A type theory for synthetic ∞-categories

after a homonymous paper by E. Riehl and M. Shulman

Aras Ergus

July 1, 2020

Outline

Why should homotopy theorists care about homotopy type theory?

HoTT provides a "synthetic" framework in which one can only talk about "homotopically correct" statements and constructions.

For example, in this world,

- all maps one can write down are continuous (if between spaces) / functorial (if between categories),
- two functions are "equal" if they are homotopic,
- in particular, one cannot distinguish between (homotopy) equivalent objects,
- "unique" means that the space of choices is contractible.

Why should homotopy theorists care about homotopy type theory?

HoTT provides a "synthetic" framework in which one can only talk about "homotopically correct" statements and constructions.

For example, in this world,

- all maps one can write down are continuous (if between spaces) / functorial (if between categories),
- two functions are "equal" if they are homotopic,
- in particular, one cannot distinguish between (homotopy) equivalent objects,
- "unique" means that the space of choices is contractible.

Why should homotopy theorists care about homotopy type theory?

HoTT provides a "synthetic" framework in which one can only talk about "homotopically correct" statements and constructions.

For example, in this world,

- all maps one can write down are continuous (if between spaces) / functorial (if between categories),
- two functions are "equal" if they are homotopic,
- in particular, one cannot distinguish between (homotopy) equivalent objects,
- "unique" means that the space of choices is contractible.

Why should homotopy theorists care about homotopy type theory?

HoTT provides a “synthetic” framework in which one can only talk about “homotopically correct” statements and constructions.

For example, in this world,

- all maps one can write down are continuous (if between spaces) / functorial (if between categories),
- two functions are “equal” if they are homotopic,
- in particular, one cannot distinguish between (homotopy) equivalent objects,
- “unique” means that the space of choices is contractible.

Why should homotopy theorists care about homotopy type theory?

HoTT provides a "synthetic" framework in which one can only talk about "homotopically correct" statements and constructions.

For example, in this world,

- all maps one can write down are continuous (if between spaces) / functorial (if between categories),
- two functions are "equal" if they are homotopic,
- in particular, one cannot distinguish between (homotopy) equivalent objects,
- "unique" means that the space of choices is contractible.

Why should homotopy theorists care about homotopy type theory?

HoTT provides a “synthetic” framework in which one can only talk about “homotopically correct” statements and constructions.

For example, in this world,

- all maps one can write down are continuous (if between spaces) / functorial (if between categories),
- two functions are “equal” if they are homotopic,
- in particular, one cannot distinguish between (homotopy) equivalent objects,
- “unique” means that the space of choices is contractible.

What does this buy us?

This alternative language is arguably simpler and more intuitive.

For example, one can prove the following version of the Yoneda lemma which (with a little bit of sloppy notation) looks a lot like the classical one:

Theorem (Theorem 9.1)

Let $\mathcal{A}$ be an ∞-category, $\mathcal{C} \to \mathcal{A}$ a covariant family and $a \in \mathcal{A}$. Then the following is an equivalence:

$$\begin{aligned} \mathcal{C}_a &\longleftrightarrow \mathrm{Nat}(\mathrm{Hom}(a,-),\mathcal{C}_{(-)}) \\ u &\longmapsto (f \mapsto \mathcal{C}_f(u)) \\ \varphi(\mathrm{Id}_a) &\longleftarrow\!\shortmid \varphi \end{aligned}$$

What does this buy us?

This alternative language is arguably simpler and more intuitive.

For example, one can prove the following version of the Yoneda lemma which (with a little bit of sloppy notation) looks a lot like the classical one:

Theorem (Theorem 9.1)

Let $\mathcal{A}$ be an ∞-category, $\mathcal{C} \to \mathcal{A}$ a covariant family and $a \in \mathcal{A}$. Then the following is an equivalence:

$$\begin{aligned} \mathcal{C}_a &\longleftrightarrow \mathrm{Nat}(\mathrm{Hom}(a,-), \mathcal{C}_{(-)}) \\ u &\longmapsto (f \mapsto \mathcal{C}_f(u)) \\ \varphi(\mathrm{Id}_a) &\longleftarrow\!\shortmid\; \varphi \end{aligned}$$

What does this buy us?

This alternative language is arguably simpler and more intuitive.

For example, one can prove the following version of the Yoneda lemma which (with a little bit of sloppy notation) looks a lot like the classical one:

Theorem (Theorem 9.1)

Let $\mathcal{A}$ be an ∞-category, $\mathcal{C} \to \mathcal{A}$ a covariant family and $a \in \mathcal{A}$. Then the following is an equivalence:

$$\begin{aligned} \mathcal{C}_a &\longleftrightarrow \mathrm{Nat}(\mathrm{Hom}(a,-), \mathcal{C}_{(-)}) \\ u &\longmapsto (f \mapsto \mathcal{C}_f(u)) \\ \varphi(\mathrm{Id}_a) &\longleftarrow\!\shortmid \varphi \end{aligned}$$

What's the catch?

