

A type theory for synthetic ∞ -categories after a homonymous paper by E. Riehl and M. Shulman

Aras Ergus

July 1, 2020

This work is licensed under a Creative Commons "Attribution-ShareAlike 4.0 International" license.





▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

Motivation

Why a synthetic theory of complete Segal spaces?

Syntax of type theory

Semantics of type theory

The type theory with shapes

Motivation	Why CSS?	Interlude	Syntax	Semantics	TT with shapes
•00	0000	0	0000	0000	000000

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.

Motivation	Why CSS?	Interlude	Syntax	Semantics	TT with shapes
•00	0000	0	0000	0000	0000000

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.

Motivation	Why CSS?	Interlude	Syntax	Semantics	TT with shapes
●00	0000	0	0000	0000	0000000

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.

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

Motivation	Why CSS?	Interlude	Syntax	Semantics	TT with shapes
●00	0000	0	0000	0000	0000000

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.

Motivation	Why CSS?	Interlude	Syntax	Semantics	TT with shapes
●00	0000	0	0000	0000	0000000

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.

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

Motivation	Why CSS?	Interlude	Syntax	Semantics	TT with shapes
●00	0000	0	0000	0000	0000000

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.

Motivation
000

Interlude 0 Syntax 0000 Semantics 0000 TT with shapes

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:

$$\mathcal{C}_a \longleftrightarrow \operatorname{Nat}(\operatorname{Hom}(a, -), \mathcal{C}_{(-)})$$
$$u \longmapsto (f \mapsto \mathcal{C}_f(u))$$
$$a(\operatorname{Id}_{a}) \xleftarrow{} da_{a}$$

・ロト・日本・日本・日本・日本・日本

Motivation 000 Why CSS?

Interlude 0 Syntax 0000 Semantics 0000

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

TT with shapes

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:

$$\mathcal{C}_{a} \longleftrightarrow \operatorname{Nat}(\operatorname{Hom}(a, -), \mathcal{C}_{(-)})$$
$$u \longmapsto (f \mapsto \mathcal{C}_{f}(u))$$
$$c(\operatorname{Id}_{a}) \longleftrightarrow c(g)$$

Motivation 000 Why CSS? 0000 Interlude 0 Syntax 0000 Semantics 0000 TT with shapes

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:

$$\mathcal{C}_{a} \longleftrightarrow \mathsf{Nat}(\mathsf{Hom}(a, -), \mathcal{C}_{(-)})$$
$$u \longmapsto (f \mapsto \mathcal{C}_{f}(u))$$
$$\varphi(\mathsf{Id}_{a}) \longleftrightarrow \varphi$$

・ロト・日本・日本・日本・日本・日本



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.

▲ロト ▲冊 ▶ ▲ ヨ ▶ ▲ ヨ ▶ ● の Q @



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.

▲ロト ▲冊 ▶ ▲ ヨ ▶ ▲ ヨ ▶ ● の Q @



The model we will mimic

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

▲ロト ▲冊 ▶ ▲ ヨ ▶ ▲ ヨ ▶ ● の Q @

There are several reasons for this choice.



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".



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".



TT with shapes 0000000

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

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 X is the simplicial space X is given by $X_{k,l} := X_k$ for all $k, l \in \mathbb{N}$. For $n \in \mathbb{N}$ and $i \in \{0, ..., n\}$, 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

$Map(F(2), X) \rightarrow Map(L(2, 1), X)$



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 X is the simplicial space X is given by $X_{k,l} := X_k$ for all $k, l \in \mathbb{N}$. For $n \in \mathbb{N}$ and $i \in \{0, ..., n\}$, let F(n) be the horizontal embedding of Δ^n and L(n, i) the horizontal embedding of Λ_i^n .

Theorem (Theorem A.21)

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

$$Map(F(2), X) \rightarrow Map(L(2, 1), X)$$

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @



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 X is the simplicial space X is given by $X_{k,l} := X_k$ for all $k, l \in \mathbb{N}$. For $n \in \mathbb{N}$ and $i \in \{0, ..., n\}$, let F(n) be the horizontal embedding of Δ^n and L(n, i) the horizontal embedding of Λ_i^n .

Theorem (Theorem A.21)

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

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

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @



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

 $Map(E(1), X) \rightarrow Map(F(0), X) \cong X$

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @



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

```
\mathsf{Map}(E(1),X) \to \mathsf{Map}(F(0),X) \cong X
```

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @



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

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

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @



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.

