Emily Riehl

Johns Hopkins University

# A synthetic theory of $\infty$-categories in homotopy type theory

joint with Michael Shulman

Workshop on Higher Category Approach
to Certifiably Correct Quantum Information Processing Systems

# The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

# The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects

# The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows

# The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows
- with composition associative up to invertible 3-arrows (and unital)

# The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows
- with composition associative up to invertible 3-arrows (and unital)
- with these witnesses coherent up to invertible arrows all the way up

# The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows
- with composition associative up to invertible 3-arrows (and unital)
- with these witnesses coherent up to invertible arrows all the way up

The composition operation and associativity and unit axioms in a 1-category become higher data in an $(\infty, 1)$-category.

# The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows
- with composition associative up to invertible 3-arrows (and unital)
- with these witnesses coherent up to invertible arrows all the way up

The composition operation and associativity and unit axioms in a 1-category become higher data in an $(\infty, 1)$-category.

The schematic notion of $\infty$-category is made precise by several models: quasi-categories, Rezk spaces, Segal categories, 1-complicial sets, …

# The analytic vs synthetic theory of $\infty$-categories

Q: How might you develop the category theory of $\infty$-categories?

# The analytic vs synthetic theory of $\infty$-categories

Q: How might you develop the category theory of $\infty$-categories?

Strategies:

- work analytically to give categorical definitions and prove theorems using the combinatorics of one model

(eg., Joyal, Lurie, Gepner-Haugseng, Cisinski in $\mathrm{q}\mathcal{C}\mathrm{at}$;
Kazhdan-Varshavsky, Rasekh in $\mathcal{R}\mathrm{ezk}$; Simpson in $\mathcal{S}\mathrm{egal}$)

# The analytic vs synthetic theory of $\infty$-categories

Q: How might you develop the category theory of $\infty$-categories?

Strategies:

- work analytically to give categorical definitions and prove theorems using the combinatorics of one model

  (eg., Joyal, Lurie, Gepner-Haugseng, Cisinski in $\mathrm{q}\mathcal{C}\mathrm{at}$; Kazhdan-Varshavsky, Rasekh in $\mathcal{R}\mathrm{ezk}$; Simpson in $\mathcal{S}\mathrm{egal}$)

- work synthetically to give categorical definitions and prove theorems in all four models $\mathrm{q}\mathcal{C}\mathrm{at}$, $\mathcal{R}\mathrm{ezk}$, $\mathcal{S}\mathrm{egal}$, $1\text{-}\mathcal{C}\mathrm{omp}$ at once

  (R-Verity: an $\infty$-cosmos axiomatizes the common features of the categories $\mathrm{q}\mathcal{C}\mathrm{at}$, $\mathcal{R}\mathrm{ezk}$, $\mathcal{S}\mathrm{egal}$, $1\text{-}\mathcal{C}\mathrm{omp}$ of $\infty$-categories)

# The analytic vs synthetic theory of $\infty$-categories

Q: How might you develop the category theory of $\infty$-categories?

Strategies:

- work analytically to give categorical definitions and prove theorems using the combinatorics of one model

  (eg., Joyal, Lurie, Gepner-Haugseng, Cisinski in $\mathrm{q}\mathcal{C}\mathrm{at}$; Kazhdan-Varshavsky, Rasekh in $\mathcal{R}\mathrm{ezk}$; Simpson in $\mathcal{S}\mathrm{egal}$)

- work synthetically to give categorical definitions and prove theorems in all four models $\mathrm{q}\mathcal{C}\mathrm{at}$, $\mathcal{R}\mathrm{ezk}$, $\mathcal{S}\mathrm{egal}$, $1\text{-}\mathcal{C}\mathrm{omp}$ at once

  (R-Verity: an $\infty$-cosmos axiomatizes the common features of the categories $\mathrm{q}\mathcal{C}\mathrm{at}$, $\mathcal{R}\mathrm{ezk}$, $\mathcal{S}\mathrm{egal}$, $1\text{-}\mathcal{C}\mathrm{omp}$ of $\infty$-categories)

- work synthetically in a simplicial type theory augmenting homotopy type theory to prove theorems in $\mathcal{R}\mathrm{ezk}$

  (R-Shulman: an $\infty$-category is a type with unique binary composites in which isomorphism is equivalent to identity)

# Plan

1. Homotopy type theory

2. A directed type theory for $\infty$-categories

3. The synthetic theory of $\infty$-categories

1

# Homotopy type theory

# Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$

# Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$

# Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$

# Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$ including in particular identity types $x, y : A \vdash x =_A y$.

# Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$ including in particular identity types $x, y : A \vdash x =_A y$.

Type constructors build new types and terms from given ones:

# Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$ including in particular identity types $x, y : A \vdash x =_A y$.

Type constructors build new types and terms from given ones:

- products $A \times B$, coproducts $A + B$, function types $A \to B$,

# Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$ including in particular identity types $x, y : A \vdash x =_A y$.

Type constructors build new types and terms from given ones:

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

# Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$ including in particular identity types $x, y : A \vdash x =_A y$.

Type constructors build new types and terms from given ones:

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

Each type constructor comes with rules:

(i) formation: a way to construct new types

(ii) introduction: ways to construct terms of these types

(iii) elimination: ways to use them to construct other terms

(iv) computation: what happens when we follow (ii) by (iii)

# The extended Curry-Howard correspondence

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

# The extended Curry-Howard correspondence

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

# The extended Curry-Howard correspondence

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

# The extended Curry-Howard correspondence

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

# The extended Curry-Howard correspondence

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

# The extended Curry-Howard correspondence

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

# Identity types and path induction

## Formation rule for identity types

$$\frac{x, y : A}{x =_A y \text{ type}}$$

# Identity types and path induction

**Formation and introduction rules for identity types**

$$\frac{x, y : A}{x =_A y \text{ type}} \qquad \frac{x : A}{\mathrm{refl}_x : x =_A x}$$

The elimination rule for the identity type — which says that the identity type family is freely generated by the terms $\mathrm{refl}_x : x =_A x$ — is packaged in a proof technique called path induction.

# Identity types and path induction

**Formation and introduction rules for identity types**

$$\frac{x, y : A}{x =_A y \text{ type}} \qquad \frac{x : A}{\mathrm{refl}_x : x =_A x}$$

The elimination rule for the identity type — which says that the identity type family is freely generated by the terms $\mathrm{refl}_x : x =_A x$ — is packaged in a proof technique called path induction.

**Path induction.** If $B(x, y, p)$ is a type family dependent on $x, y : A$ and $p : x =_A y$, then to prove $B(x, y, p)$ it suffices to assume $y$ is $x$ and $p$ is $\mathrm{refl}_x$.

# Identity types and path induction

## Formation and introduction rules for identity types

$$\frac{x, y : A}{x =_A y \text{ type}} \qquad \frac{x : A}{\mathrm{refl}_x : x =_A x}$$

The elimination rule for the identity type — which says that the identity type family is freely generated by the terms $\mathrm{refl}_x : x =_A x$ — is packaged in a proof technique called path induction.

**Path induction.** If $B(x, y, p)$ is a type family dependent on $x, y : A$ and $p : x =_A y$, then to prove $B(x, y, p)$ it suffices to assume $y$ is $x$ and $p$ is $\mathrm{refl}_x$. I.e., there is a function

$$\text{path-ind} : \left(\prod_{x:A} B(x, x, \mathrm{refl}_x)\right) \to \left(\prod_{x,y:A} \prod_{p:x=_A y} B(x, y, p)\right).$$

# Identity types and path induction

## Formation and introduction rules for identity types

$$\frac{x, y : A}{x =_A y \text{ type}} \qquad \frac{x : A}{\mathrm{refl}_x : x =_A x}$$

The elimination rule for the identity type — which says that the identity type family is freely generated by the terms $\mathrm{refl}_x : x =_A x$ — is packaged in a proof technique called path induction.

**Path induction.** If $B(x, y, p)$ is a type family dependent on $x, y : A$ and $p : x =_A y$, then to prove $B(x, y, p)$ it suffices to assume $y$ is $x$ and $p$ is $\mathrm{refl}_x$. I.e., there is a function

$$\text{path-ind} : \left(\prod_{x:A} B(x, x, \mathrm{refl}_x)\right) \to \left(\prod_{x,y:A} \prod_{p:x=_A y} B(x, y, p)\right).$$

# The $\infty$-groupoid of paths

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

# The $\infty$-groupoid of paths

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

The $\infty$-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 $\infty$-groupoid of paths

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

The $\infty$-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) $\mathrm{refl}_x : x = x$
- reversal (symmetry) $p : x = y$ yields $p^{-1} : y = x$
- concatenation (transitivity) $p : x = y$ and $q : y = z$ yield $q * p : x = z$

# The $\infty$-groupoid of paths

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

The $\infty$-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) $\mathrm{refl}_x : x = x$
- reversal (symmetry) $p : x = y$ yields $p^{-1} : y = x$
- concatenation (transitivity) $p : x = y$ and $q : y = z$ yield $q * p : x = z$

and furthermore

- concatenation is associative
- the associators are coherent, ...

2

# A directed type theory for $\infty$-categories

# Shapes in the theory of the directed interval

Our types may depend on other types and also on shapes $\Phi \subset 2^n$, polytopes embedded in a directed cube, defined in a language

$$\top, \bot, \wedge, \vee, \equiv \qquad \text{and} \qquad 0, 1, \leq$$

satisfying intuitionistic logic and strict interval axioms.

# Shapes in the theory of the directed interval

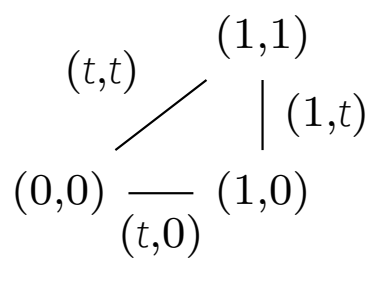
Our types may depend on other types and also on shapes $\Phi \subset 2^n$, polytopes embedded in a directed cube, defined in a language

$$\top, \bot, \wedge, \vee, \equiv \qquad \text{and} \qquad 0, 1, \leq$$

satisfying intuitionistic logic and strict interval axioms.

$$\Delta^n := \{(t_1, \ldots, t_n) : 2^n \mid t_n \leq \cdots \leq t_1\}$$

e.g. $\Delta^1 := 2$ $\quad \Delta^2 :=$




$$\partial\Delta^2 := \{(t_1, t_2) : 2^2 \mid (t_2 \leq t_1) \wedge ((0 \equiv t_2) \vee (t_2 \equiv t_1) \vee (t_1 \equiv 1))\}$$
$$\Lambda^2_1 := \{(t_1, t_2) : 2^2 \mid (t_2 \leq t_1) \wedge ((0 \equiv t_2) \vee (t_1 \equiv 1))\}$$

Because $\phi \wedge \psi$ implies $\phi$, there are shape inclusions $\Lambda^2_1 \subset \partial\Delta^2 \subset \Delta^2$.

# Extension types

## Formation rule for extension types

$$\frac{\Phi \subset \Psi \text{ shape} \qquad A \text{ type} \qquad a : \Phi \to A}{\left\langle \begin{tikzcd} \Phi \arrow[r, "a"] \arrow[d, tail] & A \\ \Psi \arrow[ur, dashed] & \end{tikzcd} \right\rangle \text{ type}}$$

# Extension types

## Formation rule for extension types

$$\frac{\Phi \subset \Psi \text{ shape} \qquad A \text{ type} \qquad a : \Phi \to A}{\left\langle \begin{tikzcd} \Phi \arrow[r, "a"] \arrow[d, tail] & A \\ \Psi \arrow[ur, dashed] & \end{tikzcd} \right\rangle \text{ type}}$$

A term $f : \left\langle \begin{tikzcd} \Phi \arrow[r, "a"] \arrow[d, tail] & A \\ \Psi \arrow[ur, dashed] & \end{tikzcd} \right\rangle$ defines

# Extension types

## Formation rule for extension types

$$\frac{\Phi \subset \Psi \text{ shape} \qquad A \text{ type} \qquad a : \Phi \to A}{\left\langle \begin{tikzcd} \Phi \arrow[r, "a"] \arrow[d, tail] & A \\ \Psi \arrow[ur, dashed] & \end{tikzcd} \right\rangle \text{ type}}$$

A term $f : \left\langle \begin{tikzcd} \Phi \arrow[r, "a"] \arrow[d, tail] & A \\ \Psi \arrow[ur, dashed] & \end{tikzcd} \right\rangle$ defines

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

# Extension types

### Formation rule for extension types

$$\frac{\Phi \subset \Psi \text{ shape} \qquad A \text{ type} \qquad a : \Phi \to A}{\left\langle \begin{tikzcd} \Phi \arrow[r, "a"] \arrow[d, tail] & A \\ \Psi \arrow[ur, dashed] & \end{tikzcd} \right\rangle \text{ type}}$$

A term $f : \left\langle \begin{tikzcd} \Phi \arrow[r, "a"] \arrow[d, tail] & A \\ \Psi \arrow[ur, dashed] & \end{tikzcd} \right\rangle$ 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.

3

# The synthetic theory of $\infty$-categories

# Hom types

The hom type for $A$ depends on two terms in $A$:

$$x, y : A \vdash \mathrm{hom}_A(x, y)$$

# Hom types

The hom type for $A$ depends on two terms in $A$:

$$x, y : A \vdash \mathsf{hom}_A(x, y)$$

$$\frac{\partial\Delta^1 \subset \Delta^1 \text{ shape} \qquad A \text{ type} \qquad [x, y] : \partial\Delta^1 \to A}{\mathsf{hom}_A(x, y) := \left\langle \begin{tikzcd} \partial\Delta^1 \arrow[r, "{[x,y]}"] \arrow[d, tail] & A \\ \Delta^1 \arrow[ur, dashed] & \end{tikzcd} \right\rangle \text{ type}}$$

# Hom types

The hom type for $A$ depends on two terms in $A$:

$$x, y : A \vdash \mathsf{hom}_A(x, y)$$

$$\frac{\partial\Delta^1 \subset \Delta^1 \text{ shape} \qquad A \text{ type} \qquad [x, y] : \partial\Delta^1 \to A}{\mathsf{hom}_A(x, y) := \left\langle \begin{tikzcd} \partial\Delta^1 \arrow[r, "{[x,y]}"] \arrow[d, tail] & A \\ \Delta^1 \arrow[ur, dashed] & \end{tikzcd} \right\rangle \text{ type}}$$

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

# Segal types $\equiv$ types with binary composition

A type $A$ is Segal iff every composable pair of arrows has a unique composite.

# Segal types $\equiv$ types with binary composition

A type $A$ is Segal iff every composable pair of arrows has a unique composite, i.e., for every $f : \mathrm{Hom}_A(x, y)$ and $g : \mathrm{Hom}_A(y, z)$ the type

$$\left\langle \begin{tikzcd} \Lambda^2_1 \arrow[r, "{[f,g]}"] \arrow[d, tail] & A \\ \Delta^2 \arrow[ur, dashed] & \end{tikzcd} \right\rangle$$

is contractible.

# Segal types $\equiv$ types with binary composition

A type $A$ is Segal iff every composable pair of arrows has a unique composite, i.e., for every $f : \mathrm{Hom}_A(x, y)$ and $g : \mathrm{Hom}_A(y, z)$ the type

$$\left\langle \begin{tikzcd} \Lambda^2_1 \arrow[r, "{[f,g]}"] \arrow[d, tail] & A \\ \Delta^2 \arrow[ur, dashed] & \end{tikzcd} \right\rangle$$

is contractible.

By contractibility, $\left\langle \begin{tikzcd} \Lambda^2_1 \arrow[r, "{[f,g]}"] \arrow[d, tail] & A \\ \Delta^2 \arrow[ur, dashed] & \end{tikzcd} \right\rangle$ has a unique inhabitant.