- In any given such framework, that are things we cannot express in it which are nevertheless interesting in concrete models.
- One has to know how to "interpet the synthetic language in the real world". Making this translation work is rather tedious.

What's the catch?

- In any given such framework, that are things we cannot express in it which are nevertheless interesting in concrete models.
- One has to know how to "interpet the synthetic language in the real world". Making this translation work is rather tedious.

The model we will mimic

While trying to axiomatize the theory of ∞-categories, we will draw inspiration from (complete) Segal spaces.

There are several reasons for this choice.

A "type-theoretic" reason

Reedy fibrant simplicial spaces is a model of "ordinary" homotopy type theory, so we can try to augment this model to be able to talk about (complete) Segal spaces.

N.B. We cannot extend the classical simplicial model of HoTT to quasicategories because types in this model corresponds to Kan complexes, so not every ∞-category would have an "underlying type".

A "type-theoretic" reason

Reedy fibrant simplicial spaces is a model of "ordinary" homotopy type theory, so we can try to augment this model to be able to talk about (complete) Segal spaces.

N.B. We cannot extend the classical simplicial model of HoTT to quasicategories because types in this model corresponds to Kan complexes, so not every ∞-category would have an "underlying type".

First "homotopy-theoretic" reason

The Segal condition for Reedy fibrant simplicial spaces can be expressed in terms of a single equivalence of simplicial spaces:

Definition

The *horizontal embedding* of a simplicial set $\mathbb{X}$ is the simplicial space X is given by $X_{k,l} := \mathbb{X}_k$ for all $k, l \in \mathbb{N}$.
For $n \in \mathbb{N}$ and $i \in \{0, \ldots, n\}$, let $F(n)$ be the horizontal embedding of Δ^n and $L(n,i)$ the horizontal embedding of Λ^n_i.

Theorem (Theorem A.21)

A Reedy fibrant simplicial space X is a Segal space if and only if

$$\mathrm{Map}(F(2), X) \to \mathrm{Map}(L(2,1), X)$$

is a trivial Reedy fibration.

First "homotopy-theoretic" reason

The Segal condition for Reedy fibrant simplicial spaces can be expressed in terms of a single equivalence of simplicial spaces:

Definition

The *horizontal embedding* of a simplicial set $\mathbb{X}$ is the simplicial space X is given by $X_{k,l} := \mathbb{X}_k$ for all $k, l \in \mathbb{N}$.
For $n \in \mathbb{N}$ and $i \in \{0, \ldots, n\}$, let $F(n)$ be the horizontal embedding of Δ^n and $L(n, i)$ the horizontal embedding of Λ^n_i.

Theorem (Theorem A.21)

A Reedy fibrant simplicial space X is a Segal space if and only if

$$\mathrm{Map}(F(2), X) \to \mathrm{Map}(L(2, 1), X)$$

is a trivial Reedy fibration.

First "homotopy-theoretic" reason

The Segal condition for Reedy fibrant simplicial spaces can be expressed in terms of a single equivalence of simplicial spaces:

Definition

The *horizontal embedding* of a simplicial set $\mathbb{X}$ is the simplicial space X is given by $X_{k,l} := \mathbb{X}_k$ for all $k, l \in \mathbb{N}$.
For $n \in \mathbb{N}$ and $i \in \{0, \ldots, n\}$, let $F(n)$ be the horizontal embedding of Δ^n and $L(n,i)$ the horizontal embedding of Λ^n_i.

Theorem (Theorem A.21)

A Reedy fibrant simplicial space X is a Segal space if and only if

$$\mathrm{Map}(F(2), X) \to \mathrm{Map}(L(2,1), X)$$

is a trivial Reedy fibration.

Another “homotopy-theoretic” reason

Similarly, completeness for Segal spaces can be expressed in terms of a single equivalence of simplicial spaces:

Definition

Let $E(1)$ be the horizontal embedding of the nerve of the “walking isomorphism” (i. e. the category with two distinct objects and a unique isomorphism between them).

Theorem (Theorem A.25)

A Segal space X is complete if and only if

$$\mathrm{Map}(E(1), X) \to \mathrm{Map}(F(0), X) \cong X$$

is a trivial Reedy fibration.

Another "homotopy-theoretic" reason

Similarly, completeness for Segal spaces can be expressed in terms of a single equivalence of simplicial spaces:

Definition

Let $E(1)$ be the horizontal embedding of the nerve of the "walking isomorphism" (i. e. the category with two distinct objects and a unique isomorphism between them).

Theorem (Theorem A.25)

A Segal space X is complete if and only if

$$\mathrm{Map}(E(1), X) \to \mathrm{Map}(F(0), X) \cong X$$

is a trivial Reedy fibration.

Another "homotopy-theoretic" reason

Similarly, completeness for Segal spaces can be expressed in terms of a single equivalence of simplicial spaces:

Definition

Let $E(1)$ be the horizontal embedding of the nerve of the "walking isomorphism" (i. e. the category with two distinct objects and a unique isomorphism between them).

Theorem (Theorem A.25)

A Segal space X is complete if and only if

$$\mathrm{Map}(E(1), X) \to \mathrm{Map}(F(0), X) \cong X$$

is a trivial Reedy fibration.

Bear with me please

We will need to know a bit about the syntax and semantics of type theory to understand/appreciate why the synthetic theory works the way it does.

However, we won't state all the rules needed to make our type theory work (and be sloppy while dealing with rules we do state), and gloss over many details while describing how type-theoretic and homotopy-/category-theoretic notions relate to each other.

Bear with me please

We will need to know a bit about the syntax and semantics of type theory to understand/appreciate why the synthetic theory works the way it does.

However, we won't state all the rules needed to make our type theory work (and be sloppy while dealing with rules we do state), and gloss over many details while describing how type-theoretic and homotopy-/category-theoretic notions relate to each other.

Judgements

Definition

A *judgement* φ is, morally speaking, a statement we make in our theory.

Example

- $\bot$
- $s \leqslant t \wedge t \leqslant u$
- $1 = 0$
- A type
- $(a, b) : A \times B$
- $p \equiv (\pi_1(p), \pi_2(p))$
- $\prod_{x:A} B$ type (where x can occur in B as a variable)

Judgements

Definition

A *judgement* φ is, morally speaking, a statement we make in our theory.

Example

- $\bot$
- $s \leqslant t \wedge t \leqslant u$
- $1 = 0$
- A type
- $(a, b) : A \times B$
- $p \equiv (\pi_1(p), \pi_2(p))$
- $\prod_{x:A} B$ type (where x can occur in B as a variable)

Contexts

Definition

A *context* Γ is a ("well-formed") finite list of judgements. It can be thought of as the collection of assumptions we are currently working under.

We will work with expressions that look like

$$\Gamma \vdash \varphi,$$

which more or less means that

the judgement φ can be made under the assumption of the judgements in Γ.

Contexts

Definition

A *context* Γ is a (“well-formed”) finite list of judgements. It can be thought of as the collection of assumptions we are currently working under.

We will work with expressions that look like

$$\Gamma \vdash \varphi,$$

which more or less means that

the judgement φ *can be made under the assumption of the judgements in* Γ.

Inference rules

Inference rules are essentially the axioms which tell us when we can deduce statements of the form $\Gamma \vdash \varphi$. We depict such a rule as a list of premises separated from their conclusion by a vertical line.

Example

$$\frac{}{\Gamma \vdash \star : \mathbf{1}}, \qquad \frac{\Gamma \vdash x \equiv y \qquad \Gamma \vdash y \equiv z}{\Gamma \vdash x \equiv z}, \qquad \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi},$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash a : \mathbf{0}}{\Gamma \vdash \mathrm{ind}_{\mathbf{0}}(A, a) : A}, \qquad \frac{\Gamma \vdash A \text{ type} \qquad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \prod_{x:A} B \text{ type}}.$$

N.B. The last example demonstrates that we do need the extra inference layer to write down all axioms – we couldn't introduce and then bind variables with just $\vdash$.

Inference rules

Inference rules are essentially the axioms which tell us when we can deduce statements of the form $\Gamma \vdash \varphi$. We depict such a rule as a list of premises separated from their conclusion by a vertical line.

Example

$$\frac{}{\Gamma \vdash \star : \mathbf{1}}, \qquad \frac{\Gamma \vdash x \equiv y \qquad \Gamma \vdash y \equiv z}{\Gamma \vdash x \equiv z}, \qquad \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi},$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash a : \mathbf{0}}{\Gamma \vdash \mathrm{ind}_{\mathbf{0}}(A, a) : A}, \qquad \frac{\Gamma \vdash A \text{ type} \qquad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \prod_{x:A} B \text{ type}}.$$

N.B. The last example demonstrates that we do need the extra inference layer to write down all axioms – we couldn't introduce and then bind variables with just $\vdash$.

Inference rules

Inference rules are essentially the axioms which tell us when we can deduce statements of the form $\Gamma \vdash \varphi$. We depict such a rule as a list of premises separated from their conclusion by a vertical line.