イロト 不得 ト イヨト イヨト 三日



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.



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

Example

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



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

Example

- ⊥
- $s \leqslant t \land t \leqslant u$
- 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)



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 .



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 .

ion	Wh
	00

	n	t	e	r	l	u	d	1
1	0							

Syntax 00●0 Semantics 0000 TT with shapes

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

$$\frac{\Gamma \vdash x \equiv y \quad \Gamma \vdash y \equiv z}{\Gamma \vdash x \equiv z}, \qquad \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \lor \psi},$$

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

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 .

1	Why CSS?
	0000

Int	erl	ud	е
0			

Syntax 00●0 Semantics 0000 TT with shapes

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

$$\frac{\Gamma \vdash x \equiv y \quad \Gamma \vdash y \equiv z}{\Gamma \vdash x \equiv z}, \qquad \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \lor \psi},$$
$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash a : \mathbf{0}}{\Gamma \vdash \text{ ind}_{\mathbf{0}}(A, a) : A}, \qquad \frac{\Gamma \vdash A \text{ type} \quad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \prod_{x : A} B \text{ type}}.$$

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 .

1	Why CSS?
	0000

	n	t	e	r	l	u	d	e	
¢	С								

Syntax 0000 Semantics 0000 TT with shapes

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

$$\frac{\Gamma \vdash x \equiv y \quad \Gamma \vdash y \equiv z}{\Gamma \vdash x \equiv z}, \qquad \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \lor \psi},$$
$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash a : \mathbf{0}}{\Gamma \vdash \text{ ind}_{\mathbf{0}}(A, a) : A}, \qquad \frac{\Gamma \vdash A \text{ type} \quad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \prod_{x : A} B \text{ type}}.$$

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 .



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.

¹A 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.



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.

¹A 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.



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.

¹A 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.



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 $\infty\text{-}categories$ will have two additional layers dealing with the combinatorics of simplices, their boundaries, horns etc.

¹A 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.



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



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



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 , **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	0/1
categorical product	product type

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

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @



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	0/1
categorical product	product type

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

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

Why CSS?InterludeSyntax000000000

Semantics

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

TT with shapes 0000000

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 .

- 1. We will work with a category \mathcal{C} of *contexts*, where morphisms are "inferences".
- We will have a Grothendieck fibration T → C where the fiber over Γ ∈ 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 $(\Gamma, x : A) \to \Gamma$.

Semantics

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

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 .

- 1. We will work with a category C of *contexts*, where morphisms are "inferences".
- We will have a Grothendieck fibration T → C where the fiber over Γ ∈ C corresponds to "the category of types that exist in the context Γ".
- We will have a functor T → C^[1] that corresponds to mapping A over Γ (meaning that Γ ⊢ A type) to (Γ, x : A) → Γ.

Semantics

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

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 .

- 1. We will work with a category ${\mathcal C}$ of $\mathit{contexts},$ 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 ".
- We will have a functor T → C^[1] that corresponds to mapping A over Γ (meaning that Γ ⊢ A type) to (Γ, x : A) → Γ.

Semantics

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

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 .

- 1. We will work with a category ${\mathcal C}$ of $\mathit{contexts},$ 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 ".
- We will have a functor T → C^[1] that corresponds to mapping A over Γ (meaning that Γ ⊢ A type) to (Γ, x : A) → Γ.

Semantics

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

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 .

- 1. We will work with a category ${\mathcal C}$ of $\mathit{contexts},$ 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 ".
- We will have a functor T → C^[1] that corresponds to mapping A over Γ (meaning that Γ ⊢ A type) to (Γ, x : A) → Γ.



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 $D \to C \to *$	$c: \mathcal{C} \vdash D_c$ type $\rightsquigarrow \vdash \sum_{c:\mathcal{C}} D_c$ type
right adjoint to $C imes_{igstar} (-)$	$c: C \vdash D_c$ type $\rightsquigarrow \vdash \prod_{c:C} D_c$ type

▲ロト ▲周ト ▲ヨト ▲ヨト ヨー のくで



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
$D \to C \text{ in } \mathcal{C}_{/C}$	$c: C \vdash D_c$ type
composite $D \to C \to *$	$c: C \vdash D_c$ type $\rightsquigarrow \vdash \sum_{c:C} D_c$ type
right adjoint to $C \times_{\mathbf{*}} (-)$	$c: C \vdash D_c$ type $\rightsquigarrow \vdash \prod_{c:C} D_c$ type

▲ロト ▲周ト ▲ヨト ▲ヨト ヨー のくで



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 Map(K, X) for horizontal embeddings K of certain "small" simplicial sets like Δ^n or Λ_i^n .

