematical work) have been contributed by:

Dagur Asgeirsson, Alvaro Belmonte, Mario Carneiro, Daniel Carranza, Johan Commelin, Jon Eugster, Jack McKoen, Yuma Mizuno, Pietro Monticone, Matej Penciak, Nima Rasekh, Emily Riehl, Joël Riou, Joseph Tooby-Smith, Adam Topaz, Dominic Verity, Nick Ward, and Zeyi Zhao.

Anyone is welcome to join us!

emilyriehl.github.io/infinity-cosmos

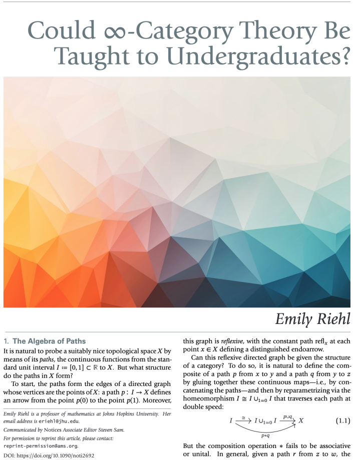
3

Formalizing synthetic ∞-category theory in simplicial HoTT in `Rzk`

Could ∞-category theory be taught to undergraduates?

Recall ∞-categories are like categories where all the sets are replaced by ∞-groupoids:

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



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 unit interval $I \coloneqq [0,1] \subset \mathbb{R}$ 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 $p : I \to X$ defines an arrow from the point $p(0)$ to the point $p(1)$. Moreover, this graph is *reflexive*, with the constant path refl_x at each point $x \in X$ defining a distinguished endoarrow.

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 $I \cong I \cup_{1=0} I$ that traverses each path at double speed:

$$I \xrightarrow{\cong} I \cup_{1=0} I \xrightarrow{p \cup q} X, \quad p * q : I \to X \tag{1.1}$$

But the composition operation $*$ fails to be associative or unital. In general, given a path r from z to w, the

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

Communicated by Notices *Associate Editor Steven Sam.*

For permission to reprint this article, please contact: reprint-permission@ams.org.

DOI: https://doi.org/10.1090/noti2692

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 based on Martin-Löf's dependent type theory where the primative types have "higher structure" such as

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

∞-categories in simplicial 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 ∞-categories, Higher Structures 1(1):116–193, 2017

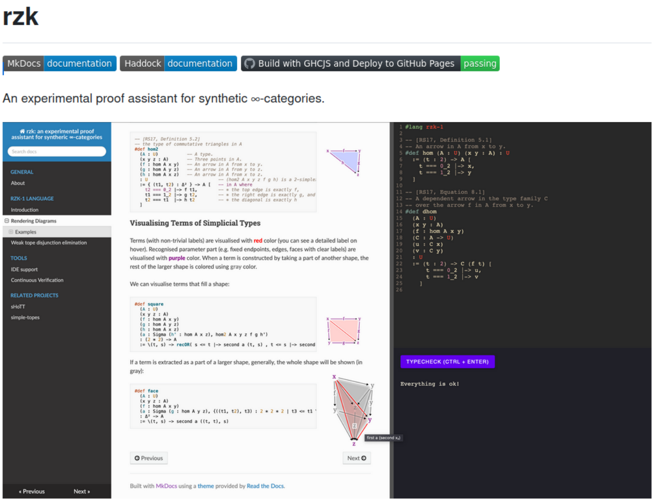
each type A also has a family of hom types $\text{Hom}_A(x, y)$ over $x, y : A$ whose terms $f : \text{Hom}_A(x, y)$ are called arrows.

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?

An experimental proof assistant Rzk for ∞-category theory



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/fizruk/sHoTT and https://github.com/emilyriehl/yoneda. `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 tope 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/fizruk/simple-topes. `simple-topes` supports used-defined cubes, topes, and tope 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

Extension types in simplicial homotopy type theory

Formation rule for extension types

$$\frac{\Phi \subset \Psi \text{ shape} \qquad A \text{ type} \qquad a : \Phi \to A}{\left\langle \begin{array}{ccc} \Phi & \xrightarrow{a} & A \\ \downarrow & \nearrow & \\ \Psi & & \end{array} \right\rangle \text{ type}}$$

A term $f : \left\langle \begin{array}{ccc} \Phi & \xrightarrow{a} & A \\ \downarrow & \nearrow & \\ \Psi & & \end{array} \right\rangle$ defines

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

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

Hom types

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

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

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

The type $\mathrm{Hom}_A(x,y)$ as the mapping ∞-groupoid in A from x to y.