Example

$$\frac{}{\Gamma \vdash \star : \mathbf{1}}, \qquad \frac{\Gamma \vdash x \equiv y \qquad \Gamma \vdash y \equiv z}{\Gamma \vdash x \equiv z}, \qquad \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi},$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash a : \mathbf{0}}{\Gamma \vdash \text{ind}_{\mathbf{0}}(A, a) : A}, \qquad \frac{\Gamma \vdash A \text{ type} \qquad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \prod_{x:A} B \text{ type}}.$$

N.B. The last example demonstrates that we do need the extra inference layer to write down all axioms – we couldn't introduce and then bind variables with just $\vdash$.

(Dependent) type theories

Definition

A *logical system* consists of a grammar describing well-formed judgements and a collection of inference rules.

Type theory is a blurry term for a logical system that involves "type-level" judgements like A type, "inhabitance" judgements like $a : A$ and "term-level" judgements like $x \equiv y$.

A type theory is called *dependent* if its types are allowed to depend on terms of other types.[1]

In fact, our "type theory" for ∞-categories will have two additional layers dealing with the combinatorics of simplices, their boundaries, horns etc.

[1] A typical case when a type depends on terms of other types is when one has an "equality type" $x =_A y$ for $x, y : A$.

(Dependent) type theories

Definition

A *logical system* consists of a grammar describing well-formed judgements and a collection of inference rules.

Type theory is a blurry term for a logical system that involves "type-level" judgements like A type, "inhabitance" judgements like $a : A$ and "term-level" judgements like $x \equiv y$.

A type theory is called *dependent* if its types are allowed to depend on terms of other types.[1]

In fact, our "type theory" for ∞-categories will have two additional layers dealing with the combinatorics of simplices, their boundaries, horns etc.

[1] A typical case when a type depends on terms of other types is when one has an "equality type" $x =_A y$ for $x, y : A$.

(Dependent) type theories

Definition

A *logical system* consists of a grammar describing well-formed judgements and a collection of inference rules.

Type theory is a blurry term for a logical system that involves “type-level” judgements like A type, “inhabitance” judgements like $a : A$ and “term-level” judgements like $x \equiv y$.

A type theory is called *dependent* if its types are allowed to depend on terms of other types.[1]

In fact, our “type theory” for ∞-categories will have two additional layers dealing with the combinatorics of simplices, their boundaries, horns etc.

[1] A typical case when a type depends on terms of other types is when one has an “equality type” $x =_A y$ for $x, y : A$.

(Dependent) type theories

Definition

A *logical system* consists of a grammar describing well-formed judgements and a collection of inference rules.
Type theory is a blurry term for a logical system that involves "type-level" judgements like A type, "inhabitance" judgements like $a : A$ and "term-level" judgements like $x \equiv y$.
A type theory is called *dependent* if its types are allowed to depend on terms of other types.[1]

In fact, our "type theory" for ∞-categories will have two additional layers dealing with the combinatorics of simplices, their boundaries, horns etc.

[1] A typical case when a type depends on terms of other types is when one has an "equality type" $x =_A y$ for $x, y : A$.

Models

Definition

A *model* of a logical system is a concrete mathematical object in which its judgements correspond to concrete mathematical statements.

Usually, one agrees on a common form for those "models".

Example

There is a logical system for group theory, with judgements like $\forall g\ \forall h\ g \cdot h \cdot g^{-1} \cdot h = \mathbf{e}$ and inference rules describing group axioms.

One can interpret $\cdot$, $\mathbf{e}$ and $(-)^{-1}$ in any group, so it's reasonable to say that every concrete group is a model for this logical system.

Models

Definition

A *model* of a logical system is a concrete mathematical object in which its judgements correspond to concrete mathematical statements.

Usually, one agrees on a common form for those "models".

Example

There is a logical system for group theory, with judgements like $\forall g\ \forall h\ g \cdot h \cdot g^{-1} \cdot h = \mathbf{e}$ and inference rules describing group axioms.

One can interpret $\cdot$, $\mathbf{e}$ and $(-)^{-1}$ in any group, so it's reasonable to say that every concrete group is a model for this logical system.

Models

Definition

A *model* of a logical system is a concrete mathematical object in which its judgements correspond to concrete mathematical statements.

Usually, one agrees on a common form for those “models”.

Example

There is a logical system for group theory, with judgements like $\forall g\ \forall h\ g \cdot h \cdot g^{-1} \cdot h = \mathbf{e}$ and inference rules describing group axioms.
One can interpret $\cdot$, $\mathbf{e}$ and $(-)^{-1}$ in any group, so it's reasonable to say that every concrete group is a model for this logical system.

Basic categorical semantics for type theories

There is an evident similarity between type-theoretic and category-theoretic constructions which you may have seen quite a few times by now:

Category theory	**Type theory**
object	type
morphism	term of a function type
initial/terminal object	$\mathbf{0}/\mathbf{1}$
categorical product	product type
...	...

Hence we should be able to interpret certain categories as a "category of types" modeling our type theory.

Basic categorical semantics for type theories

There is an evident similarity between type-theoretic and category-theoretic constructions which you may have seen quite a few times by now:

Category theory	**Type theory**
object	type
morphism	term of a function type
initial/terminal object	$\mathbf{0}/\mathbf{1}$
categorical product	product type
...	...

Hence we should be able to interpret certain categories as a "category of types" modeling our type theory.

Incorporating dependence into the semantics

Given a type A, there are (in general) types B which only exist under the assumption $x : A$ (i. e. $x : A \vdash B$ type). Moreover, we could iterate this process, i. e. consider types that exist in the context $x : A, y : B$; or more generally have a "category of types" for every context Γ.

In order to reflect this in our models, we will enhance our "category of types" in the following ways:

1. We will work with a category $\mathcal{C}$ of *contexts*, where morphisms are "inferences".
2. We will have a Grothendieck fibration $\mathcal{T} \to \mathcal{C}$ where the fiber over $\Gamma \in \mathcal{C}$ corresponds to "the category of types that exist in the context Γ".
3. We will have a functor $\mathcal{T} \to \mathcal{C}^{[1]}$ that corresponds to mapping A over Γ (meaning that $\Gamma \vdash A$ type) to $(\Gamma, x : A) \to \Gamma$.

Incorporating dependence into the semantics

Given a type A, there are (in general) types B which only exist under the assumption $x : A$ (i. e. $x : A \vdash B$ type). Moreover, we could iterate this process, i. e. consider types that exist in the context $x : A, y : B$; or more generally have a "category of types" for every context Γ.

In order to reflect this in our models, we will enhance our "category of types" in the following ways:

1. We will work with a category $\mathcal{C}$ of *contexts*, where morphisms are "inferences".
2. We will have a Grothendieck fibration $\mathcal{T} \to \mathcal{C}$ where the fiber over $\Gamma \in \mathcal{C}$ corresponds to "the category of types that exist in the context Γ".
3. We will have a functor $\mathcal{T} \to \mathcal{C}^{[1]}$ that corresponds to mapping A over Γ (meaning that $\Gamma \vdash A$ type) to $(\Gamma, x : A) \to \Gamma$.

Incorporating dependence into the semantics

Given a type A, there are (in general) types B which only exist under the assumption $x : A$ (i. e. $x : A \vdash B$ type). Moreover, we could iterate this process, i. e. consider types that exist in the context $x : A, y : B$; or more generally have a "category of types" for every context Γ.

In order to reflect this in our models, we will enhance our "category of types" in the following ways:

1. We will work with a category $\mathcal{C}$ of *contexts*, where morphisms are "inferences".
2. We will have a Grothendieck fibration $\mathcal{T} \to \mathcal{C}$ where the fiber over $\Gamma \in \mathcal{C}$ corresponds to "the category of types that exist in the context Γ".
3. We will have a functor $\mathcal{T} \to \mathcal{C}^{[1]}$ that corresponds to mapping A over Γ (meaning that $\Gamma \vdash A$ type) to $(\Gamma, x : A) \to \Gamma$.

Incorporating dependence into the semantics

Given a type A, there are (in general) types B which only exist under the assumption $x : A$ (i. e. $x : A \vdash B$ type). Moreover, we could iterate this process, i. e. consider types that exist in the context $x : A, y : B$; or more generally have a "category of types" for every context Γ.

In order to reflect this in our models, we will enhance our "category of types" in the following ways:

1. We will work with a category $\mathcal{C}$ of *contexts*, where morphisms are "inferences".
2. We will have a Grothendieck fibration $\mathcal{T} \to \mathcal{C}$ where the fiber over $\Gamma \in \mathcal{C}$ corresponds to "the category of types that exist in the context Γ".
3. We will have a functor $\mathcal{T} \to \mathcal{C}^{[1]}$ that corresponds to mapping A over Γ (meaning that $\Gamma \vdash A$ type) to $(\Gamma, x : A) \to \Gamma$.

Incorporating dependence into the semantics

Given a type A, there are (in general) types B which only exist under the assumption $x : A$ (i. e. $x : A \vdash B$ type). Moreover, we could iterate this process, i. e. consider types that exist in the context $x : A, y : B$; or more generally have a "category of types" for every context Γ.

In order to reflect this in our models, we will enhance our "category of types" in the following ways:

1. We will work with a category $\mathcal{C}$ of *contexts*, where morphisms are "inferences".
2. We will have a Grothendieck fibration $\mathcal{T} \to \mathcal{C}$ where the fiber over $\Gamma \in \mathcal{C}$ corresponds to "the category of types that exist in the context Γ".
3. We will have a functor $\mathcal{T} \to \mathcal{C}^{[1]}$ that corresponds to mapping A over Γ (meaning that $\Gamma \vdash A$ type) to $(\Gamma, x : A) \to \Gamma$.

