Emily Riehl

Johns Hopkins University

A synthetic theory of ∞-categories in homotopy type theory

joint with Michael Shulman

CT2017 UBC, Vancouver, Canada

Motivation

Why do I study category theory?

Motivation

Why do I study category theory?

— I find category theoretic arguments to be aesthetically appealing.

Motivation

Why do I study category theory?

— I find category theoretic arguments to be aesthetically appealing.

What draws me to homotopy type theory?

Motivation

Why do I study category theory?

— I find category theoretic arguments to be aesthetically appealing.

What draws me to homotopy type theory?

— I find homotopy type theoretic arguments to be aesthetically appealing.

Plan

1. Homotopy type theory
2. A type theory for synthetic $(\infty, 1)$-categories
3. Segal types and Rezk types
4. The synthetic theory of $(\infty, 1)$-categories

Plan

1. Homotopy type theory

2. A type theory for synthetic $(\infty, 1)$-categories

3. Segal types and Rezk types

4. The synthetic theory of $(\infty, 1)$-categories

Main takeaway: the dependent Yoneda lemma is a directed analogue of path induction in HoTT.

I

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.

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

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.

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

Types A can be regarded simultaneously as both mathematical constructions and mathematical assertions, a conception also known as propositions as types;

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 can be regarded simultaneously as both mathematical constructions and mathematical assertions, a conception also known as propositions as types; accordingly, a term $a : A$ can be regarded as a proof of the proposition A.

Types, terms, and type constructors

Homotopy type theory has:

- types A, B, …

Types, terms, and type constructors

Homotopy type theory has:

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

Types, terms, and type constructors

Homotopy type theory has:

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

Types, terms, and type constructors

Homotopy type theory has:

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

Type constructors build new types and terms from given ones:

Types, terms, and type constructors

Homotopy type theory has:

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

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 $x : A \vdash B(x)$ type, $x, y : A \vdash B(x, y)$ type

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 $\prod_{x:A} B(x)$, and 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 $x : A \vdash B(x)$ type, $x, y : A \vdash B(x, y)$ type

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 $\prod_{x:A} B(x)$, and identity types $x, y : A \vdash x =_A y$.

Propositions as types:

$A \times B$	A and B	$\sum_{x:A} B(x)$	$\exists x.B(x)$
$A + B$	A or B	$\prod_{x:A} B(x)$	$\forall x.B(x)$
$A \to B$	A implies B	$x =_A y$	x equals y

Dependent sums and products

Formation rules for dependent sums and products

$$\frac{x : A \vdash B(x) \text{ type}}{\sum_{x:A} B(x) \text{ type}} \qquad \frac{x : A \vdash B(x) \text{ type}}{\prod_{x:A} B(x) \text{ type}}$$

Dependent sums and products

Formation rules for dependent sums and products

$$\frac{x : A \vdash B(x) \text{ type}}{\sum_{x:A} B(x) \text{ type}} \qquad \frac{x : A \vdash B(x) \text{ type}}{\prod_{x:A} B(x) \text{ type}}$$

Semantics

$(a, u) : \sum_{x:A} B(x)$

$$\begin{tikzcd} B(a) \arrow[r] \arrow[d, two heads] \arrow[dr, phantom, "\lrcorner"] & \sum_{x:A} B(x) \arrow[d, two heads] \\ 1 \arrow[u, dashed, bend left, "u"] \arrow[r, dashed, "a"] & A \end{tikzcd}$$

$f : \prod_{x:A} B(x)$

$$\begin{tikzcd} \sum_{x:A} B(x) \arrow[d, two heads] \\ A \arrow[u, dashed, bend right, "f"'] \end{tikzcd}$$

Dependent sums and products

Formation rules for dependent sums and products

$$\frac{x : A \vdash B(x) \text{ type}}{\sum_{x:A} B(x) \text{ type}} \qquad \frac{x : A \vdash B(x) \text{ type}}{\prod_{x:A} B(x) \text{ type}}$$

Semantics

$$(a, u) : \textstyle\sum_{x:A} B(x) \qquad f : \textstyle\prod_{x:A} B(x)$$

$$\begin{tikzcd} B(a) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_{x:A} B(x) \arrow[d] \\ 1 \arrow[u, bend left, dashed, "u"] \arrow[r, dashed, "a"] & A \end{tikzcd} \qquad \begin{tikzcd} \sum_{x:A} B(x) \arrow[d] \\ A \arrow[u, bend right, dashed, "f"'] \end{tikzcd}$$

In the case $x : A \vdash B$ type, the dependent sum becomes $A \times B$ while the dependent product becomes $A \to B$.