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

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @



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 Map(K, X) for horizontal embeddings K of certain "small" simplicial sets like Δ^n or Λ_i^n .

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

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

1	Why CSS?	Inter
	0000	0

Interlude O Syntax 0000 Semantics 0000

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

TT with shapes

A solution

We will restrict our attention to subsimplicial sets of $(\Delta^1)^{n'}$ s (which include in particular $\Delta^{n'}$ s, $\partial \Delta^{n'}$ s, $\Lambda_i^{n'}$ 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 , \leq , 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.

- $\Delta^n \cong \{(t_1, \ldots, t_n) \in [1]^n | t_1 \leqslant \ldots \leqslant t_n\}$ can be realized as a shape.
- Λ_1^2 can be realized as $\{(s,t) \in [1]^2 | s \equiv 1 \lor t \equiv 0\}$.

ion	Why CSS?	Interlu
	0000	0

I	n	t	e	r	I	u	d	е	
1									

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

TT with shapes 000000

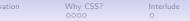
A solution

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_{i}^{n} 's etc.).

Definition

A cube is a finite product of copies of |1|. We consider it as a partial order.

- $\Delta^n \cong \{(t_1, \ldots, t_n) \in [1]^n | t_1 \leqslant \ldots \leqslant t_n\}$ can be realized as a
- Λ_1^2 can be realized as $\{(s, t) \in [1]^2 | s \equiv 1 \lor t \equiv 0\}$.



Synta

Semantics 0000

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

TT with shapes

A solution

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

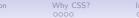
Definition

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 , \leq , 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.

- $\Delta^n \cong \{(t_1, \ldots, t_n) \in [1]^n | t_1 \leqslant \ldots \leqslant t_n\}$ can be realized as a shape.
- Λ_1^2 can be realized as $\{(s,t) \in [1]^2 | s \equiv 1 \lor t \equiv 0\}$.



I	n	t	e	r	l	u	d	е
4								

Syntax 0000 Semantics 0000

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

TT with shapes

A solution

We will restrict our attention to subsimplicial sets of $(\Delta^1)^{n'}$ s (which include in particular $\Delta^{n'}$ s, $\partial \Delta^{n'}$ s, $\Lambda_i^{n'}$ 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 , \leq , 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.

- $\Delta^n \cong \{(t_1, \ldots, t_n) \in [1]^n | t_1 \leqslant \ldots \leqslant t_n\}$ can be realized as a shape.
- Λ_1^2 can be realized as $\{(s,t) \in [1]^2 | s \equiv 1 \lor t \equiv 0\}.$



I	n	t	e	r	l	u	d	e	
(

Syntax 0000 Semantics 0000

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

TT with shapes

A solution

We will restrict our attention to subsimplicial sets of $(\Delta^1)^{n'}$ s (which include in particular $\Delta^{n'}$ s, $\partial \Delta^{n'}$ s, $\Lambda_i^{n'}$ 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 , \leq , 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.

- $\Delta^n \cong \{(t_1, \dots, t_n) \in [1]^n | t_1 \leqslant \dots \leqslant t_n\}$ can be realized as a shape.
- Λ_1^2 can be realized as $\{(s,t) \in [1]^2 | s \equiv 1 \lor t \equiv 0\}.$

on	Why CSS?	In
	0000	0

In	te	er	lι	ld	е	
0						

Syntax 0000 Semantics 0000 TT with shapes

A solution

We will restrict our attention to subsimplicial sets of $(\Delta^1)^{n'}$ s (which include in particular $\Delta^{n'}$ s, $\partial \Delta^{n'}$ s, $\Lambda_i^{n'}$ 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 , \leq , 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.