# Segal types $\equiv$ types with binary composition

A type $A$ is Segal iff every composable pair of arrows has a unique composite, i.e., for every $f : \mathrm{Hom}_A(x, y)$ and $g : \mathrm{Hom}_A(y, z)$ the type

$$\left\langle \begin{tikzcd} \Lambda^2_1 \arrow[r, "{[f,g]}"] \arrow[d, tail] & A \\ \Delta^2 \arrow[ur, dashed] & \end{tikzcd} \right\rangle$$

is contractible.

By contractibility, $\left\langle \begin{tikzcd} \Lambda^2_1 \arrow[r, "{[f,g]}"] \arrow[d, tail] & A \\ \Delta^2 \arrow[ur, dashed] & \end{tikzcd} \right\rangle$ has a unique inhabitant. Write $g \circ f : \mathrm{Hom}_A(x, z)$ for its inner face, *the* composite of $f$ and $g$.

# Identity arrows

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

$$\mathrm{id}_x \coloneqq \lambda t.x : \mathrm{Hom}_A(x, x) \coloneqq \left\langle \begin{tikzcd} \partial\Delta^1 \arrow[r, "{[x,x]}"] \arrow[d, tail] & A \\ \Delta^1 \arrow[ur, dashed] & \end{tikzcd} \right\rangle,$$

which we denote by $\mathrm{id}_x$ and call the identity arrow.

# Identity arrows

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

$$\mathrm{id}_x \coloneqq \lambda t.x : \mathrm{Hom}_A(x,x) \coloneqq \left\langle \begin{tikzcd} \partial\Delta^1 \arrow[r, "{[x,x]}"] \arrow[d, tail] & A \\ \Delta^1 \arrow[ur, dashed] & \end{tikzcd} \right\rangle,$$

which we denote by $\mathrm{id}_x$ and call the identity arrow.

For any $f : \mathrm{Hom}_A(x,y)$ in a Segal type $A$, the term

$$\lambda(s,t).f(t) : \left\langle \begin{tikzcd} \Lambda^2_1 \arrow[r, "{[\mathrm{id}_x, f]}"] \arrow[d, tail] & A \\ \Delta^2 \arrow[ur, dashed] & \end{tikzcd} \right\rangle$$

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

# Associativity of composition

Let $A$ be a Segal type with arrows

$$f : \mathrm{Hom}_A(x, y), \quad g : \mathrm{Hom}_A(y, z), \quad h : \mathrm{Hom}_A(z, w).$$

# Associativity of composition

Let $A$ be a Segal type with arrows

$$f : \mathrm{Hom}_A(x, y), \quad g : \mathrm{Hom}_A(y, z), \quad h : \mathrm{Hom}_A(z, w).$$

Prop.
$$h \circ (g \circ f) = (h \circ g) \circ f.$$

# Associativity of composition

Let $A$ be a Segal type with arrows

$$f : \mathrm{Hom}_A(x, y), \quad g : \mathrm{Hom}_A(y, z), \quad h : \mathrm{Hom}_A(z, w).$$

Prop. $$h \circ (g \circ f) = (h \circ g) \circ f.$$

Proof: Consider the composable arrows in the Segal type $\Delta^1 \to A$:

$$\begin{tikzcd}
 & y \arrow[dr, "g"] \arrow[dd, "g"'] \arrow[ddr, "hog"] & \\
x \arrow[ur, "f"] \arrow[dr, "gof"] \arrow[dd, "f"'] & & z \arrow[dd, "h"] \\
 & z \arrow[dr, "h"] & \\
y \arrow[ur, "g"'] & & w
\end{tikzcd}$$

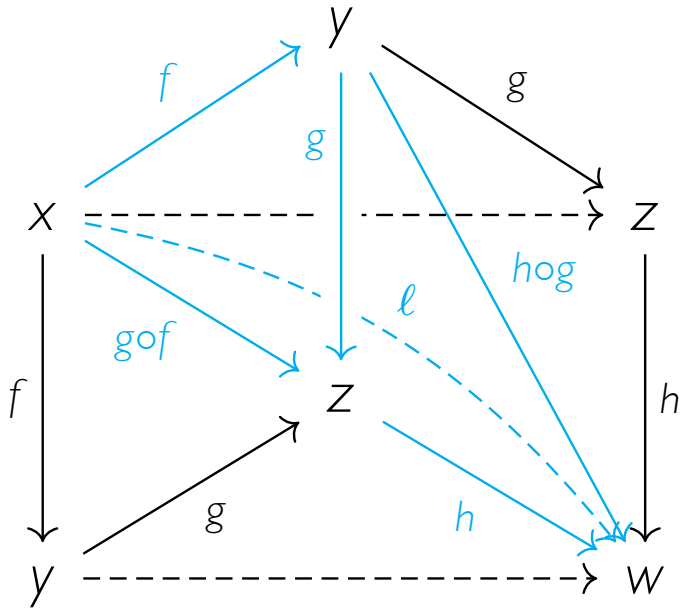
# Associativity of composition

Let $A$ be a Segal type with arrows

$$f : \mathrm{Hom}_A(x, y), \quad g : \mathrm{Hom}_A(y, z), \quad h : \mathrm{Hom}_A(z, w).$$

Prop. $$h \circ (g \circ f) = (h \circ g) \circ f.$$

Proof: Consider the composable arrows in the Segal type $\Delta^1 \to A$:




Composing defines a term in the type $\Delta^2 \to (\Delta^1 \to A)$

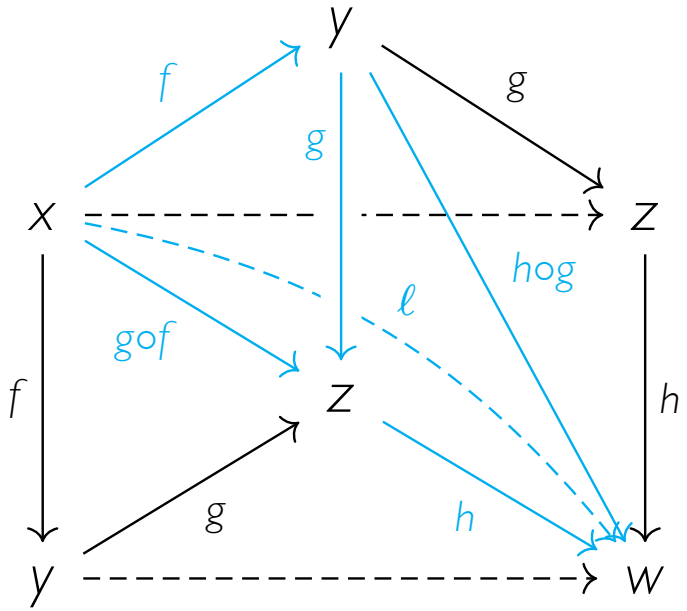
# Associativity of composition

Let $A$ be a Segal type with arrows

$$f : \mathrm{Hom}_A(x, y), \quad g : \mathrm{Hom}_A(y, z), \quad h : \mathrm{Hom}_A(z, w).$$

Prop. $$h \circ (g \circ f) = (h \circ g) \circ f.$$

Proof: Consider the composable arrows in the Segal type $\Delta^1 \to A$:




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

# Isomorphisms

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

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

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

# Isomorphisms

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

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

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

$$\mathrm{isiso}(f) :\equiv \left( \sum_{g : \mathrm{Hom}_A(y,x)} g \circ f = \mathrm{id}_x \right) \times \left( \sum_{h : \mathrm{Hom}_A(y,x)} f \circ h = \mathrm{id}_y \right).$$

# Isomorphisms

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

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

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

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

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

$$x \cong_A y := \sum_{f : \mathrm{Hom}_A(x,y)} \mathrm{isiso}(f).$$

# Rezk types $\equiv$ $\infty$-categories

By path induction, to define a map

$$\text{path-to-iso} : (x =_A y) \to (x \cong_A y)$$

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

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

# Rezk types $\equiv$ $\infty$-categories

By path induction, to define a map

$$\text{path-to-iso} : (x =_A y) \to (x \cong_A y)$$

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

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

A Segal type $A$ is Rezk iff every isomorphism is an identity

# Rezk types $\equiv$ $\infty$-categories

By path induction, to define a map

$$\text{path-to-iso} : (x =_A y) \to (x \cong_A y)$$

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

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

A Segal type $A$ is Rezk iff every isomorphism is an identity, i.e., iff the map

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

is an equivalence.

# Discrete types $\equiv \infty$-groupoids

Similarly by path induction define

$$\text{path-to-arr} \colon (x =_A y) \to \mathrm{Hom}_A(x, y)$$

for all $x, y : A$ by $\text{path-to-arr}(\mathrm{refl}_x) :\equiv \mathrm{id}_x$.

# Discrete types $\equiv \infty$-groupoids

Similarly by path induction define

$$\text{path-to-arr}\colon (x =_A y) \to \mathrm{Hom}_A(x, y)$$

for all $x, y : A$ by $\text{path-to-arr}(\mathrm{refl}_x) :\equiv \mathrm{id}_x$.

A type $A$ is discrete iff every arrow is an identity, i.e., iff path-to-arr is an equivalence.

# Discrete types $\equiv$ $\infty$-groupoids

Similarly by path induction define

$$\text{path-to-arr}\colon (x =_A y) \to \mathrm{Hom}_A(x, y)$$

for all $x, y : A$ by $\text{path-to-arr}(\mathrm{refl}_x) :\equiv \mathrm{id}_x$.

A type $A$ is discrete iff every arrow is an identity, i.e., iff path-to-arr is an equivalence.

Prop. A type is discrete if and only if it is Rezk and all of its arrows are isomorphisms.

Proof:

$$\begin{tikzcd}
x =_A y \arrow[rr, "\text{path-to-arr}"] \arrow[dr, "\text{path-to-iso}"'] & & \mathrm{Hom}_A(x, y) \\
& x \cong_A y \arrow[ur, hook] &
\end{tikzcd}$$

# $\infty$-categories for undergraduates

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

$$\text{path-to-arr}\colon (x =_A y) \to \mathrm{Hom}_A(x, y) \text{ is an equivalence.}$$

# $\infty$-categories for undergraduates

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

$$\text{path-to-arr}\colon (x =_A y) \to \mathrm{Hom}_A(x, y) \text{ is an equivalence.}$$

defn. An $\infty$-category is a type

- which has unique binary composites of arrows:

$$\left\langle \begin{tikzcd} \Lambda^2_1 \arrow[r, "{[f,g]}"] \arrow[d, tail] & A \\ \Delta^2 \arrow[ur, dashed] & \end{tikzcd} \right\rangle \quad \text{is contractible}$$

# $\infty$-categories for undergraduates

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

$$\text{path-to-arr}\colon (x =_A y) \to \mathrm{Hom}_A(x, y) \text{ is an equivalence.}$$

defn. An $\infty$-category is a type

- which has unique binary composites of arrows:

$$\left\langle \begin{tikzcd} \Lambda^2_1 \arrow[r, "{[f,g]}"] \arrow[d, tail] & A \\ \Delta^2 \arrow[ur, dashed] & \end{tikzcd} \right\rangle \quad \text{is contractible}$$

- and in which isomorphisms are equivalent to identities:

$$\text{path-to-iso}\colon (x =_A y) \to (x \cong_A y) \text{ is an equivalence.}$$

# References

For considerably more, see:

Emily Riehl and Michael Shulman

- A type theory for synthetic $\infty$-categories, Higher Structures 1(1):116–193, 2017; arXiv:1705.07442

To explore homotopy type theory:

Homotopy Type Theory: Univalent Foundations of Mathematics, `https://homotopytypetheory.org/book/`

Michael Shulman, Homotopy type theory: the logic of space, `arXiv:1703.03007`

Thank you!