Dependent sums and products

Formation rules for dependent sums and products

$$\frac{x : A \vdash B(x)\ \text{type}}{\sum_{x:A} B(x)\ \text{type}} \qquad \frac{x : A \vdash B(x)\ \text{type}}{\prod_{x:A} B(x)\ \text{type}}$$

Semantics

$$(a, u) : \textstyle\sum_{x:A} B(x) \qquad\qquad f : \textstyle\prod_{x:A} B(x)$$

$$\begin{tikzcd} B(a) \arrow[r] \arrow[d, two heads] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_{x:A} B(x) \arrow[d, two heads] \\ 1 \arrow[u, dashed, bend left, "u"] \arrow[r, dashed, "a"] & A \end{tikzcd} \qquad\qquad \begin{tikzcd} \sum_{x:A} B(x) \arrow[d, two heads] \\ A \arrow[u, dashed, bend right, "f"'] \end{tikzcd}$$

In the case $x : A \vdash B$ type, the dependent sum becomes $A \times B$ while the dependent product becomes $A \to B$.

Propositions as types: If $B(x)$ is a proposition depending on $x : A$

Dependent sums and products

Formation rules for dependent sums and products

$$\frac{x : A \vdash B(x) \text{ type}}{\sum_{x:A} B(x) \text{ type}} \qquad \frac{x : A \vdash B(x) \text{ type}}{\prod_{x:A} B(x) \text{ type}}$$

Semantics

$(a, u) : \sum_{x:A} B(x)$ $\qquad$ $f : \prod_{x:A} B(x)$

$$\begin{tikzcd} B(a) \arrow[r] \arrow[d, two heads] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_{x:A} B(x) \arrow[d, two heads] \\ 1 \arrow[u, bend left, dashed, "u"] \arrow[r, dashed, "a"] & A \end{tikzcd} \qquad \begin{tikzcd} \sum_{x:A} B(x) \arrow[d, two heads] \\ A \arrow[u, bend right, dashed, "f"'] \end{tikzcd}$$

In the case $x : A \vdash B$ type, the dependent sum becomes $A \times B$ while the dependent product becomes $A \to B$.

Propositions as types: If $B(x)$ is a proposition depending on $x : A$ then (a, u) proves $\exists x.B(x)$ (constructively!)

Dependent sums and products

Formation rules for dependent sums and products

$$\frac{x : A \vdash B(x) \text{ type}}{\sum_{x:A} B(x) \text{ type}} \qquad\qquad \frac{x : A \vdash B(x) \text{ type}}{\prod_{x:A} B(x) \text{ type}}$$

Semantics

$$(a, u) : \textstyle\sum_{x:A} B(x) \qquad\qquad f : \textstyle\prod_{x:A} B(x)$$

$$\begin{tikzcd}
B(a) \arrow[r] \arrow[d, two heads] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_{x:A} B(x) \arrow[d, two heads] \\
1 \arrow[u, dashed, bend left, "u"] \arrow[r, dashed, "a"] & A
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\sum_{x:A} B(x) \arrow[d, two heads] \\
A \arrow[u, dashed, bend right, "f"']
\end{tikzcd}$$

In the case $x : A \vdash B$ type, the dependent sum becomes $A \times B$ while the dependent product becomes $A \to B$.

Propositions as types: If $B(x)$ is a proposition depending on $x : A$ then (a, u) proves $\exists x.B(x)$ (constructively!) while f proves $\forall x.B(x)$.

Identity types

Formation rule for identity types

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

Identity types

Formation and introduction rules for identity types

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

Identity types

Formation and introduction rules for identity types

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

Semantics
$$\begin{tikzcd}
 & \sum_{x,y:A} x =_A y \arrow[d, two heads] \\
A \arrow[ur, dashed, "\mathrm{refl}_x"] \arrow[r, "\Delta"'] & A \times A
\end{tikzcd}$$

Identity types

Formation and introduction rules for identity types

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

Semantics

$$\begin{tikzcd}
 & \sum_{x,y:A} x =_A y \arrow[d, two heads] \\
A \arrow[ur, dashed, "\mathrm{refl}_x"] \arrow[r, "\Delta"'] & A \times A
\end{tikzcd}$$