- $\Delta^n \cong \{(t_1, \dots, t_n) \in [1]^n | t_1 \leqslant \dots \leqslant t_n\}$ can be realized as a shape.
- Λ_1^2 can be realized as $\{(s,t) \in [1]^2 | s \equiv 1 \lor t \equiv 0\}.$



Vhy CSS?

nterlude

Syntax 0000 Semantics 0000

▲ロト ▲周ト ▲ヨト ▲ヨト ヨー のくで

TT with shapes

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:

$$\frac{I \text{ cube } J \text{ cube }}{I \times J \text{ cube }}, \frac{}{\Xi \vdash 0:[1]},$$
$$\frac{\Xi \vdash s: I \quad \Xi \vdash t: I}{\Xi \vdash s \equiv t \text{ tope }}, \frac{\Xi | \Phi \vdash \phi \land \psi}{\Xi | \Phi \vdash \psi}, \frac{}{x:[1]| \cdot \vdash x \leqslant 1}$$



/hy CSS? 000 Interlude O Syntax 0000 Semantics 0000

▲ロト ▲周ト ▲ヨト ▲ヨト ヨー のくで

TT with shapes

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:

$$\frac{I \text{ cube } J \text{ cube }}{I \times J \text{ cube }}, \frac{}{\Xi \vdash 0:[1]},$$
$$\frac{\Xi \vdash s: I \quad \Xi \vdash t: I}{\Xi \vdash s \equiv t \text{ tope }}, \frac{\Xi | \Phi \vdash \phi \land \psi}{\Xi | \Phi \vdash \psi}, \frac{}{x:[1]| \cdot \vdash x \leq 1}$$



hy CSS?

Interlude O Syntax 0000 Semantics 0000

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

TT with shapes

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:

$$\frac{I \text{ cube } J \text{ cube }}{I \times J \text{ cube }}, \frac{1}{\Xi \vdash 0:[1]},$$
$$\frac{\Xi \vdash s: I \quad \Xi \vdash t: I}{\Xi \vdash s \equiv t \text{ tope }}, \frac{\Xi \mid \Phi \vdash \phi \land \psi}{\Xi \mid \Phi \vdash \psi}, \frac{1}{x:[1] \mid \cdot \vdash x \leqslant 1}$$



Shapes in type theory

Convention

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

will be a shorthand for

I cube and $t: I \vdash \phi$ tope.

▲ロト ▲周ト ▲ヨト ▲ヨト ヨー のくで



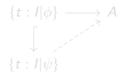
Why CSS? 0000 Interlude O Syntax 0000 Semantics 0000 TT with shapes

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 $\{t : I | \psi\} \rightarrow A$.

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



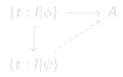
for $t : I | \phi \vdash \psi$.



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 $\{t : I | \psi\} \rightarrow A$.

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



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

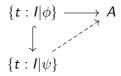


Extension types

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 $\{t : I | \psi\} \rightarrow A$.

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



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

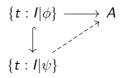


Extension types

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 $\{t : I | \psi\} \rightarrow A$.

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



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

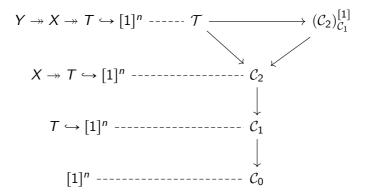


Some rules for extension types

$$\begin{cases} t: I | \phi \rangle \text{ shape} \\ \{t: I | \psi \} \text{ shape} \\ t: I | \phi \vdash \psi \quad \equiv | \Phi \vdash \Gamma \text{ context} \\ \exists, t: I | \Phi, \psi | \Gamma \vdash A \text{ type} \quad \exists, t: I | \Phi, \phi | \Gamma \vdash a: A \\ \hline \equiv | \Phi | \Gamma \vdash \langle \prod_{t:I | \psi} A |_a^{\phi} \rangle \\ \\ \{t: I | \psi \} \text{ shape} \quad t: I | \phi \vdash \psi \quad \equiv | \Phi \vdash \Gamma \text{ context} \\ \exists, t: I | \Phi, \psi | \Gamma \vdash A \text{ type} \quad \exists, t: I | \Phi, \phi | \Gamma \vdash a: A \\ \hline \equiv, t: I | \Phi, \psi | \Gamma \vdash b: A \quad \exists, t: I | \Phi, \phi | \Gamma \vdash b \equiv a \\ \hline \equiv | \Phi | \Gamma \vdash \lambda t^{I | \psi} . b: \langle \prod_{t:I | \psi} A |_a^{\phi} \rangle \end{cases}$$

◆□ ▶ ◆□ ▶ ◆ □ ▶ ◆ □ ▶ ● □ ● ● ● ●





▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @