Emily Riehl

Johns Hopkins University

Could we teach ∞-category theory to undergraduates or to a computer?

London Mathematical Society Hardy Lecture

1

What is category theory for?

What is category theory for?

Corollary. For a function $f \colon A \to B$, the inverse image function $f^{-1} \colon \mathfrak{P}(B) \to \mathfrak{P}(A)$ between the powersets of A and B preserves both unions and intersections, while the direct image function $f_* \colon \mathfrak{P}(A) \to \mathfrak{P}(B)$ preserves only unions.

What is category theory for?

Corollary. For a function $f: A \to B$, the inverse image function $f^{-1}: \mathfrak{P}(B) \to \mathfrak{P}(A)$ between the powersets of A and B preserves both unions and intersections, while the direct image function $f_*: \mathfrak{P}(A) \to \mathfrak{P}(B)$ preserves only unions.

Corollary. For vector spaces U, V, and W, $U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)$.

Corollary. For groups G, H, K, $G \times (H * K) \cong (G \times H) * (G \times K)$.

What is category theory for?

Corollary. For a function $f\colon A \to B$, the inverse image function $f^{-1}\colon \mathfrak{P}(B) \to \mathfrak{P}(A)$ between the powersets of A and B preserves both unions and intersections, while the direct image function $f_*\colon \mathfrak{P}(A) \to \mathfrak{P}(B)$ preserves only unions.

Corollary. For vector spaces U, V, and W, $U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)$.

Corollary. For groups G, H, K, $G \times (H * K) \cong (G \times H) * (G \times K)$.

All of these results are true for the same reason and have the same proof:

Theorem. Left adjoint functors preserve coproducts.
Dually right adjoint functors preserve products.

A categorical proof in linear algebra

Corollary. For any vector spaces U, V, and W, $U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)$.

A categorical proof in linear algebra

Corollary. For any vector spaces U, V, and W, $U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)$.

The proof uses the Yoneda lemma: to show $U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)$ it suffices to give natural correspondences between linear maps to another vector space X:

$$U \otimes (V \oplus W) \to X \qquad \leftrightsquigarrow \qquad (U \otimes V) \oplus (U \otimes W) \to X.$$

A categorical proof in linear algebra

Corollary. For any vector spaces U, V, and W, $U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)$.

The proof uses the Yoneda lemma: to show $U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)$ it suffices to give natural correspondences between linear maps to another vector space X:

$$U \otimes (V \oplus W) \to X \qquad \leftrightsquigarrow \qquad (U \otimes V) \oplus (U \otimes W) \to X.$$

Proof:

$$\begin{array}{c}
U \otimes (V \oplus W) \to X \\
\hline
V \oplus W \to \mathrm{Lin}(U, X) \\
\hline
V \to \mathrm{Lin}(U, X) \quad \text{and} \quad W \to \mathrm{Lin}(U, X) \\
\hline
U \otimes V \to X \quad \text{and} \quad U \otimes W \to X \\
\hline
(U \otimes V) \oplus (U \otimes W) \to X
\end{array}
\begin{array}{l}
\\
\textsc{by currying (adjunction)} \\
\textsc{by cases (coproduct)} \\
\textsc{by currying (adjunction)} \\
\textsc{by cases (coproduct)} \\
\\
\end{array}$$

A categorical proof in linear algebra

Corollary. For any vector spaces U, V, and W, $U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)$.

The proof uses the Yoneda lemma: to show $U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)$ it suffices to give natural correspondences between linear maps to another vector space X:

$$U \otimes (V \oplus W) \to X \qquad \leftrightsquigarrow \qquad (U \otimes V) \oplus (U \otimes W) \to X.$$

Proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{U \otimes (V \oplus W) \to X}{V \oplus W \to \mathsf{Lin}(U, X)}\ \text{BY CURRYING (ADJUNCTION)}}{V \to \mathsf{Lin}(U, X) \quad \text{and} \quad W \to \mathsf{Lin}(U, X)}\ \text{BY CASES (COPRODUCT)}}{U \otimes V \to X \quad \text{and} \quad U \otimes W \to X}\ \text{BY CURRYING (ADJUNCTION)}}{(U \otimes V) \oplus (U \otimes W) \to X}\ \text{BY CASES (COPRODUCT)}$$

The isomorphism is explicit: letting X equal $U \otimes (V \oplus W)$ or $(U \otimes V) \oplus (U \otimes W)$, the identity maps define linear maps $U \otimes (V \oplus W) \rightleftarrows (U \otimes V) \oplus (U \otimes W)$, which are inverses (by naturality).

Abstraction and understanding

Since left adjoints preserve coproducts:

the direct image preserves unions, the tensor product distributes over direct sums, the cartesian product distributes over free products, ...

Abstraction and understanding

Since left adjoints preserve coproducts:

the direct image preserves unions, the tensor product distributes over direct sums, the cartesian product distributes over free products, ...

Abstractions from category theory are used to:

- clarify arguments (removing inessential details, shortening proofs)
- generalize constructions and proofs to other settings
- upload more complicated mathematical ideas into one's working memory

Abstraction and understanding

Since left adjoints preserve coproducts:

the direct image preserves unions, the tensor product distributes over direct sums, the cartesian product distributes over free products, ...

Abstractions from category theory are used to:

- clarify arguments (removing inessential details, shortening proofs)
- generalize constructions and proofs to other settings
- upload more complicated mathematical ideas into one's working memory

Category theory defines the language of (much of) 20th century mathematics. Modern algebraic geometry, algebraic topology, representation theory, and certain aspects of mathematical physics are all expressed in the language of category theory.

Abstraction and understanding

Since left adjoints preserve coproducts:

the direct image preserves unions, the tensor product distributes over direct sums, the cartesian product distributes over free products, ...

Abstractions from category theory are used to:

- clarify arguments (removing inessential details, shortening proofs)
- generalize constructions and proofs to other settings
- upload more complicated mathematical ideas into one's working memory

Category theory defines the language of (much of) 20th century mathematics. Modern algebraic geometry, algebraic topology, representation theory, and certain aspects of mathematical physics are all expressed in the language of category theory.

Without categorical language, we couldn't (easily) define spectral sequences, simplicial sets, cohomology theories, sheaves, schemes, topological quantum field theories, ...

2

What is ∞-category theory for?

What are ∞-categories?

> *[A category] frames a possible template for any mathematical theory: the theory should have nouns and verbs, i.e., objects, and morphisms, and there should be an explicit notion of composition related to the morphisms.*
>
> *—Barry Mazur, "When is one thing equal to some other thing?"*

What are ∞-categories?

> *[A category] frames a possible template for any mathematical theory: the theory should have nouns and verbs, i.e., objects, and morphisms, and there should be an explicit notion of composition related to the morphisms.*
>
> *—Barry Mazur, "When is one thing equal to some other thing?"*

An ∞-category frames a template with nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, interjections,...

What are ∞-categories?

> *[A category] frames a possible template for any mathematical theory: the theory should have nouns and verbs, i.e., objects, and morphisms, and there should be an explicit notion of composition related to the morphisms.*
>
> —*Barry Mazur, "When is one thing equal to some other thing?"*

An ∞-category frames a template with nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, interjections,... which has:

- objects x and 1-morphisms between them $w \xrightarrow{f} x$

What are ∞-categories?