Indiscernability of identicals: If $B(x)$ is a type family dependent on $x : A$,

$$\phi : \prod_{x,y:A} \prod_{p:x=_A y} B(x) \to B(y).$$

Identity types

Formation and introduction rules for identity types

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

Semantics

$$\begin{tikzcd}
 & \sum_{x,y:A} x =_A y \arrow[d, two heads] \\
A \arrow[ur, dashed, "\mathsf{refl}_x"] \arrow[r, "\Delta"'] & A \times A
\end{tikzcd}$$

Indiscernability of identicals: If $B(x)$ is a type family dependent on $x : A$,

$$\phi : \prod_{x,y:A} \prod_{p:x=_A y} B(x) \to B(y).$$

Thus, if $x =_A y$ then $B(x) \to B(y)$.

Path induction

The identity type family is freely generated by the terms $\mathsf{refl}_x : x =_A x$.

Path induction

The identity type family is freely generated by the terms $\mathsf{refl}_x : x =_A x$.

Path induction: If $B(x, y, p)$ is a type family dependent on $x, y : A$ and $p : x =_A y$, then there is a function

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

Path induction

The identity type family is freely generated by the terms $\mathrm{refl}_x : x =_A x$.

Path induction: If $B(x, y, p)$ is a type family dependent on $x, y : A$ and $p : x =_A y$, then there is a function

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

Thus, to prove $B(x, y, p)$ it suffices to assume y is x and p is refl_x.

Path induction

The identity type family is freely generated by the terms $\mathsf{refl}_x : x =_A x$.

Path induction: If $B(x, y, p)$ is a type family dependent on $x, y : A$ and $p : x =_A y$, then there is a function

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

Thus, to prove $B(x, y, p)$ it suffices to assume y is x and p is refl_x.

The ∞-groupoid structure of A with

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

arises automatically from the path induction principle.

2

A type theory for synthetic $(\infty, 1)$-categories

The intended model

$$
\begin{array}{ccccccc}
\mathcal{S}\mathrm{et}^{\mathbb{\Delta}^{\mathrm{op}} \times \mathbb{\Delta}^{\mathrm{op}}} & \supset & \mathcal{R}\mathrm{eedy} & \supset & \mathcal{S}\mathrm{egal} & \supset & \mathcal{R}\mathrm{ezk} \\
\shortparallel & & \shortparallel & & \shortparallel & & \shortparallel \\
\text{bisimplicial sets} & & \text{types} & & \text{types with} & & \text{types with} \\
 & & & & \text{composition} & & \text{composition} \\
 & & & & & & \text{\& univalence}
\end{array}
$$

The intended model

$$\begin{array}{ccccccc}
\mathcal{S}\mathrm{et}^{\Delta^{\mathrm{op}} \times \Delta^{\mathrm{op}}} & \supset & \mathcal{R}\mathrm{eedy} & \supset & \mathcal{S}\mathrm{egal} & \supset & \mathcal{R}\mathrm{ezk} \\
\parallel & & \parallel & & \parallel & & \parallel \\
\text{bisimplicial sets} & & \text{types} & & \text{types with composition} & & \text{types with composition \& univalence}
\end{array}$$

Theorem (Shulman). Homotopy type theory is modeled by the category of Reedy fibrant bisimplicial sets.

The intended model

$$\begin{array}{ccccccc} \mathcal{S}\mathrm{et}^{\Delta^{op}\times\Delta^{op}} & \supset & \mathcal{R}\mathrm{eedy} & \supset & \mathcal{S}\mathrm{egal} & \supset & \mathcal{R}\mathrm{ezk} \\ \| & & \| & & \| & & \| \\ \text{bisimplicial sets} & & \text{types} & & \text{types with composition} & & \text{types with composition \& univalence} \end{array}$$

Theorem (Shulman). Homotopy type theory is modeled by the category of Reedy fibrant bisimplicial sets.

Theorem (Rezk). $(\infty, 1)$-categories are modeled by Rezk spaces aka complete Segal spaces.

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,

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

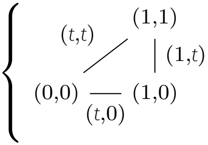
Our types may depend on other types and also on shapes $\Phi \subset 2^n$, polytopes embedded in a directed cube, defined in a language

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

satisfying intuitionistic logic and strict interval axioms.

$$\Delta^n := \{(t_1, \ldots, t_n) : 2^n \mid t_n \leq \cdots \leq t_1\} \qquad \text{e.g.} \qquad \Delta^1 := 2$$

$\Delta^2 :=$



Shapes in the theory of the directed interval

Our types may depend on other types and also on shapes $\Phi \subset \mathbb{2}^n$, 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.

$$\Delta^n := \{(t_1, \ldots, t_n) : \mathbb{2}^n \mid t_n \leq \cdots \leq t_1\} \qquad \text{e.g.} \qquad \Delta^1 := \mathbb{2}$$

$$\Delta^2 := \left\{ \begin{tikzcd} & (1,1) \\ (0,0) \arrow[ur, dash, "{(t,t)}"] \arrow[r, dash, "{(t,0)}"'] & (1,0) \arrow[u, dash, "{(1,t)}"'] \end{tikzcd} \right.$$

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

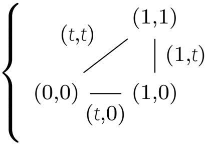
Shapes in the theory of the directed interval

Our types may depend on other types and also on shapes $\Phi \subset \mathbb{2}^n$, 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.

$$\Delta^n := \{(t_1, \ldots, t_n) : \mathbb{2}^n \mid t_n \leq \cdots \leq t_1\} \qquad \text{e.g.} \qquad \Delta^1 := \mathbb{2}$$

$\Delta^2 :=$ 

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

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

Shapes in the theory of the directed interval

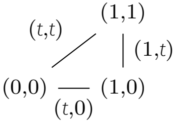
Our types may depend on other types and also on shapes $\Phi \subset \mathbb{2}^n$, polytopes embedded in a directed cube, defined in a language

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

satisfying intuitionistic logic and strict interval axioms.

$$\Delta^n := \{(t_1, \ldots, t_n) : \mathbb{2}^n \mid t_n \leq \cdots \leq t_1\} \qquad \text{e.g.} \qquad \Delta^1 := \mathbb{2}$$

$$\Delta^2 := \Bigg\{$$



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

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

Extension types

shape inclusion: $\Phi := \{t \in 2^n \mid \phi\}$ and $\Psi = \{t \in 2^n \mid \psi\}$ so that ϕ implies ψ, i.e., so that $\Phi \subset \Psi$.

Extension types

shape inclusion: $\Phi := \{t \in 2^n \mid \phi\}$ and $\Psi = \{t \in 2^n \mid \psi\}$ so that ϕ implies ψ, i.e., so that $\Phi \subset \Psi$.

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

shape inclusion: $\Phi := \{t \in 2^n \mid \phi\}$ and $\Psi = \{t \in 2^n \mid \psi\}$ so that ϕ implies ψ, i.e., so that $\Phi \subset \Psi$.

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

shape inclusion: $\Phi := \{t \in 2^n \mid \phi\}$ and $\Psi = \{t \in 2^n \mid \psi\}$ so that ϕ implies ψ, i.e., so that $\Phi \subset \Psi$.

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

Segal types and Rezk types

Hom types

Formation rule for extension types

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

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

$$x, y : A \vdash \hom_A(x, y)$$

Hom types

Formation rule for extension types

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

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

$$x, y : A \vdash \hom_A(x, y)$$

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

Hom types

Formation rule for extension types

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

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

$$x, y : A \vdash \hom_A(x, y)$$

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

A term $f : \hom_A(x, y)$ defines an arrow from x to y.

Segal types have unique binary composites

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

Segal types have unique binary composites

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

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

is contractible.

Segal types have unique binary composites

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

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

Prop. A Reedy fibrant bisimplicial set A is *Segal* if and only if $A^{\Delta^2} \twoheadrightarrow A^{\Lambda^2_1}$ is a Reedy trivial fibration.

Segal types have unique binary composites

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

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

Prop. A Reedy fibrant bisimplicial set A is *Segal* if and only if $A^{\Delta^2} \twoheadrightarrow A^{\Lambda^2_1}$ is a Reedy trivial fibration.

Notation. Let $\mathrm{comp}_{g,f} : \left\langle \begin{tikzcd} \Lambda^2_1 \arrow[r, "{[f,g]}"] \arrow[d, tail] & A \\ \Delta^2 \arrow[ur, dashed] & \end{tikzcd} \right\rangle$ denote the unique inhabitant

Segal types have unique binary composites

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

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

Prop. A Reedy fibrant bisimplicial set A is *Segal* if and only if $A^{\Delta^2} \twoheadrightarrow A^{\Lambda^2_1}$ is a Reedy trivial fibration.

Notation. Let $\mathrm{comp}_{g,f} : \left\langle \begin{tikzcd} \Lambda^2_1 \arrow[r, "{[f,g]}"] \arrow[d, tail] & A \\ \Delta^2 \arrow[ur, dashed] & \end{tikzcd} \right\rangle$ denote the unique inhabitant and write $g \circ f : \hom_A(x, z)$ for its inner face, *the* composite of f and g.

Identity arrows

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

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

which we denote by id_x and call the identity arrow.

Identity arrows

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

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

which we denote by id_x and call the identity arrow.

For any $f : \hom_A(x,y)$ in a Segal type A, the term

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

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

Associativity of composition

Let A be a Segal type with arrows

$$f : \hom_A(x, y), \quad g : \hom_A(y, z), \quad h : \hom_A(z, w).$$

Associativity of composition

Let A be a Segal type with arrows

$$f : \hom_A(x, y), \quad g : \hom_A(y, z), \quad h : \hom_A(z, w).$$

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

Associativity of composition

Let A be a Segal type with arrows

$$f : \hom_A(x, y), \quad g : \hom_A(y, z), \quad h : \hom_A(z, w).$$

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

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

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

Associativity of composition

Let A be a Segal type with arrows

$$f : \hom_A(x, y), \quad g : \hom_A(y, z), \quad h : \hom_A(z, w).$$

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

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

$$\begin{tikzcd}
 & y \arrow[rd, "g"] \arrow[dd, "g"'] \arrow[dddr, "h \circ g"] & \\
x \arrow[ru, "f"] \arrow[rr, dashed] \arrow[rd, "g \circ f"'] \arrow[ddrr, dashed, "\ell"] \arrow[dd, "f"'] & & z \arrow[dd, "h"] \\
 & z \arrow[rd, "h"] & \\
y \arrow[ru, "g"] \arrow[rr, dashed] & & w
\end{tikzcd}$$

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

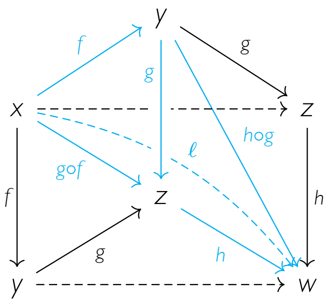
Associativity of composition

Let A be a Segal type with arrows

$$f : \hom_A(x,y), \quad g : \hom_A(y,z), \quad h : \hom_A(z,w).$$

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 $\ell : \hom_A(x,w)$ so that $\ell = h \circ (g \circ f)$ and $\ell = (h \circ g) \circ f$.

Isomorphisms

An arrow $f\colon \hom_A(x,y)$ in a Segal type is an isomorphism if it has a two-sided inverse $g\colon \hom_A(y,x)$. However, the type

$$\sum_{g\colon \hom_A(y,x)} (g \circ f = \mathrm{id}_x) \times (f \circ g = \mathrm{id}_y)$$

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

Isomorphisms

An arrow $f \colon \hom_A(x,y)$ in a Segal type is an isomorphism if it has a two-sided inverse $g \colon \hom_A(y,x)$. However, the type

$$\sum_{g \colon \hom_A(y,x)} (g \circ f = \mathrm{id}_x) \times (f \circ g = \mathrm{id}_y)$$

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

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

Isomorphisms

An arrow $f\colon \hom_A(x,y)$ in a Segal type is an isomorphism if it has a two-sided inverse $g\colon \hom_A(y,x)$. However, the type

$$\sum_{g\colon \hom_A(y,x)} (g \circ f = \mathrm{id}_x) \times (f \circ g = \mathrm{id}_y)$$

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

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

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

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

Rezk types

By path induction, to define a map

$$\mathsf{id\text{-}to\text{-}iso} \colon (x =_A y) \to (x \cong_A y)$$

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

$$\mathsf{id\text{-}to\text{-}iso}(\mathsf{refl}_x) :\equiv \mathsf{id}_x.$$

Rezk types

By path induction, to define a map

$$\text{id-to-iso}\colon (x =_A y) \to (x \cong_A y)$$

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

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

A Segal type A is Rezk if every isomorphism is an identity

Rezk types

By path induction, to define a map

$$\text{id-to-iso}\colon (x =_A y) \to (x \cong_A y)$$

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

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

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

$$\text{id-to-iso}\colon (x =_A y) \to (x \cong_A y)$$

is an equivalence.

