Two-level type theory in Lean

Semi-simplicial types in homotopy type theory

Adrien Mathieu

Thursday 12th, September

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of type. A type is a general form of collection.

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of type. A type is a general form of collection.

Set theory	Type theory

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of type. A type is a general form of collection.

Set theory	Type theory
$n \in \mathbb{N}$	$n : \mathbb{N}$

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of type. A type is a general form of collection.

Set theory	Type theory
$n \in \mathbb{N}$	$n : \mathbb{N}$
π is a proof of P	$\pi : P$

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of type. A type is a general form of collection.

Set theory	Type theory
$n \in \mathbb{N}$	$n : \mathbb{N}$
π is a proof of P	$\pi : P$
$\mathbb{N}$ is a set	$\mathbb{N}$: Type

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of type. A type is a general form of collection.

Set theory	Type theory
$n \in \mathbb{N}$	$n : \mathbb{N}$
π is a proof of P	$\pi : P$
$\mathbb{N}$ is a set	$\mathbb{N}$: Type
bijection	equivalence

Origins

Voevodsky introduced homotopy type theory to formalize "up to isomorphism" reasoning in proof assistants.

Origins

Voevodsky introduced homotopy type theory to formalize “up to isomorphism” reasoning in proof assistants.The goal is to work in mathematics in which “up to isomorphism” holds trivially, so no gory details should be provided.

Origins

Voevodsky introduced homotopy type theory to formalize “up to isomorphism” reasoning in proof assistants.The goal is to work in mathematics in which “up to isomorphism” holds trivially, so no gory details should be provided.
The solution is simple: *make the equality equivalent to the equivalences*!

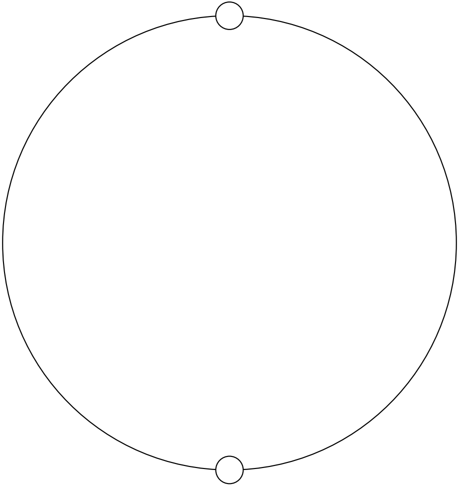
$$(A \simeq B) \simeq (A = B)$$

Equality in HoTT

A type should be thought of as a space, and two points of the space are considered equal if there is a path between them.

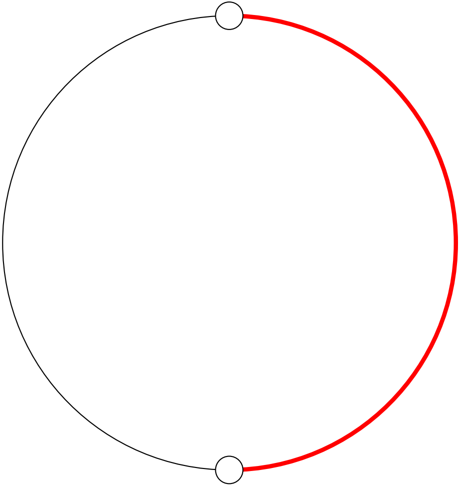
Equality in HoTT

A type should be thought of as a space, and two points of the space are considered equal if there is a path between them.



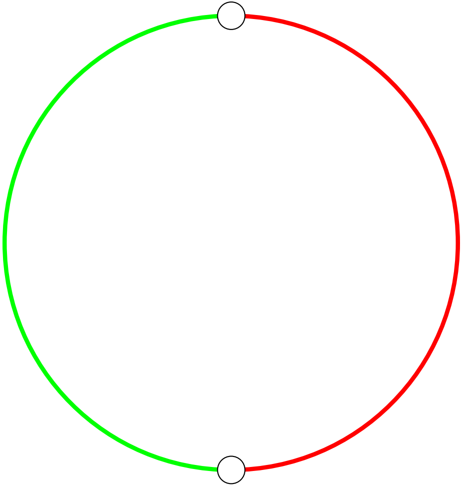
Equality in HoTT

A type should be thought of as a space, and two points of the space are considered equal if there is a path between them.