> *[A category] frames a possible template for any mathematical theory: the theory should have nouns and verbs, i.e., objects, and morphisms, and there should be an explicit notion of composition related to the morphisms.*
>
> *—Barry Mazur, "When is one thing equal to some other thing?"*

An ∞-category frames a template with nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, interjections,... which has:

- objects x and 1-morphisms between them $w \xrightarrow{f} x$
- composition witnessed by invertible 2-morphisms

$$\begin{tikzcd} & x \arrow[dr, "g"] \arrow[d, phantom, "\wr\|\alpha"] & \\ w \arrow[ur, "f"] \arrow[rr, dashed, "g\circ f"'] & & y \end{tikzcd}$$

What are ∞-categories?

> *[A category] frames a possible template for any mathematical theory: the theory should have nouns and verbs, i.e., objects, and morphisms, and there should be an explicit notion of composition related to the morphisms.*
>
> —*Barry Mazur, "When is one thing equal to some other thing?"*

An ∞-category frames a template with nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, interjections,... which has:

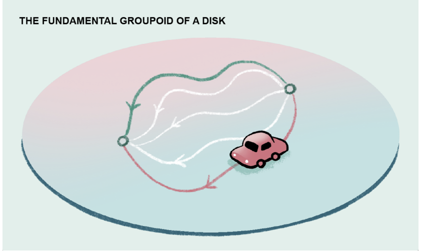
- objects x and 1-morphisms between them $w \xrightarrow{f} x$
- composition witnessed by invertible 2-morphisms
$$\begin{tikzcd} & x \arrow[dr, "g"] \arrow[d, phantom, "\wr|\alpha"] & \\ w \arrow[ur, "f"] \arrow[rr, dashed, "g\circ f"'] & {} & y \end{tikzcd}$$
- associativity
$$\begin{tikzcd} w \arrow[rrrr, dashed, "h\circ g\circ f"] \arrow[drr, "f"] \arrow[ddrr, dashed, "g\circ f"'] & & \wr|\gamma & & z \\ & \wr|\alpha & x \arrow[urr, dashed, "h\circ g"'] \arrow[d, "g"] & \wr|\beta & \\ & & y \arrow[uurr, "h"'] & & \end{tikzcd}$$
witnessed by invertible 3-morphisms

with these witnesses coherent up to invertible morphisms all the way up.

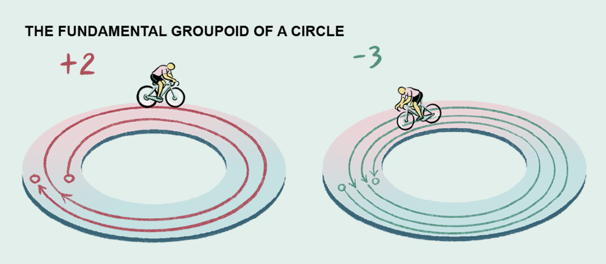
Example: homotopy types as groupoids

The fundamental groupoid of a topological space is the category whose

- objects are points in the space
- arrows are paths in the space up to based homotopy



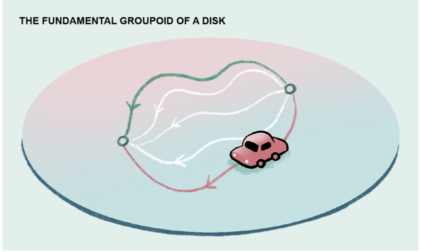
Images by Matteo Farinella



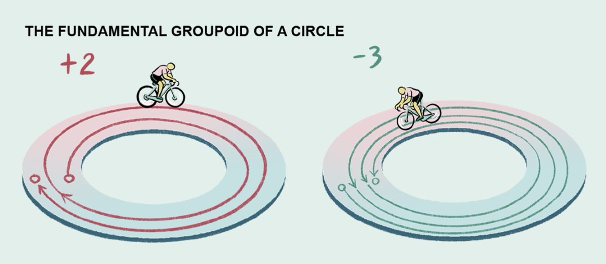
Example: homotopy types as groupoids

The fundamental groupoid of a topological space is the category whose

- objects are points in the space
- arrows are paths in the space up to based homotopy



Images by Matteo Farinella

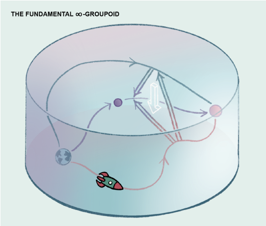


Problem: the fundamental groupoid, while easy to define, does not see "higher structure":

for all $n > 1$, the fundamental groupoid of S^n is trivial!

Example: homotopy types as ∞-groupoids

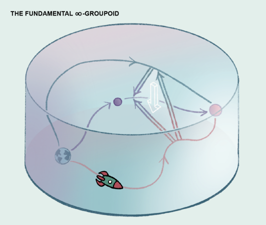
The full homotopy type of a topological space is captured by its fundamental ∞-groupoid whose



- objects are points, 1-morphisms are paths,
- 2-morphisms are homotopies between paths,
- 3-morphisms are homotopies between homotopies, ...

Example: homotopy types as ∞-groupoids

The full homotopy type of a topological space is captured by its fundamental ∞-groupoid whose

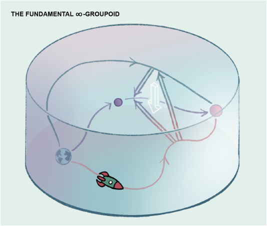


- objects are points, 1-morphisms are paths,
- 2-morphisms are homotopies between paths,
- 3-morphisms are homotopies between homotopies, ...

The quotient homotopy category recovers the fundamental groupoid of points and based homotopy classes of paths.

Example: homotopy types as ∞-groupoids

The full homotopy type of a topological space is captured by its fundamental ∞-groupoid whose



- objects are points, 1-morphisms are paths,
- 2-morphisms are homotopies between paths,
- 3-morphisms are homotopies between homotopies, ...

The quotient homotopy category recovers the fundamental groupoid of points and based homotopy classes of paths.

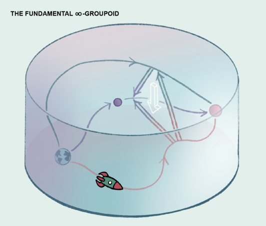
Grothendieck's Homotopy Hypothesis:
The fundamental ∞-groupoid defines an equivalence between spaces (up to weak homotopy equivalence) and ∞-groupoids (up to equivalence).

Scholze: an ∞-groupoid is an anima, the "soul" of a space.

∞-categories and their quotient 1-categories

The fundamental ∞-groupoid of a topological space is the ∞-category whose

- objects are points, 1-morphisms are paths,
- 2-morphisms are homotopies between paths,
- 3-morphisms are homotopies between homotopies, ...

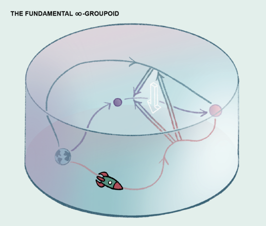


The quotient homotopy category recovers the fundamental groupoid of points and based homotopy classes of paths.

∞-categories and their quotient 1-categories

The fundamental ∞-groupoid of a topological space is the ∞-category whose

- objects are points, 1-morphisms are paths,
- 2-morphisms are homotopies between paths,
- 3-morphisms are homotopies between homotopies, ...



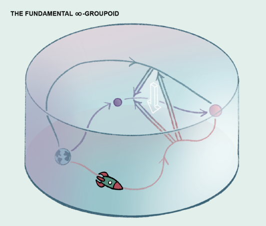
The quotient homotopy category recovers the fundamental groupoid of points and based homotopy classes of paths. Similarly:

- The derived category of a ring is the quotient homotopy category of the ∞-category of chain complexes.
- The classical homotopy category of spaces is the quotient homotopy category of the ∞-category of spaces.