Discrete types

Similarly by path induction define

$$\text{id-to-arr}\colon \prod_{x,y:A} (x =_A y) \to \hom_A(x,y) \qquad \text{by} \qquad \text{id-to-arr}(\mathsf{refl}_x) :\equiv \mathrm{id}_x.$$

Discrete types

Similarly by path induction define

$$\text{id-to-arr}\colon \prod_{x,y:A}(x =_A y) \to \hom_A(x,y) \qquad \text{by} \qquad \text{id-to-arr}(\mathrm{refl}_x) :\equiv \mathrm{id}_x.$$

A type A is discrete if id-to-arr is an equivalence for all $x, y : A$.

Discrete types

Similarly by path induction define

$$\text{id-to-arr} \colon \prod_{x,y:A} (x =_A y) \to \hom_A(x,y) \qquad \text{by} \qquad \text{id-to-arr}(\text{refl}_x) :\equiv \text{id}_x.$$

A type A is discrete if id-to-arr is an equivalence for all $x, y : A$.

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

Discrete types

Similarly by path induction define

$$\text{id-to-arr}\colon \prod_{x,y:A} (x =_A y) \to \hom_A(x,y) \qquad \text{by} \qquad \text{id-to-arr}(\text{refl}_x) :\equiv \text{id}_x.$$

A type A is discrete if id-to-arr is an equivalence for all $x, y : A$.

Prop. A type is discrete if and only if it is Rezk and all of its arrows are isomorphisms. If the Rezk types are $(\infty, 1)$-categories, then the discrete types are ∞-groupoids.

Discrete types

Similarly by path induction define

$$\text{id-to-arr}\colon \prod_{x,y:A} (x =_A y) \to \hom_A(x,y) \qquad \text{by} \qquad \text{id-to-arr}(\text{refl}_x) :\equiv \text{id}_x.$$

A type A is discrete if id-to-arr is an equivalence for all $x, y : A$.

Prop. A type is discrete if and only if it is Rezk and all of its arrows are isomorphisms. If the Rezk types are $(\infty, 1)$-categories, then the discrete types are ∞-groupoids.

Proof:

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

4

The synthetic theory of $(\infty, 1)$-categories

Covariant fibrations I

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x, y)$ and $u : B(x)$ there is a unique lift of f with domain u.

Covariant fibrations I

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x, y)$ and $u : B(x)$ there is a unique lift of f with domain u, i.e., if

$$\sum_{v:B(y)} \hom_{B(f)}(u, v) \quad \text{is contractible.}$$

Covariant fibrations I

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x, y)$ and $u : B(x)$ there is a unique lift of f with domain u, i.e., if

$$\sum_{v:B(y)} \hom_{B(f)}(u, v) \quad \text{is contractible.}$$

Recall

$$\hom_A(x, y) := \left\langle \begin{tikzcd} \partial\Delta^1 \arrow[r, "{[x,y]}"] \arrow[d, tail] & A \\ \Delta^1 \arrow[ur, dashed] & \end{tikzcd} \right\rangle$$

is the type of arrows in A from x to y.

Covariant fibrations I

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x,y)$ and $u : B(x)$ there is a unique lift of f with domain u, i.e., if

$$\sum_{v:B(y)} \hom_{B(f)}(u,v) \quad \text{is contractible.}$$

Here

$$\hom_{B(f)}(u,v) :\equiv \left\langle \begin{tikzcd} & B(f) \arrow[d, two heads] \\ \partial\Delta^1 \arrow[ur, "{[u,v]}"] \arrow[r, tail] & \Delta^1 \arrow[u, dashed, bend right] \end{tikzcd} \right\rangle \quad \text{where} \quad \begin{tikzcd} B(f) \arrow[r] \arrow[d, two heads] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, two heads] \\ \Delta^1 \arrow[r, "f"'] & A \end{tikzcd}$$

is the type of arrows in B from u to v over f.

Covariant fibrations I

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x,y)$ and $u : B(x)$ there is a unique lift of f with domain u, i.e., if

$$\sum_{v:B(y)} \hom_{B(f)}(u,v) \quad \text{is contractible.}$$

Here

$$\hom_{B(f)}(u,v) :\equiv \left\langle \begin{tikzcd} & B(f) \arrow[d, two heads] \\ \partial\Delta^1 \arrow[ur, "{[u,v]}"] \arrow[r, tail] & \Delta^1 \arrow[u, dashed, bend right] \end{tikzcd} \right\rangle \quad \text{where} \quad \begin{tikzcd} B(f) \arrow[r] \arrow[d, two heads] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, two heads] \\ \Delta^1 \arrow[r, "f"'] & A \end{tikzcd}$$

is the type of arrows in B from u to v over f.