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.,

$$\left\langle \begin{array}{ccc} \Lambda^2_1 & \xrightarrow{[f,g]} & A \\ \downarrow & \nearrow & \\ \Delta^2 & & \end{array} \right\rangle \quad \text{is contractible.}^{a}$$

[a] A type C is contractible just when $\sum_{c:C} \prod_{x:C} c = x$.

By contractibility, $\left\langle \begin{array}{ccc} \Lambda^2_1 & \xrightarrow{[f,g]} & A \\ \downarrow & \nearrow & \\ \Delta^2 & & \end{array} \right\rangle$ has a unique inhabitant $\text{comp}_{f,g} : \Delta^2 \to A$.

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

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

which we denote by id_x and call the identity arrow.

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

$$\lambda(s,t).f(t) : \left\langle \begin{array}{ccc} \Lambda^2_1 & \xrightarrow{[\mathsf{id}_x, f]} & A \\ \downarrow & \nearrow & \\ \Delta^2 & & \end{array} \right\rangle$$

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

Stating the Yoneda lemma

Let A be a pre-∞-category and fix $a, b : A$.

Yoneda lemma. Evaluation at the identity defines an equivalence

$$\mathsf{evid} := \lambda\phi.\phi_a(\mathsf{id}_a) : \left(\prod_{z:A} \mathsf{Hom}_A(z,a) \to \mathsf{Hom}_A(z,b)\right) \to \mathsf{Hom}_A(a,b)$$

While terms $\phi : \prod_{z:A} \mathsf{Hom}_A(z,a) \to \mathsf{Hom}_A(z,b)$ are just families of maps

$$\phi_z : \mathsf{Hom}_A(z,a) \to \mathsf{Hom}_A(z,b)$$

indexed by terms $z : A$ such families are automatically natural:

Prop. Any family of maps $\phi : \prod_{z:A} \hom_A(z,a) \to \hom_A(z,b)$ is natural:

for any $g : \hom_A(y,a)$ and $h : \hom_A(x,y)$

$$\phi_y(g) \circ h = \phi_x(g \circ h).$$

Proving the Yoneda lemma

Let A be a pre-∞-category and fix $a, b : A$.

Yoneda lemma. Evaluation at the identity defines an equivalence

$$\mathsf{evid} := \lambda\phi.\phi_a(\mathsf{id}_a) : \left(\prod_{z:A} \mathsf{Hom}_A(z,a) \to \mathsf{Hom}_A(z,b)\right) \to \mathsf{Hom}_A(a,b)$$

The proof is (a simplification of) the standard argument for 1-categories!

Proof: Define an inverse map by

$$\mathsf{yon} := \lambda v.\lambda x.\lambda f.f \circ v : \mathsf{Hom}_A(a,b) \to \left(\prod_{z:A} \mathsf{Hom}_A(z,a) \to \mathsf{Hom}_A(z,b)\right).$$

By definition, $\mathsf{evid} \circ \mathsf{yon}(v) := v \circ \mathsf{id}_a$, and since $v \circ \mathsf{id}_a = v$, so $\mathsf{evid} \circ \mathsf{yon}(v) = v$. Similarly, by definition, $\mathsf{yon} \circ \mathsf{evid}(\phi)_z(f) := \phi_a(\mathsf{id}_a) \circ f$. By naturality of ϕ and another identity law $\phi_a(\mathsf{id}_a) \circ f = \phi_z(\mathsf{id}_a \circ f) = \phi_z(f)$, so $\mathsf{yon} \circ \mathsf{evid}(\phi)_z(f) = \phi_z(f)$. □

A formalized proof of the ∞-categorical Yoneda lemma

Nikolai Kudasov, Jonathan Weinberger, and I formalized the ∞-Yoneda lemma:

For any pre-∞-category `A` terms `a b : A`, the contravariant Yoneda lemma provides an equivalence between the type `(z : A) → Hom A z a → Hom A z b` of natural transformations and the type `Hom A a b`.