∞-categories and their quotient 1-categories

The fundamental ∞-groupoid of a topological space is the ∞-category whose

- objects are points, 1-morphisms are paths,
- 2-morphisms are homotopies between paths,
- 3-morphisms are homotopies between homotopies, ...



The quotient homotopy category recovers the fundamental groupoid of points and based homotopy classes of paths. Similarly:

- The derived category of a ring is the quotient homotopy category of the ∞-category of chain complexes.
- The classical homotopy category of spaces is the quotient homotopy category of the ∞-category of spaces.

Very roughly, an "∞-category" is a weak infinite-dimensional analog of an ordinary 1-dimensional category, with objects, 1-morphisms, and invertible higher morphisms with weak composition, associativity, and identities.

What is ∞-category theory for?

- The derived category of a ring is the quotient homotopy category of the ∞-category of chain complexes.
- The classical homotopy category of spaces is the quotient homotopy category of the ∞-category of spaces.

What is ∞-category theory for?

- The derived category of a ring is the quotient homotopy category of the ∞-category of chain complexes.
- The classical homotopy category of spaces is the quotient homotopy category of the ∞-category of spaces.

Problem: Key constructions, such as the mapping cone of an chain map, are not colimits in the derived category. Homotopy colimits are not colimits in the homotopy category!

$$
\begin{tikzcd}
A_\bullet \arrow[r, "f_\bullet"] \arrow[d] \arrow[dr, phantom, "\sim\ulcorner" near end] & B_\bullet \arrow[d, dashed] \\
0 \arrow[r, dashed] & C(f)_\bullet
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
X \arrow[r, "f"] \arrow[d] \arrow[dr, phantom, "\sim\ulcorner" near end] & Y \arrow[d, dashed] \\
* \arrow[r, dashed] & C(f)
\end{tikzcd}
$$

What is ∞-category theory for?

- The derived category of a ring is the quotient homotopy category of the ∞-category of chain complexes.
- The classical homotopy category of spaces is the quotient homotopy category of the ∞-category of spaces.

Problem: Key constructions, such as the mapping cone of an chain map, are not colimits in the derived category. Homotopy colimits are not colimits in the homotopy category!

$$\begin{tikzcd} A_\bullet \arrow[r, "f_\bullet"] \arrow[d] & B_\bullet \arrow[d, dashed] \\ 0 \arrow[r, dashed] \arrow[ur, phantom, "\sim \ulcorner"] & C(f)_\bullet \end{tikzcd} \qquad \begin{tikzcd} X \arrow[r, "f"] \arrow[d] & Y \arrow[d, dashed] \\ * \arrow[r, dashed] \arrow[ur, phantom, "\sim \ulcorner"] & C(f) \end{tikzcd}$$

$\rightsquigarrow$ Hence the theorem left adjoints preserves colimits does not apply.

What is ∞-category theory for?

- The derived category of a ring is the quotient homotopy category of the ∞-category of chain complexes.
- The classical homotopy category of spaces is the quotient homotopy category of the ∞-category of spaces.

Problem: Key constructions, such as the mapping cone of an chain map, are not colimits in the derived category. Homotopy colimits are not colimits in the homotopy category!

$$\begin{tikzcd} A_\bullet \arrow[r, "f_\bullet"] \arrow[d] & B_\bullet \arrow[d, dashed] \\ 0 \arrow[r, dashed] \arrow[ur, phantom, "\sim\ulcorner"] & C(f)_\bullet \end{tikzcd} \qquad \begin{tikzcd} X \arrow[r, "f"] \arrow[d] & Y \arrow[d, dashed] \\ * \arrow[r, dashed] \arrow[ur, phantom, "\sim\ulcorner"] & C(f) \end{tikzcd}$$

$\rightsquigarrow$ Hence the theorem left adjoints preserves colimits does not apply.

Solution: Replace the derived category by the ∞-category of cochain complexes, where the mapping cone is a colimit in the ∞-categorical sense.

$\rightsquigarrow$ Now the analogous ∞-categorical theorem left adjoints preserve colimits applies!

∞-categories in set theory

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

$$\text{sets} :: \infty\text{-groupoids}$$
$$\text{categories} :: \infty\text{-categories}$$

∞-categories in set theory

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

$$\text{sets} :: \infty\text{-groupoids}$$
$$\text{categories} :: \infty\text{-categories}$$

Where

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

∞-categories in set theory

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

$$\text{sets} :: \infty\text{-groupoids}$$
$$\text{categories} :: \infty\text{-categories}$$

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

∞-categories in set theory

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

$$\text{sets} :: \infty\text{-groupoids}$$
$$\text{categories} :: \infty\text{-categories}$$

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).

∞-categories in set theory

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

$$\text{sets} :: \infty\text{-groupoids}$$
$$\text{categories} :: \infty\text{-categories}$$

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 define within set theory.

Definitions of ∞-categories

∞-categories are 1-categories in which all the sets have been replaced by ∞-groupoids.

An ∞-category is presented by a quasi-category, which is a simplicial set

$$\begin{tikzcd}
X_0 \arrow[r, shift left=2] & X_1 \arrow[l, shift left=4] \arrow[l] \arrow[r, shift left=4] \arrow[r] & X_2 \arrow[l, shift left=6] \arrow[l, shift left=2] \arrow[l, shift right=2] & \cdots
\end{tikzcd}$$

in which every inner horn has a filler.

An ∞-category is presented by a complete Segal spaces, which is a bisimplicial set

$$\begin{tikzcd}
\vdots \arrow[d, shift right=6] \arrow[d, shift right=2] \arrow[d, shift left=2] \arrow[d, shift left=6] & \vdots \arrow[d, shift right=6] \arrow[d, shift right=2] \arrow[d, shift left=2] \arrow[d, shift left=6] & \iddots \\
X_{01} \arrow[u, shift right=4] \arrow[u, shift left=4] \arrow[d, shift right=4] \arrow[d, shift left=4] \arrow[r, shift left=2] & X_{11} \arrow[l, shift left=4] \arrow[l] \arrow[u, shift right=4] \arrow[u, shift left=4] \arrow[d, shift right=4] \arrow[d, shift left=4] \arrow[r, shift left=4] \arrow[r] & \cdots \arrow[l, shift left=6] \arrow[l, shift left=2] \arrow[l, shift right=2] \\
X_{00} \arrow[u] \arrow[r, shift left=2] & X_{10} \arrow[l, shift left=4] \arrow[l] \arrow[u] \arrow[r, shift left=4] \arrow[r] & \cdots \arrow[l, shift left=6] \arrow[l, shift left=2] \arrow[l, shift right=2]
\end{tikzcd}$$

that is Reedy fibrant and in which the Segal and completeness maps are equivalences.