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

Covariant fibrations I

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x,y)$ and $u : B(x)$ there is a unique lift of f with domain u, i.e., if

$$\sum_{v:B(y)} \hom_{B(f)}(u,v) \quad \text{is contractible.}$$

Here

$$\hom_{B(f)}(u,v) :\equiv \left\langle \begin{tikzcd} & B(f) \arrow[d, two heads] \\ \partial\Delta^1 \arrow[ur, "{[u,v]}"] \arrow[r, tail] & \Delta^1 \arrow[u, dashed, bend right] \end{tikzcd} \right\rangle \quad \text{where} \quad \begin{tikzcd} B(f) \arrow[r] \arrow[d, two heads] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, two heads] \\ \Delta^1 \arrow[r, "f"'] & A \end{tikzcd}$$

is the type of arrows in B from u to v over f.

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

Prop. For $u : B(x)$, $f : \hom_A(x,y)$, and $g : \hom_A(y,z)$,

$$g_*(f_*u) = (g \circ f)_*u \qquad \text{and} \qquad (\mathrm{id}_x)_*u = u.$$

Covariant fibrations II

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x, y)$ and $u : B(x)$ there is a unique lift of f with domain u, i.e., if

$$\sum_{v:B(y)} \hom_{B(f)}(u, v) \quad \text{is contractible.}$$

Covariant fibrations II

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x, y)$ and $u : B(x)$ there is a unique lift of f with domain u, i.e., if

$$\sum_{v:B(y)} \hom_{B(f)}(u, v) \quad \text{is contractible.}$$

Prop. If $x : A \vdash B(x)$ is covariant then for each $x : A$ the fiber $B(x)$ is discrete.

Covariant fibrations II

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x, y)$ and $u : B(x)$ there is a unique lift of f with domain u, i.e., if

$$\sum_{v:B(y)} \hom_{B(f)}(u, v) \quad \text{is contractible.}$$

Prop. If $x : A \vdash B(x)$ is covariant then for each $x : A$ the fiber $B(x)$ is discrete.

Prop. Fix $a : A$. The type family $x : A \vdash \hom_A(a, x)$ is covariant.

Covariant fibrations II

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x, y)$ and $u : B(x)$ there is a unique lift of f with domain u, i.e., if

$$\sum_{v : B(y)} \hom_{B(f)}(u, v) \quad \text{is contractible.}$$

Prop. If $x : A \vdash B(x)$ is covariant then for each $x : A$ the fiber $B(x)$ is discrete.

Prop. Fix $a : A$. The type family $x : A \vdash \hom_A(a, x)$ is covariant.

For $u : \hom_A(a, x)$ and $f : \hom_A(x, y)$, the transport $f_* u$ equals the composite $f \circ u$ as terms in $\hom_A(a, y)$.

Covariant fibrations II

A type family $x : A \vdash B(x)$ over a Segal type A is covariant if for every $f : \hom_A(x, y)$ and $u : B(x)$ there is a unique lift of f with domain u, i.e., if

$$\sum_{v:B(y)} \hom_{B(f)}(u, v) \quad \text{is contractible.}$$

Prop. If $x : A \vdash B(x)$ is covariant then for each $x : A$ the fiber $B(x)$ is discrete.

Prop. Fix $a : A$. The type family $x : A \vdash \hom_A(a, x)$ is covariant.

For $u : \hom_A(a, x)$ and $f : \hom_A(x, y)$, the transport $f_* u$ equals the composite $f \circ u$ as terms in $\hom_A(a, y)$, i.e., $f_*(u) = f \circ u$.

The Yoneda lemma

Let $x : A \vdash B(x)$ be a covariant family over a Segal type and fix $a : A$.

The Yoneda lemma

Let $x : A \vdash B(x)$ be a covariant family over a Segal type and fix $a : A$.

Yoneda lemma. The maps

$$\text{ev-id} :\equiv \lambda\phi.\phi(a, \mathrm{id}_a) : \left(\prod_{x:A} \hom_A(a,x) \to B(x)\right) \to B(a)$$

and

$$\text{yon} :\equiv \lambda u.\lambda x.\lambda f.f_* u : B(a) \to \left(\prod_{x:A} \hom_A(a,x) \to B(x)\right)$$

are inverse equivalences.

The Yoneda lemma

Let $x : A \vdash B(x)$ be a covariant family over a Segal type and fix $a : A$.

Yoneda lemma. The maps

$$\text{ev-id} := \lambda\phi.\phi(a, \text{id}_a) : \left(\prod_{x:A} \hom_A(a, x) \to B(x)\right) \to B(a)$$

and

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

are inverse equivalences.

Proof: The transport operation for covariant families is functorial in A and fiberwise maps between covariant families are automatically natural.

The Yoneda lemma

Let $x : A \vdash B(x)$ be a covariant family over a Segal type and fix $a : A$.

Yoneda lemma. The maps

$$\text{ev-id} := \lambda\phi.\phi(a, \mathrm{id}_a) : \left(\prod_{x:A} \hom_A(a,x) \to B(x)\right) \to B(a)$$

and

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

are inverse equivalences.

Proof: The transport operation for covariant families is functorial in A and fiberwise maps between covariant families are automatically natural.

Note. A representable *isomorphism* $\phi : \prod_{x:A} \hom_A(a,x) \cong \hom_A(b,x)$ induces an *identity* $\text{ev-id}(\phi) : b =_A a$ if the Segal type A is Rezk.

The dependent Yoneda lemma

From a type-theoretic perspective, the Yoneda lemma is a "directed" version of the "transport" operation for identity types. This suggests a "dependently typed" generalization of the Yoneda lemma, analogous to the full induction principle for identity types.

The dependent Yoneda lemma

From a type-theoretic perspective, the Yoneda lemma is a "directed" version of the "transport" operation for identity types. This suggests a "dependently typed" generalization of the Yoneda lemma, analogous to the full induction principle for identity types.

Dependent Yoneda lemma. If A is a Segal type and $B(x, y, f)$ is a covariant family dependent on $x, y : A$ and $f : \hom_A(x, y)$, then evaluation at (x, x, id_x) defines an equivalence

$$\text{ev-id} : \left(\prod_{x,y:A} \prod_{f:\hom_A(x,y)} B(x, y, f) \right) \to \prod_{x:A} B(x, x, \mathrm{id}_x)$$

The dependent Yoneda lemma

From a type-theoretic perspective, the Yoneda lemma is a "directed" version of the "transport" operation for identity types. This suggests a "dependently typed" generalization of the Yoneda lemma, analogous to the full induction principle for identity types.

Dependent Yoneda lemma. If A is a Segal type and $B(x, y, f)$ is a covariant family dependent on $x, y : A$ and $f : \hom_A(x, y)$, then evaluation at (x, x, id_x) defines an equivalence

$$\text{ev-id} : \left(\prod_{x,y:A} \prod_{f:\hom_A(x,y)} B(x, y, f) \right) \to \prod_{x:A} B(x, x, \mathrm{id}_x)$$

This is useful for proving equivalences between various types of coherent or incoherent adjunction data.

Dependent Yoneda is directed path induction

Takeaway: the dependent Yoneda lemma is directed path induction.

Dependent Yoneda is directed path induction

Takeaway: the dependent Yoneda lemma is directed path induction.

Path induction: If $B(x, y, p)$ is a type family dependent on $x, y : A$ and $p : x =_A y$, then there is a function

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

Thus, to prove $B(x, y, p)$ it suffices to assume y is x and p is refl_x.

Dependent Yoneda is directed path induction

Takeaway: the dependent Yoneda lemma is directed path induction.

Path induction: If $B(x, y, p)$ is a type family dependent on $x, y : A$ and $p : x =_A y$, then there is a function

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

Thus, to prove $B(x, y, p)$ it suffices to assume y is x and p is refl_x.

Dependent Yoneda Lemma: If $B(x, y, f)$ is a covariant family dependent on $x, y : A$ and $f : \hom_A(x, y)$ and A is Segal, then there is a function

$$\text{id-ind} : \left(\prod_{x:A} B(x, x, \mathrm{id}_x) \right) \to \left(\prod_{x,y:A} \prod_{f:\hom_A(x,y)} B(x, y, f) \right).$$

Thus, to prove $B(x, y, p)$ it suffices to assume y is x and f is id_x.

References

For considerably more, see:

Emily Riehl and Michael Shulman, A type theory for synthetic ∞-categories, `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!