Dependent types in practice

Usually, the fiber $\mathcal{T}_C$ over $C \in \mathcal{C}$ is some sort of overcategory $\mathcal{C}_{/C}$, so we can extend our analogy as follows:

Category theory	Type theory
terminal object $*$	empty context
$D \to C$ in $\mathcal{C}_{/C}$	$c : C \vdash D_c$ type
composite $D \to C \to *$	$c : C \vdash D_c$ type $\rightsquigarrow \vdash \sum_{c:C} D_c$ type
right adjoint to $C \times_* (-)$	$c : C \vdash D_c$ type $\rightsquigarrow \vdash \prod_{c:C} D_c$ type
...	...

Dependent types in practice

Usually, the fiber $\mathcal{T}_C$ over $C \in \mathcal{C}$ is some sort of overcategory $\mathcal{C}_{/C}$, so we can extend our analogy as follows:

Category theory	**Type theory**
terminal object $*$	empty context
$D \to C$ in $\mathcal{C}_{/C}$	$c : C \vdash D_c$ type
composite $D \to C \to *$	$c : C \vdash D_c$ type $\rightsquigarrow \vdash \sum_{c:C} D_c$ type
right adjoint to $C \times_* (-)$	$c : C \vdash D_c$ type $\rightsquigarrow \vdash \prod_{c:C} D_c$ type
...	...

A problem

Recall that in order to be able to express the Segal condition and completeness for a simplicial space X, we need to work with simplicial spaces of the form $\mathrm{Map}(K, X)$ for horizontal embeddings K of certain "small" simplicial sets like Δ^n or Λ^n_i.

However, these horizontal embeddings are not (necessarily) Reedy fibrant, so we need to incorporate them into our synthetic theory separately.

A problem

Recall that in order to be able to express the Segal condition and completeness for a simplicial space X, we need to work with simplicial spaces of the form $\mathrm{Map}(K, X)$ for horizontal embeddings K of certain "small" simplicial sets like Δ^n or Λ^n_i.

However, these horizontal embeddings are not (necessarily) Reedy fibrant, so we need to incorporate them into our synthetic theory separately.

A solution

We will restrict our attention to subsimplicial sets of $(\Delta^1)^n$'s (which include in particular Δ^n's, $\partial\Delta^n$'s, Λ^n_i's etc.).

Definition

A *cube* is a finite product of copies of $[1]$. We consider it as a partial order.

A *tope* ϕ *(in* k *variables)* is a proposition in k variables constructed using 0 and 1 (where $0, 1 \in [1]$), $\equiv$, $\leqslant$, conjuctions and disjunctions.

A *shape* is a subset of a cube $[1]^k$ of the form $\{(t_1, \ldots, t_k) \in I | \phi(t_1, \ldots, t_k)\}$ where ϕ is a tope in k variables.

Example

- $\Delta^n \cong \{(t_1, \ldots, t_n) \in [1]^n | t_1 \leqslant \ldots \leqslant t_n\}$ can be realized as a shape.
- Λ^2_1 can be realized as $\{(s, t) \in [1]^2 | s \equiv 1 \vee t \equiv 0\}$.

A solution

We will restrict our attention to subsimplicial sets of $(\Delta^1)^n$'s (which include in particular Δ^n's, $\partial\Delta^n$'s, Λ^n_i's etc.).

Definition

A *cube* is a finite product of copies of $[1]$. We consider it as a partial order.

A *tope* ϕ *(in* k *variables)* is a proposition in k variables constructed using 0 and 1 (where $0, 1 \in [1]$), $\equiv$, $\leqslant$, conjuctions and disjunctions.

A *shape* is a subset of a cube $[1]^k$ of the form $\{(t_1, \ldots, t_k) \in I | \phi(t_1, \ldots, t_k)\}$ where ϕ is a tope in k variables.

Example

- $\Delta^n \cong \{(t_1, \ldots, t_n) \in [1]^n | t_1 \leqslant \ldots \leqslant t_n\}$ can be realized as a shape.
- Λ^2_1 can be realized as $\{(s, t) \in [1]^2 | s \equiv 1 \vee t \equiv 0\}$.

A solution

We will restrict our attention to subsimplicial sets of $(\Delta^1)^n$'s (which include in particular Δ^n's, $\partial\Delta^n$'s, Λ^n_i's etc.).

Definition

A *cube* is a finite product of copies of $[1]$. We consider it as a partial order.

A *tope* ϕ *(in* k *variables)* is a proposition in k variables constructed using 0 and 1 (where $0, 1 \in [1]$), $\equiv$, $\leqslant$, conjuctions and disjunctions.

A *shape* is a subset of a cube $[1]^k$ of the form $\{(t_1, \ldots, t_k) \in I | \phi(t_1, \ldots, t_k)\}$ where ϕ is a tope in k variables.

Example

- $\Delta^n \cong \{(t_1, \ldots, t_n) \in [1]^n | t_1 \leqslant \ldots \leqslant t_n\}$ can be realized as a shape.
- Λ^2_1 can be realized as $\{(s, t) \in [1]^2 | s \equiv 1 \vee t \equiv 0\}$.

A solution

We will restrict our attention to subsimplicial sets of $(\Delta^1)^n$'s (which include in particular Δ^n's, $\partial\Delta^n$'s, Λ^n_i's etc.).

Definition

A *cube* is a finite product of copies of $[1]$. We consider it as a partial order.

A *tope* ϕ *(in* k *variables)* is a proposition in k variables constructed using 0 and 1 (where $0, 1 \in [1]$), $\equiv$, $\leqslant$, conjuctions and disjunctions.

A *shape* is a subset of a cube $[1]^k$ of the form $\{(t_1, \ldots, t_k) \in I | \phi(t_1, \ldots, t_k)\}$ where ϕ is a tope in k variables.

Example

- $\Delta^n \cong \{(t_1, \ldots, t_n) \in [1]^n | t_1 \leqslant \ldots \leqslant t_n\}$ can be realized as a shape.
- Λ^2_1 can be realized as $\{(s, t) \in [1]^2 | s \equiv 1 \vee t \equiv 0\}$.

A solution

We will restrict our attention to subsimplicial sets of $(\Delta^1)^n$'s (which include in particular Δ^n's, $\partial\Delta^n$'s, Λ^n_i's etc.).

Definition

A *cube* is a finite product of copies of $[1]$. We consider it as a partial order.
A *tope* ϕ *(in* k *variables)* is a proposition in k variables constructed using 0 and 1 (where $0, 1 \in [1]$), $\equiv$, $\leqslant$, conjuctions and disjunctions.
A *shape* is a subset of a cube $[1]^k$ of the form $\{(t_1, \ldots, t_k) \in I | \phi(t_1, \ldots, t_k)\}$ where ϕ is a tope in k variables.

Example

- $\Delta^n \cong \{(t_1, \ldots, t_n) \in [1]^n | t_1 \leqslant \ldots \leqslant t_n\}$ can be realized as a shape.
- Λ^2_1 can be realized as $\{(s, t) \in [1]^2 | s \equiv 1 \vee t \equiv 0\}$.

A solution

We will restrict our attention to subsimplicial sets of $(\Delta^1)^n$'s (which include in particular Δ^n's, $\partial\Delta^n$'s, Λ^n_i's etc.).

Definition

A *cube* is a finite product of copies of $[1]$. We consider it as a partial order.

A *tope* ϕ *(in* k *variables)* is a proposition in k variables constructed using 0 and 1 (where $0, 1 \in [1]$), $\equiv$, $\leqslant$, conjuctions and disjunctions.

A *shape* is a subset of a cube $[1]^k$ of the form $\{(t_1, \ldots, t_k) \in I | \phi(t_1, \ldots, t_k)\}$ where ϕ is a tope in k variables.

Example

- $\Delta^n \cong \{(t_1, \ldots, t_n) \in [1]^n | t_1 \leqslant \ldots \leqslant t_n\}$ can be realized as a shape.
- Λ^2_1 can be realized as $\{(s, t) \in [1]^2 | s \equiv 1 \vee t \equiv 0\}$.

Cubes and topes in type theory

We will introduce new syntax in our logical system to encode cubes and topes.

This will result in three different types of judgements and contexts: One for cubes (usually Ξ), one for topes (usually Φ) and one for usual type theory with dependent types (usually Γ).

Some inference rules for the first two "layers" are:

Example

$$\frac{I \text{ cube} \qquad J \text{ cube}}{I \times J \text{ cube}}, \quad \frac{}{\Xi \vdash 0 : [1]},$$

$$\frac{\Xi \vdash s : I \qquad \Xi \vdash t : I}{\Xi \vdash s \equiv t \text{ tope}}, \quad \frac{\Xi | \Phi \vdash \phi \wedge \psi}{\Xi | \Phi \vdash \psi}, \quad \frac{}{x : [1] | \cdot \vdash x \leqslant 1}$$

Cubes and topes in type theory

We will introduce new syntax in our logical system to encode cubes and topes.
This will result in three different types of judgements and contexts: One for cubes (usually Ξ), one for topes (usually Φ) and one for usual type theory with dependent types (usually Γ).
Some inference rules for the first two "layers" are:

Example

$$\frac{I \text{ cube} \qquad J \text{ cube}}{I \times J \text{ cube}}, \quad \frac{}{\Xi \vdash 0 : [1]},$$

$$\frac{\Xi \vdash s : I \qquad \Xi \vdash t : I}{\Xi \vdash s \equiv t \text{ tope}}, \quad \frac{\Xi | \Phi \vdash \phi \wedge \psi}{\Xi | \Phi \vdash \psi}, \quad \frac{}{x : [1] | \cdot \vdash x \leqslant 1}$$