Challenge: everything takes too long to make technically precise

Problem: ∞-category theory is a prerequisite for understanding the literature in a variety of cutting-edge areas of mathematics:

- Blumberg, Gepner, and Tabuada "A universal characterization of higher algebraic K-theory" 2013
- Gaitsgory and Rozenblyum "A study in derived algebraic geometry" 2017
- Nikolaus and Scholze "On topological cyclic homotopy" 2018
- Nadler and Tanaka "A stable ∞-category of Lagrangian cobordisms" 2020
- Bauer, Burke, and Ching "Tangent ∞-categories and Goodwillie calculus" 2023
- Fargues and Scholze "Geometrization of the local Langlands correspondence" 2024

Challenge: everything takes too long to make technically precise

Problem: ∞-category theory is a prerequisite for understanding the literature in a variety of cutting-edge areas of mathematics:

- Blumberg, Gepner, and Tabuada "A universal characterization of higher algebraic K-theory" 2013
- Gaitsgory and Rozenblyum "A study in derived algebraic geometry" 2017
- Nikolaus and Scholze "On topological cyclic homotopy" 2018
- Nadler and Tanaka "A stable ∞-category of Lagrangian cobordisms" 2020
- Bauer, Burke, and Ching "Tangent ∞-categories and Goodwillie calculus" 2023
- Fargues and Scholze "Geometrization of the local Langlands correspondence" 2024

Where will researchers in these areas find the time to learn ∞-category theory?

Challenge: everything takes too long to make technically precise

Problem: ∞-category theory is a prerequisite for understanding the literature in a variety of cutting-edge areas of mathematics:

- Blumberg, Gepner, and Tabuada "A universal characterization of higher algebraic K-theory" 2013
- Gaitsgory and Rozenblyum "A study in derived algebraic geometry" 2017
- Nikolaus and Scholze "On topological cyclic homotopy" 2018
- Nadler and Tanaka "A stable ∞-category of Lagrangian cobordisms" 2020
- Bauer, Burke, and Ching "Tangent ∞-categories and Goodwillie calculus" 2023
- Fargues and Scholze "Geometrization of the local Langlands correspondence" 2024

Where will researchers in these areas find the time to learn ∞-category theory?

How will we teach ∞-category theory to undergraduates?

Dream from the future

∞-category theory would be easier to explain if the foundations of mathematics — set theory and logic — weren't so far away.

Dream from the future

∞-category theory would be easier to explain if the foundations of mathematics — set theory and logic — weren't so far away.

Thesis statement: In a formal system more like homotopy type theory, we could teach ∞-category theory to undergraduates or to a computer.

Dream from the future

∞-category theory would be easier to explain if the foundations of mathematics — set theory and logic — weren't so far away.

Thesis statement: In a formal system more like homotopy type theory, we could teach ∞-category theory to undergraduates or to a computer.

Plan:

§1 What is category theory for?

§2 What is ∞-category theory for?

§3 An informal introduction to homotopy type theory

§4 ∞-category theory for undergraduates

§5 ∞-category theory for computers

3

An informal introduction to homotopy type theory

Homotopy type theory

Homotopy type theory is:

- a formal system for mathematical constructions and proofs

Homotopy type theory

Homotopy type theory is:

- a formal system for mathematical constructions and proofs
- in which the basic objects, types, may be regarded as "spaces" or ∞-groupoids

Homotopy type theory

Homotopy type theory is:

- a formal system for mathematical constructions and proofs
- in which the basic objects, types, may be regarded as "spaces" or ∞-groupoids
- and all constructions are automatically "continuous" or equivalence-invariant.

Homotopy type theory

Homotopy type theory is:

- a formal system for mathematical constructions and proofs
- in which the basic objects, types, may be regarded as "spaces" or ∞-groupoids
- and all constructions are automatically "continuous" or equivalence-invariant.

Homotopy type theory is $\begin{cases} \text{(homotopy type) theory} \\ \\ \end{cases}$

Homotopy type theory

Homotopy type theory is:

- a formal system for mathematical constructions and proofs
- in which the basic objects, types, may be regarded as "spaces" or ∞-groupoids
- and all constructions are automatically "continuous" or equivalence-invariant.

Homotopy type theory is $\begin{cases} \text{(homotopy type) theory} \\ \text{homotopy (type theory)} \end{cases}$

Homotopy type theory

Homotopy type theory is:

- a formal system for mathematical constructions and proofs
- in which the basic objects, types, may be regarded as "spaces" or ∞-groupoids
- and all constructions are automatically "continuous" or equivalence-invariant.

$$\text{Homotopy type theory is } \begin{cases} \text{(homotopy type) theory} \\ \text{homotopy (type theory)} \end{cases}$$

Types A are used to describe mathematical structures but are also used to encode mathematical assertions:

- In the first case, a term $a : A$ constructs a particular instance of the structure A.
- In the second case, a term $a : A$ provides a proof of the proposition A.

Homotopy type theory

Homotopy type theory is:

- a formal system for mathematical constructions and proofs
- in which the basic objects, types, may be regarded as "spaces" or ∞-groupoids
- and all constructions are automatically "continuous" or equivalence-invariant.

Homotopy type theory is $\begin{cases} \text{(homotopy type) theory} \\ \text{homotopy (type theory)} \end{cases}$

Types A are used to describe mathematical structures but are also used to encode mathematical assertions:

- In the first case, a term $a : A$ constructs a particular instance of the structure A.
- In the second case, a term $a : A$ provides a proof of the proposition A.

The slogan propositions as types means that propositions (grammatically correct mathematical assertions) are special cases of types, which in general may have non-trivial higher dimensional structure.

Types, terms, and rules

Homotopy type theory has:

- types A, B

Types, terms, and rules

Homotopy type theory has:

- types A, B; e.g., $\mathbb{N}$, $\mathbb{R}$, Group

Types, terms, and rules

Homotopy type theory has:

- types A, B; e.g., $\mathbb{N}$, $\mathbb{R}$, Group
- terms $x : A$, $y : B$

Types, terms, and rules

Homotopy type theory has:

- types A, B; e.g., $\mathbb{N}$, $\mathbb{R}$, Group
- terms $x : A$, $y : B$; e.g., $17 : \mathbb{N}$, $\sqrt{2} : \mathbb{R}$, $K_4 : \mathsf{Group}$

Types, terms, and rules

Homotopy type theory has:

- types A, B; e.g., $\mathbb{N}$, $\mathbb{R}$, Group
- terms $x : A$, $y : B$; e.g., $17 : \mathbb{N}$, $\sqrt{2} : \mathbb{R}$, $K_4 : \mathsf{Group}$
- type families $x : A \vdash B(x)$, $x : A, y : B(x) \vdash C(x, y)$

Types, terms, and rules

Homotopy type theory has:

- types A, B; e.g., $\mathbb{N}$, $\mathbb{R}$, Group
- terms $x : A$, $y : B$; e.g., $17 : \mathbb{N}$, $\sqrt{2} : \mathbb{R}$, $K_4 : \mathsf{Group}$
- type families $x : A \vdash B(x)$, $x : A, y : B(x) \vdash C(x, y)$;
  e.g., $n : \mathbb{N} \vdash n\text{-is-prime}$, $n : \mathbb{N} \vdash \mathsf{Field}_{\mathsf{char}(n)}$, $G : \mathsf{Group}, k : \mathsf{Field} \vdash \mathsf{Rep}_k(G)$

Types, terms, and rules

Homotopy type theory has:

- types A, B; e.g., $\mathbb{N}$, $\mathbb{R}$, Group
- terms $x : A$, $y : B$; e.g., $17 : \mathbb{N}$, $\sqrt{2} : \mathbb{R}$, $K_4 : \mathsf{Group}$
- type families $x : A \vdash B(x)$, $x : A, y : B(x) \vdash C(x, y)$;
  e.g., $n : \mathbb{N} \vdash n\text{-is-prime}$, $n : \mathbb{N} \vdash \mathsf{Field}_{\mathsf{char}(n)}$, $G : \mathsf{Group}, k : \mathsf{Field} \vdash \mathsf{Rep}_k(G)$
- term families $x : A \vdash b_x : B(x)$, $x : A, y : B(x) \vdash c_{x,y} : C(x, y)$

Types, terms, and rules

Homotopy type theory has:

- types A, B; e.g., $\mathbb{N}$, $\mathbb{R}$, Group
- terms $x : A$, $y : B$; e.g., $17 : \mathbb{N}$, $\sqrt{2} : \mathbb{R}$, $K_4 : \mathsf{Group}$
- type families $x : A \vdash B(x)$, $x : A, y : B(x) \vdash C(x, y)$;
  e.g., $n : \mathbb{N} \vdash n\text{-is-prime}$, $n : \mathbb{N} \vdash \mathsf{Field}_{\mathsf{char}(n)}$, $G : \mathsf{Group}, k : \mathsf{Field} \vdash \mathsf{Rep}_k(G)$
- term families $x : A \vdash b_x : B(x)$, $x : A, y : B(x) \vdash c_{x,y} : C(x, y)$;
  e.g., $n : \mathbb{N} \vdash \mathbb{Z}/n : \mathsf{Ring}_{\mathsf{char}(n)}$, $G : \mathsf{Group}, k : \mathsf{Field} \vdash k[G] : \mathsf{Rep}_k(G)$

Types, terms, and rules

Homotopy type theory has:

- types A, B; e.g., $\mathbb{N}$, $\mathbb{R}$, Group
- terms $x : A$, $y : B$; e.g., $17 : \mathbb{N}$, $\sqrt{2} : \mathbb{R}$, $K_4 : \mathsf{Group}$
- type families $x : A \vdash B(x)$, $x : A, y : B(x) \vdash C(x, y)$;
  e.g., $n : \mathbb{N} \vdash n\text{-is-prime}$, $n : \mathbb{N} \vdash \mathsf{Field}_{\mathsf{char}(n)}$, $G : \mathsf{Group}, k : \mathsf{Field} \vdash \mathsf{Rep}_k(G)$
- term families $x : A \vdash b_x : B(x)$, $x : A, y : B(x) \vdash c_{x,y} : C(x, y)$;
  e.g., $n : \mathbb{N} \vdash \mathbb{Z}/n : \mathsf{Ring}_{\mathsf{char}(n)}$, $G : \mathsf{Group}, k : \mathsf{Field} \vdash k[G] : \mathsf{Rep}_k(G)$

Formation rules build new types from given ones:

- products $A \times B$, coproducts $A + B$, function types $A \to B$,
- dependent sums $\sum_{x:A} B(x)$, dependent products $\prod_{x:A} B(x)$,
- identity types $x, y : A \vdash x =_A y$.

and come with introduction and elimination rules that construct and use their terms.

The homotopy type theoretic Rosetta stone

type theory	logic	set theory	homotopy theory
A	proposition	set	space
$x : A$	proof	element	point

The homotopy type theoretic Rosetta stone

type theory	logic	set theory	homotopy theory
A	proposition	set	space
$x : A$	proof	element	point
$\varnothing, 1$	$\bot, \top$	$\varnothing, \{\varnothing\}$	$\varnothing, *$
$A \times B$	A and B	set of pairs	product space
$A + B$	A or B	disjoint union	coproduct
$A \to B$	A implies B	set of functions	function space

The homotopy type theoretic Rosetta stone

type theory	logic	set theory	homotopy theory
A	proposition	set	space
$x : A$	proof	element	point
$\varnothing, 1$	$\bot, \top$	$\varnothing, \{\varnothing\}$	$\varnothing, *$
$A \times B$	A and B	set of pairs	product space
$A + B$	A or B	disjoint union	coproduct
$A \to B$	A implies B	set of functions	function space
$x : A \vdash B(x)$	predicate	family of sets	fibration

The homotopy type theoretic Rosetta stone

type theory	logic	set theory	homotopy theory
A	proposition	set	space
$x : A$	proof	element	point
$\varnothing, 1$	$\bot, \top$	$\varnothing, \{\varnothing\}$	$\varnothing, *$
$A \times B$	A and B	set of pairs	product space
$A + B$	A or B	disjoint union	coproduct
$A \to B$	A implies B	set of functions	function space
$x : A \vdash B(x)$	predicate	family of sets	fibration
$x : A \vdash b_x : B(x)$	conditional proof	family of elements	section

The homotopy type theoretic Rosetta stone

type theory	logic	set theory	homotopy theory
A	proposition	set	space
$x : A$	proof	element	point
$\varnothing, 1$	$\bot, \top$	$\varnothing, \{\varnothing\}$	$\varnothing, *$
$A \times B$	A and B	set of pairs	product space
$A + B$	A or B	disjoint union	coproduct
$A \to B$	A implies B	set of functions	function space
$x : A \vdash B(x)$	predicate	family of sets	fibration
$x : A \vdash b_x : B(x)$	conditional proof	family of elements	section
$\prod_{x:A} P(x)$	$\forall x.P(x)$	product	space of sections
$\sum_{x:A} P(x)$	$\exists x.P(x)$	disjoint union	total space

The homotopy type theoretic Rosetta stone

type theory	logic	set theory	homotopy theory
A	proposition	set	space
$x : A$	proof	element	point
$\varnothing, 1$	$\bot, \top$	$\varnothing, \{\varnothing\}$	$\varnothing, *$
$A \times B$	A and B	set of pairs	product space
$A + B$	A or B	disjoint union	coproduct
$A \to B$	A implies B	set of functions	function space
$x : A \vdash B(x)$	predicate	family of sets	fibration
$x : A \vdash b_x : B(x)$	conditional proof	family of elements	section
$\prod_{x:A} P(x)$	$\forall x.P(x)$	product	space of sections
$\sum_{x:A} P(x)$	$\exists x.P(x)$	disjoint union	total space
$p : x =_A y$	proof of equality	$x = y$	path from x to y
$\sum_{x,y:A} x =_A y$	equality relation	diagonal	path space for A

Identity types and path induction

Martin-Löf's identity types $x, y : A \vdash x =_A y$ are governed by the rules:

no nonsense: given a type A and terms $x, y : A$, there is a type $x =_A y$

reflexivity: given a type A and term $x : A$ there is a term $\mathsf{refl}_x : x =_A x$

indiscernibility of identicals: given a type family $x, y : A, p : x =_A y \vdash P(x, y, p)$, to prove $P(x, y, p)$ for all x, y, p, it suffices to assume y is x and p is refl_x.

The final rule defines an induction principle analogous to recursion over the natural numbers.

Identity types and path induction

Martin-Löf's identity types $x, y : A \vdash x =_A y$ are governed by the rules:

no nonsense: given a type A and terms $x, y : A$, there is a type $x =_A y$

reflexivity: given a type A and term $x : A$ there is a term $\text{refl}_x : x =_A x$

indiscernibility of identicals: given a type family $x, y : A, p : x =_A y \vdash P(x, y, p)$, to prove $P(x, y, p)$ for all x, y, p, it suffices to assume y is x and p is refl_x.

The final rule defines an induction principle analogous to recursion over the natural numbers.

Since $p : x =_A y$ are interpreted paths, we refer to this rule as path induction.

The ∞-groupoid of paths

Identity types can be iterated:

given $x, y : A$ and $p, q : x =_A y$ there is a type $p =_{x =_A y} q$.

The ∞-groupoid of paths

Identity types can be iterated:

given $x, y : A$ and $p, q : x =_A y$ there is a type $p =_{x =_A y} q$.