One of the maps in this equivalence is evaluation at the identity. The inverse map makes use of the contravariant transport operation.

The following map, `contra-evid` evaluates a natural transformation out of a representable functor at the identity arrow.

```
#def Contra-evid
  ( A : U)
  ( a b : A)
  : ( ( z : A) → Hom A z a → Hom A z b) → Hom A a b
  := \ ϕ → ϕ a (Id-hom A a)
```

The inverse map only exists for pre-∞-categories.

```
#def Contra-yon
  ( A : U)
  ( is-pre-∞-category-A : Is-pre-∞-category A)
  ( a b : A)
  : Hom A a b → ((z : A) → Hom A z a → Hom A z b)
  := \ v z f → Comp-is-pre-∞-category A is-pre-∞-category-A z a b f v
```

emilyriehl.github.io/yoneda/

A formalized proof of the ∞-categorical Yoneda lemma

It remains to show that the Yoneda maps are inverses. One retraction is straightforward:

```
#def Contra-evid-yon
  ( A : U)
  ( is-pre-∞-category-A : Is-pre-∞-category A)
  ( a b : A)
  ( v : Hom A a b)
  : Contra-evid A a b (Contra-yon A is-pre-∞-category-A a b v) = v
  :=
    Id-comp-is-pre-∞-category A is-pre-∞-category-A a b v
```

The other composite carries `ϕ` to an a priori distinct natural transformation. We first show that these are pointwise equal at all `x : A` and `f : Hom A x a` in two steps.

```
#def Contra-yon-evid-twice-pointwise
  ( ϕ : (z : A) → Hom A z a → Hom A z b)
  ( x : A)
  ( f : Hom A x a)
  : ( ( Contra-yon A is-pre-∞-category-A a b)
        ( ( Contra-evid A a b) ϕ)) x f = ϕ x f
  :=
    concat
      ( Hom A x b)
      ( ( ( Contra-yon A is-pre-∞-category-A a b)
            ( ( Contra-evid A a b) ϕ)) x f)
      ( ϕ x (Comp-is-pre-∞-category A is-pre-∞-category-A x a a f (Id-hom A a)))
      ( ϕ x f)
      ( Naturality-contravariant-fiberwise-representable-transformation
          A is-pre-∞-category-A a b x a f (Id-hom A a) ϕ)
      ( ap
        ( Hom A x a)
        ( Hom A x b)
        ( Comp-is-pre-∞-category A is-pre-∞-category-A x a a
          f (Id-hom A a))
        ( f)
        ( ϕ x)
        ( Comp-id-is-pre-∞-category A is-pre-∞-category-A x a f))
```

Contributors to the simplicial HoTT library

So far formalizations to the broader project of formalizing synthetic ∞-category theory (and work on the proof assistant `Rzk`) have been contributed by:

Abdelrahman Aly Abounegm, Fredrik Bakke, César Bardomiano Martínez, Jonathan Campbell, Robin Carlier, Theofanis Chatzidiamantis-Christoforidis, Aras Ergus, Matthias Hutzler, Nikolai Kudasov, Kenji Maillard, David Martínez Carpena, Stiéphen Pradal, Nima Rasekh, Emily Riehl, Florrie Verity, Tashi Walde, and Jonathan Weinberger.

Anyone is welcome to join us!

rzk-lang.github.io/sHoTT

You could contribute to either project!

Papers:

- Emily Riehl, Could ∞-category theory be taught to undergraduates?, Notices of the AMS 70(5):727–736, May 2023; arXiv:2302.07855
- Nikolai Kudasov, Emily Riehl, Jonathan Weinberger, Formalizing the ∞-categorical Yoneda lemma, CPP 2024: 274–290; arXiv:2309.08340

Formalization:

- Johan Commelin, Kim Morrison, Joël Riou, Adam Topaz, a nascent theory of quasi-categories in Mathlib, AlgebraicTopology/SimplicialSet/Quasicategory
- Mario Carneiro, Emily Riehl, and Dominic Verity, a blueprint of the model-independent theory, emilyriehl.github.io/infinity-cosmos
- Nikolai Kudasev et al, synthetic ∞-categories in simplicial homotopy type theory, rzk-lang.github.io/sHoTT/

Thank you!