Cubes and topes in type theory

We will introduce new syntax in our logical system to encode cubes and topes.
This will result in three different types of judgements and contexts: One for cubes (usually Ξ), one for topes (usually Φ) and one for usual type theory with dependent types (usually Γ).
Some inference rules for the first two "layers" are:

Example

$$\frac{I \text{ cube} \qquad J \text{ cube}}{I \times J \text{ cube}}, \quad \frac{}{\Xi \vdash 0 : [1]},$$

$$\frac{\Xi \vdash s : I \qquad \Xi \vdash t : I}{\Xi \vdash s \equiv t \text{ tope}}, \quad \frac{\Xi | \Phi \vdash \phi \wedge \psi}{\Xi | \Phi \vdash \psi}, \quad \frac{}{x : [1] | \cdot \vdash x \leqslant 1}$$

Shapes in type theory

Convention

$$\{t : I | \phi\} \text{ shape}$$

will be a shorthand for

$$I \text{ cube} \quad \text{and} \quad t : I \vdash \phi \text{ tope.}$$

Extension types

Next, we determine how shapes and types interact.

For every shape $\{t : I|\psi\}$ and every type A we would like to have a type of functions $\{t : I|\psi\} \to A$.

For this, it is enough to construct a type of extensions

for $t : I|\phi \vdash \psi$.

Now, we can also make the target "vary in elements of $\{t : I|\phi\}$" – in other words, instead of extensions of functions, we can more generally look for extensions of a section of a fibration.

Extension types

Next, we determine how shapes and types interact.

For every shape $\{t : I|\psi\}$ and every type A we would like to have a type of functions $\{t : I|\psi\} \to A$.

For this, it is enough to construct a type of extensions

$$\begin{array}{ccc} \{t : I|\phi\} & \longrightarrow & A \\ \big\downarrow & \nearrow & \\ \{t : I|\psi\} & & \end{array}$$

for $t : I|\phi \vdash \psi$.

Now, we can also make the target "vary in elements of $\{t : I|\phi\}$" – in other words, instead of extensions of functions, we can more generally look for extensions of a section of a fibration.

Extension types

Next, we determine how shapes and types interact.
For every shape $\{t : I|\psi\}$ and every type A we would like to have a type of functions $\{t : I|\psi\} \to A$.
For this, it is enough to construct a type of extensions

$$\begin{array}{ccc} \{t : I|\phi\} & \longrightarrow & A \\ \big\downarrow & \nearrow & \\ \{t : I|\psi\} & & \end{array}$$

for $t : I|\phi \vdash \psi$.

Now, we can also make the target "vary in elements of $\{t : I|\phi\}$" – in other words, instead of extensions of functions, we can more generally look for extensions of a section of a fibration.

Extension types

Next, we determine how shapes and types interact.
For every shape $\{t : I|\psi\}$ and every type A we would like to have a type of functions $\{t : I|\psi\} \to A$.
For this, it is enough to construct a type of extensions

$$\begin{array}{ccc} \{t : I|\phi\} & \longrightarrow & A \\ \big\downarrow & \nearrow & \\ \{t : I|\psi\} & & \end{array}$$

for $t : I|\phi \vdash \psi$.
Now, we can also make the target "vary in elements of $\{t : I|\phi\}$" – in other words, instead of extensions of functions, we can more generally look for extensions of a section of a fibration.

Some rules for extension types

$$\frac{\{t : I|\phi\} \text{ shape} \quad \{t : I|\psi\} \text{ shape} \quad t : I|\phi \vdash \psi \quad \Xi|\Phi \vdash \Gamma \text{ context} \quad \Xi, t : I|\Phi, \psi|\Gamma \vdash A \text{ type} \quad \Xi, t : I|\Phi, \phi|\Gamma \vdash a : A}{\Xi|\Phi|\Gamma \vdash \left\langle \prod_{t:I|\psi} A|^{\phi}_{a} \right\rangle}$$

$$\frac{\{t : I|\phi\} \text{ shape} \quad \{t : I|\psi\} \text{ shape} \quad t : I|\phi \vdash \psi \quad \Xi|\Phi \vdash \Gamma \text{ context} \quad \Xi, t : I|\Phi, \psi|\Gamma \vdash A \text{ type} \quad \Xi, t : I|\Phi, \phi|\Gamma \vdash a : A \quad \Xi, t : I|\Phi, \psi|\Gamma \vdash b : A \quad \Xi, t : I|\Phi, \phi|\Gamma \vdash b \equiv a}{\Xi|\Phi|\Gamma \vdash \lambda t^{I|\psi}.b : \left\langle \prod_{t:I|\psi} A|^{\phi}_{a} \right\rangle}$$

The final model