Theorem (Lumsdaine,Garner–van den Berg). The terms belonging to the iterated identity types of any type A form an ∞-groupoid.

The ∞-groupoid of paths

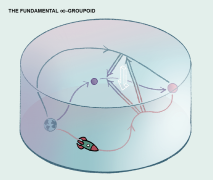
Identity types can be iterated:

given $x, y : A$ and $p, q : x =_A y$ there is a type $p =_{x=_A y} q$.

Theorem (Lumsdaine,Garner–van den Berg). The terms belonging to the iterated identity types of any type A form an ∞-groupoid.

The ∞-groupoid structure of A has

- terms $x : A$ as objects
- paths $p : x =_A y$ as 1-morphisms
- paths of paths $h : p =_{x=_A y} q$ as 2-morphisms, ...



The required structures are proven from the path induction principle:

- constant paths (reflexivity) $\mathsf{refl}_x : x = x$
- reversal (symmetry) $p : x = y$ yields $p^{-1} : y = x$
- concatenation (transitivity) $p : x = y$ and $q : y = z$ yield $p * q : x = z$

and furthermore concatenation is associative and unital, the associators are coherent ...

Contractible types and equivalences

The homotopical perspective on type theory suggests new definitions:

Contractible types and equivalences

The homotopical perspective on type theory suggests new definitions:

Definition. A type A is contractible if it comes with a term of type

$$\text{is-contr}(A) := \sum_{a:A} \prod_{x:A} a =_A x$$

Contractible types and equivalences

The homotopical perspective on type theory suggests new definitions:

Definition. A type A is contractible if it comes with a term of type

$$\text{is-contr}(A) := \sum_{a:A} \prod_{x:A} a =_A x$$

A proof of contractibility $p : \text{is-contr}(A)$ provides:

- a term $c : A$ called the center of contraction and
- a dependent function $h : \prod_{x:A} c =_A x$ called the contracting homotopy, which can be thought of as a continuous choice of paths $h(x) : c =_A x$ for each $x : A$.

Contractible types and equivalences

The homotopical perspective on type theory suggests new definitions:

Definition. A type A is contractible if it comes with a term of type

$$\text{is-contr}(A) := \sum_{a:A} \prod_{x:A} a =_A x$$

A proof of contractibility $p : \text{is-contr}(A)$ provides:

- a term $c : A$ called the center of contraction and
- a dependent function $h : \prod_{x:A} c =_A x$ called the contracting homotopy, which can be thought of as a continuous choice of paths $h(x) : c =_A x$ for each $x : A$.

Definition. An equivalence between types A and B is a term of type:

$$A \simeq B := \sum_{f:A\to B} \left(\sum_{g:B\to A} \prod_{a:A} g(f(a)) =_A a \right) \times \left(\sum_{h:B\to A} \prod_{b:B} f(h(b)) =_B b \right)$$

4

∞-category theory for undergraduates

∞-category theory for undergraduates

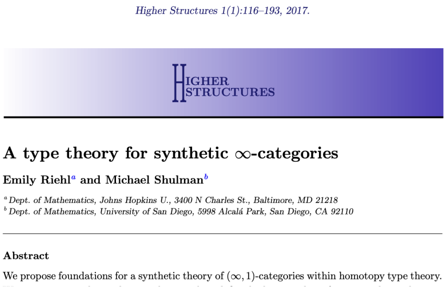
∞-category theory would be easier to explain if the foundations of mathematics — set theory and logic — weren't so far away.

Thesis statement: In a formal system more like homotopy type theory, we could teach ∞-category theory to undergraduates or to a computer.

∞-category theory for undergraduates

∞-category theory would be easier to explain if the foundations of mathematics — set theory and logic — weren't so far away.

Thesis statement: In a formal system more like homotopy type theory, we could teach ∞-category theory to undergraduates or to a computer.



Higher Structures 1(1):116–193, 2017.

HIGHER STRUCTURES

A type theory for synthetic ∞-categories

Emily Riehl[a] and Michael Shulman[b]

[a] Dept. of Mathematics, Johns Hopkins U., 3400 N Charles St., Baltimore, MD 21218
[b] Dept. of Mathematics, University of San Diego, 5998 Alcalá Park, San Diego, CA 92110

Abstract

We propose foundations for a synthetic theory of $(\infty, 1)$-categories within homotopy type theory.

Everything that follows is made rigorous in a formal framework, called simplicial homotopy type theory, which extends homotopy type theory with simplicial shapes and extension types.

Undergraduates do not need to understand the full details of this formal framework — that can be left to the experts. They just need to learn how to construct definitions and proofs.

Simplicial homotopy type theory

Simplicial homotopy type theory provides shapes defined in a syntactic language from an axiomatic bounded linear order:

$$\text{e.g., } \Delta^1 := \begin{tikzcd} \bullet \arrow[r] & \bullet \end{tikzcd}, \quad \Lambda^2_1 := \begin{tikzcd} & \bullet \arrow[dr] & \\ \bullet \arrow[ur] & & \bullet \end{tikzcd}, \quad \Delta^2 := \begin{tikzcd} & \bullet \arrow[dr] \arrow[d, phantom, "\wr\|"] & \\ \bullet \arrow[ur] \arrow[rr] & {} & \bullet \end{tikzcd}$$

Simplicial homotopy type theory

Simplicial homotopy type theory provides shapes defined in a syntactic language from an axiomatic bounded linear order:

e.g., $\Delta^1 :=$ $\begin{tikzcd} \bullet \arrow[r] & \bullet \end{tikzcd}$, $\Lambda^2_1 :=$ $\begin{tikzcd} & \bullet \arrow[dr] & \\ \bullet \arrow[ur] & & \bullet \end{tikzcd}$, $\Delta^2 :=$ $\begin{tikzcd} & \bullet \arrow[dr] \arrow[d, phantom, "\wr|"] & \\ \bullet \arrow[ur] \arrow[rr] & {} & \bullet \end{tikzcd}$

Function types are extended to allow maps from shapes into types A:

$$\text{e.g., } \Delta^1 \to A, \quad \Lambda^2_1 \to A, \quad \Delta^2 \to A$$

Simplicial homotopy type theory

Simplicial homotopy type theory provides shapes defined in a syntactic language from an axiomatic bounded linear order:

$$\text{e.g., } \Delta^1 := \bullet \longrightarrow \bullet\ , \quad \Lambda^2_1 := \begin{tikzcd}[column sep=small] & \bullet \arrow[dr] & \\ \bullet \arrow[ur] & & \bullet \end{tikzcd}\ , \quad \Delta^2 := \begin{tikzcd}[column sep=small] & \bullet \arrow[dr] \arrow[d, phantom, "\wr\|"] & \\ \bullet \arrow[ur] \arrow[rr] & {} & \bullet \end{tikzcd}$$

Function types are extended to allow maps from shapes into types A:

$$\text{e.g., } \Delta^1 \to A, \quad \Lambda^2_1 \to A, \quad \Delta^2 \to A$$

In simplicial homotopy type theory, any type A has families of:

- identity types $x, y : A \vdash x =_A y$ whose terms $p : x =_A y$ define paths
- hom types $x, y : A \vdash \mathrm{Hom}_A(x, y)$ whose terms $f : \mathrm{Hom}_A(x, y)$ define arrows

$$\begin{tikzcd} x =_A y \arrow[r, hook] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \sum_{x,y:A} x =_A y \arrow[d, "{(\mathrm{ev}_0, \mathrm{ev}_1)}"] \\ 1 \arrow[r, "{(x,y)}"] & A \times A \end{tikzcd} \qquad \begin{tikzcd} \mathrm{Hom}_A(x, y) \arrow[r, hook] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \Delta^1 \to A \arrow[d, "{(\mathrm{ev}_0, \mathrm{ev}_1)}"] \\ 1 \arrow[r, "{(x,y)}"] & A \times A \end{tikzcd}$$

Pre-∞-categories

Definition (Riehl–Shulman after Segal). A type A is a pre-∞-category if every composable pair of arrows $f : \mathrm{Hom}_A(x,y)$ and $g : \mathrm{Hom}_A(y,z)$ has a unique composite, i.e., if the fiber

$$\begin{tikzcd} \mathrm{Comp}(f,g) \arrow[r, hook] \arrow[d, "\wr"'] \arrow[dr, phantom, "\lrcorner", very near start] & \Delta^2 \to A \arrow[d, "\wr"', "\mathrm{res}"] \\ 1 \arrow[r, hook, "f \wedge g"'] & \Lambda^2_1 \to A \end{tikzcd}$$

is contractible.

Pre-∞-categories

Definition (Riehl–Shulman after Segal). A type A is a pre-∞-category if every composable pair of arrows $f : \mathrm{Hom}_A(x,y)$ and $g : \mathrm{Hom}_A(y,z)$ has a unique composite, i.e., if the fiber

$$\begin{tikzcd} \mathrm{Comp}(f,g) \arrow[r, hook] \arrow[d, "\wr"'] \arrow[dr, phantom, "\lrcorner", very near start] & \Delta^2 \to A \arrow[d, "\wr"', "\mathrm{res}"] \\ 1 \arrow[r, hook, "f \wedge g"'] & \Lambda^2_1 \to A \end{tikzcd}$$

is contractible.

By contractibility, $\mathrm{Comp}(f,g)$ has a center of contraction $\mathrm{comp}_{f,g} : \Delta^2 \to A$. Write $g \circ f : \mathrm{Hom}_A(x,z)$ for its inner face, *the* composite of f and g.

$$\begin{tikzcd} & y \arrow[dr, "g"] & \\ x \arrow[ur, "f"] \arrow[rr, "g \circ f"'] \arrow[rr, phantom, "\mathrm{comp}_{f,g}", shift left=3] & & z \end{tikzcd}$$

This defines a composition function

$$\circ : \mathrm{Hom}_A(y,z) \to \mathrm{Hom}_A(x,y) \to \mathrm{Hom}_A(x,z)$$

∞-categories

Definition (Riehl–Shulman after Segal). A type A is a pre-∞-category if every composable pair of arrows $f : \mathrm{Hom}_A(x, y)$ and $g : \mathrm{Hom}_A(y, z)$ has a unique composite.

In a pre-∞-category it follows that:

- Any $x : A$ has an identity arrow $\mathrm{id}_x : \mathrm{Hom}_A(x, x)$.
- For any $f : \mathrm{Hom}_A(x, y)$, $f \circ \mathrm{id}_x = f$ and $\mathrm{id}_y \circ f = f$.
- For any composable triple of arrows f, g, and h, $(h \circ g) \circ f = h \circ (g \circ f)$.

∞-categories

Definition (Riehl–Shulman after Segal). A type A is a pre-∞-category if every composable pair of arrows $f : \mathrm{Hom}_A(x, y)$ and $g : \mathrm{Hom}_A(y, z)$ has a unique composite.

In a pre-∞-category it follows that:

- Any $x : A$ has an identity arrow $\mathrm{id}_x : \mathrm{Hom}_A(x, x)$.
- For any $f : \mathrm{Hom}_A(x, y)$, $f \circ \mathrm{id}_x = f$ and $\mathrm{id}_y \circ f = f$.
- For any composable triple of arrows f, g, and h, $(h \circ g) \circ f = h \circ (g \circ f)$.

Note: A pre-∞-category has two ∞-groupoid cores:

one defined by the paths $p : x =_A y$ and
another defined by the isomorphisms $f : x \cong_A y$.

∞-categories

Definition (Riehl–Shulman after Segal). A type A is a pre-∞-category if every composable pair of arrows $f : \mathrm{Hom}_A(x, y)$ and $g : \mathrm{Hom}_A(y, z)$ has a unique composite.

In a pre-∞-category it follows that:

- Any $x : A$ has an identity arrow $\mathrm{id}_x : \mathrm{Hom}_A(x, x)$.
- For any $f : \mathrm{Hom}_A(x, y)$, $f \circ \mathrm{id}_x = f$ and $\mathrm{id}_y \circ f = f$.
- For any composable triple of arrows f, g, and h, $(h \circ g) \circ f = h \circ (g \circ f)$.

Note: A pre-∞-category has two ∞-groupoid cores:

one defined by the paths $p : x =_A y$ and
another defined by the isomorphisms $f : x \cong_A y$.

Definition (Riehl–Shulman after Rezk). A pre-∞-category A is ∞-category iff paths are equivalent to isomorphisms: $\prod_{x:A} \prod_{y:A} (x =_A y) \simeq (x \cong_A y)$.

Coproducts and adjunctions in ∞-categories

Definition. Two objects $a, a' : A$ in an ∞-category admit a coproduct if there is another object $a \sqcup a' : A$ and a family of equivalences

$$\prod_{x:A} \mathsf{Hom}_A(a \sqcup a', x) \simeq \mathsf{Hom}_A(a, x) \times \mathsf{Hom}_A(a', x).$$

Coproducts and adjunctions in ∞-categories

Definition. Two objects $a, a' : A$ in an ∞-category admit a coproduct if there is another object $a \sqcup a' : A$ and a family of equivalences

$$\prod_{x:A} \mathrm{Hom}_A(a \sqcup a', x) \simeq \mathrm{Hom}_A(a, x) \times \mathrm{Hom}_A(a', x).$$

Definition. A pair of functions $f\colon A \to B$ and $u\colon B \to A$ between ∞-categories define an adjunction if there is a family of equivalences

$$\prod_{a:A} \prod_{b:B} \mathrm{Hom}_B(f(a), b) \simeq \mathrm{Hom}_A(a, u(b)).$$

Coproducts and adjunctions in ∞-categories

Definition. Two objects $a, a' : A$ in an ∞-category admit a coproduct if there is another object $a \sqcup a' : A$ and a family of equivalences

$$\prod_{x:A} \mathrm{Hom}_A(a \sqcup a', x) \simeq \mathrm{Hom}_A(a, x) \times \mathrm{Hom}_A(a', x).$$

Definition. A pair of functions $f \colon A \to B$ and $u \colon B \to A$ between ∞-categories define an adjunction if there is a family of equivalences

$$\prod_{a:A} \prod_{b:B} \mathrm{Hom}_B(f(a), b) \simeq \mathrm{Hom}_A(a, u(b)).$$

Note both definitions are arguably simplier than for ordinary categories: there the equivalences must be "natural" which is automatically true here.

Left adjoints preserve coproducts

Theorem. Left adjoints $f\colon A \to B$ between between ∞-categories preserve coproducts: for $a, a' : A$ admitting a coproduct $a \sqcup a' : A$, $f(a \sqcup a') \cong f(a) \sqcup f(a')$ in B.

Left adjoints preserve coproducts

Theorem. Left adjoints $f\colon A \to B$ between between ∞-categories preserve coproducts: for $a, a' : A$ admitting a coproduct $a \sqcup a' : A$, $f(a \sqcup a') \cong f(a) \sqcup f(a')$ in B.

Proof: By the ∞-categorical Yoneda lemma, to show $f(a \sqcup a') \cong f(a) \sqcup f(a')$ it suffices to define a family of equivalences

$$\prod_{x:B} \mathrm{Hom}_B(f(a \sqcup a'), x) \simeq \mathrm{Hom}_B(f(a) \sqcup f(a'), x).$$

Left adjoints preserve coproducts

Theorem. Left adjoints $f\colon A \to B$ between between ∞-categories preserve coproducts: for $a, a' : A$ admitting a coproduct $a \sqcup a' : A$, $f(a \sqcup a') \cong f(a) \sqcup f(a')$ in B.

Proof: By the ∞-categorical Yoneda lemma, to show $f(a \sqcup a') \cong f(a) \sqcup f(a')$ it suffices to define a family of equivalences

$$\prod_{x:B} \mathrm{Hom}_B(f(a \sqcup a'), x) \simeq \mathrm{Hom}_B(f(a) \sqcup f(a'), x).$$

The equivalences defined by the coproduct and the adjunction compose as follows

$$\begin{aligned}
\mathrm{Hom}_B(f(a \sqcup a'), x) &\simeq \mathrm{Hom}_A(a \sqcup a', ux) && \text{by adjunction}\\
&\simeq \mathrm{Hom}_A(a, u(x)) \times \mathrm{Hom}_A(a', ux) && \text{by coproduct}\\
&\simeq \mathrm{Hom}_B(f(a), x) \times \mathrm{Hom}_B(f(a'), x) && \text{by adjunction}\\
&\simeq \mathrm{Hom}_B(f(a) \sqcup f(a'), x) && \text{by coproduct}
\end{aligned}$$

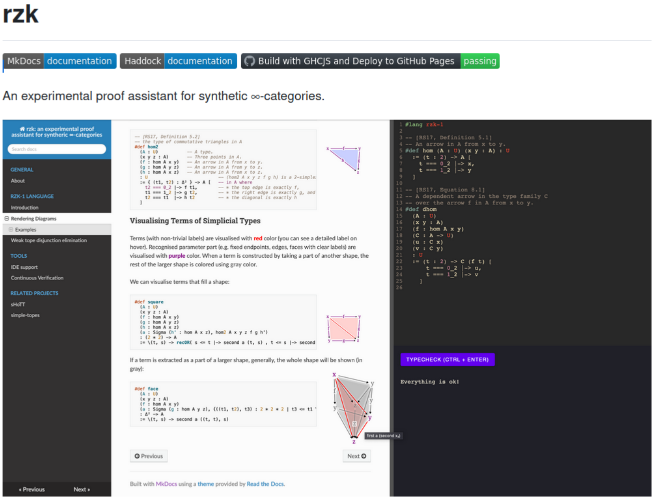
for all $x : B$. $\square$

5

∞-category theory for computers

Could ∞-category theory be taught to a computer?

These definitions, theorems, and proofs can be formally verified in a computer proof assistant that knows the rules of simplicial homotopy type theory:



The proof assistant Rzk was developed 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

Pre-∞-categories, formally

Definition (Riehl–Shulman after Segal). A type A is a pre-∞-category if every pair of arrows $f : \mathrm{Hom}_A(x, y)$ and $g : \mathrm{Hom}_A(y, z)$ has a unique composite, i.e., the type $\mathrm{Comp}(f, g)$ of composites is contractible.

A type is **a pre-∞-category** if every composable pair of arrows has a unique composite, meaning that the type of composites is contractible.

Note this is a considerable simplification of the usual Segal condition, which also requires homotopical uniqueness of higher-order composites. Here this higher-order uniqueness is a consequence of the uniqueness of binary composition.

RS17, Definition 5.3

```
#def Is-pre-∞-category
  ( A : U)
  : U
  :=
    ( x : A) → (y : A) → (z : A)
  → ( f : Hom A x y) → (g : Hom A y z)
  → is-contr (Σ (h : Hom A x z) , (Hom2 A x y z f g h))
```

Composition in a pre-∞-category, formally

By contractibility, $\mathsf{Comp}(f, g)$ has a center of contraction $\mathsf{comp}_{f,g} : \Delta^2 \to A$. Its inner face $g \circ f : \mathsf{Hom}_A(x, z)$ defines a composition function:

$$\circ : \mathsf{Hom}_A(y, z) \to \mathsf{Hom}_A(x, y) \to \mathsf{Hom}_A(x, z)$$

Pre-∞-categories have a composition functor and witnesses to the composition relation. Composition is written in diagrammatic order to match the order of arguments in `is-pre-∞-category`.

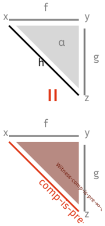
```
#def Comp-is-pre-∞-category
  ( A : U)
  ( is-pre-∞-category-A : Is-pre-∞-category A)
  ( x y z : A)
  ( f : Hom A x y)
  ( g : Hom A y z)
  : Hom A x z
  := first (first (is-pre-∞-category-A x y z f g))

#def Witness-comp-is-pre-∞-category
  ( A : U)
  ( is-pre-∞-category-A : Is-pre-∞-category A)
  ( x y z : A)
  ( f : Hom A x y)
  ( g : Hom A y z)
  : Hom2 A x y z f g (Comp-is-pre-∞-category A is-pre-∞-category-A x y z f g)
  := second (first (is-pre-∞-category-A x y z f g))
```

Uniqueness of composition in a pre-∞-category, formally

Composition in a pre-∞-category is unique in the following sense. If there is a witness that an arrow `h` is a composite of `f` and `g`, then the specified composite equals `h`.

```
#def Uniqueness-comp-is-pre-∞-category
  ( A : U)
  ( is-pre-∞-category-A : Is-pre-∞-category A)
  ( x y z : A)
  ( f : Hom A x y)
  ( g : Hom A y z)
  ( h : Hom A x z)
  ( alpha : Hom2 A x y z f g h)
  : ( Comp-is-pre-∞-category A is-pre-∞-category-A x y z f g) = h
  :=
    first-path-Σ
      ( Hom A x z)
      ( Hom2 A x y z f g)
      ( Comp-is-pre-∞-category A is-pre-∞-category-A x y z f g
      , Witness-comp-is-pre-∞-category A is-pre-∞-category-A x y z f g)
      ( h , alpha)
      ( homotopy-contraction
        ( Σ ( k : Hom A x z) , (Hom2 A x y z f g k))
        ( is-pre-∞-category-A x y z f g)
        ( h , alpha))
```



See emilyriehl.github.io/yoneda and rzk-lang.github.io/sHoTT for more.

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, 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

- Emily Riehl and Michael Shulman, A type theory for synthetic ∞-categories, Higher Structures 1(1):116–193, 2017; arXiv:1705.07442
- César Bardomiano Martínez, Limits and colimits of synthetic ∞-categories, arXiv:2202.12386
- Ulrik Buchholtz, Jonathan Weinberger, Synthetic fibered $(\infty, 1)$-category theory, Higher Structures 7(1): 74–165, 2023; arXiv:2105.01724
- Daniel Gratzer, Jonathan Weinberger, Ulrik Buchholtz, Directed univalence in simplicial homotopy type theory; arXiv:2407.09146
- Daniel Gratzer, Jonathan Weinberger, Ulrik Buchholtz, The Yoneda embedding in simplicial type theory; arXiv:2501.13229