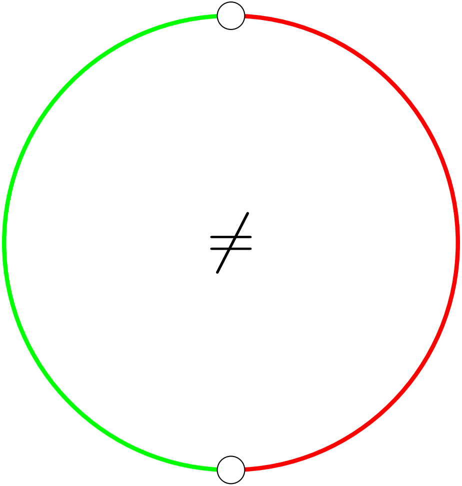
Equality in HoTT

A type should be thought of as a space, and two points of the space are considered equal if there is a path between them.



Equality in HoTT

A type should be thought of as a space, and two points of the space are considered equal if there is a path between them.



What about simplicial sets?

Simplicial sets are:

- basic tools;
- yet very useful in homotopy theory.

What about simplicial sets?

Simplicial sets are:

- basic tools;
- yet very useful in homotopy theory.

We would like to have an object that serves the same purpose in HoTT.

What about simplicial sets?

Simplicial sets are:

- basic tools;
- yet very useful in homotopy theory.

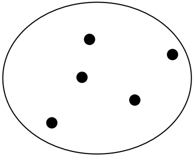
We would like to have an object that serves the same purpose in HoTT.
Let's start with semi-simplicial types.

What is a semi-simplicial type?

A semi-simplicial type is the data of

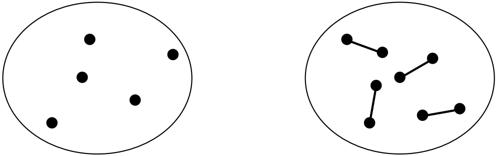
What is a semi-simplicial type?

A semi-simplicial type is the data of



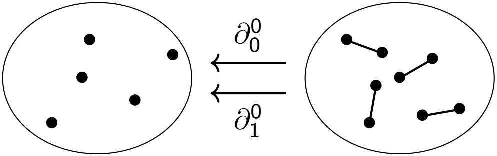
What is a semi-simplicial type?

A semi-simplicial type is the data of



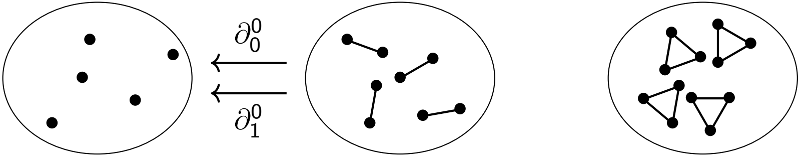
What is a semi-simplicial type?

A semi-simplicial type is the data of



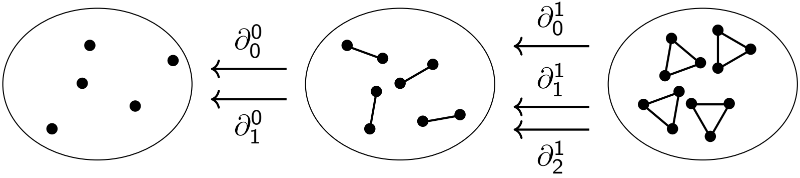
What is a semi-simplicial type?

A semi-simplicial type is the data of



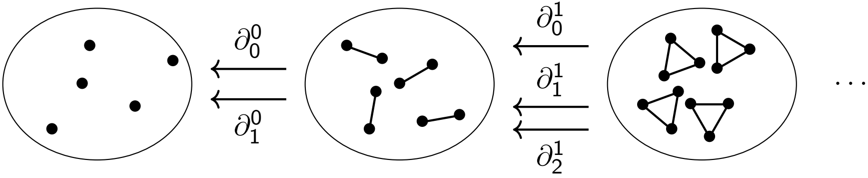
What is a semi-simplicial type?

A semi-simplicial type is the data of



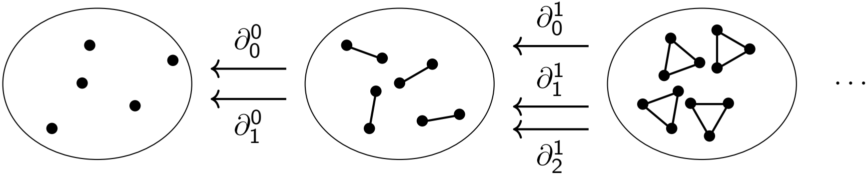
What is a semi-simplicial type?

A semi-simplicial type is the data of



What is a semi-simplicial type?

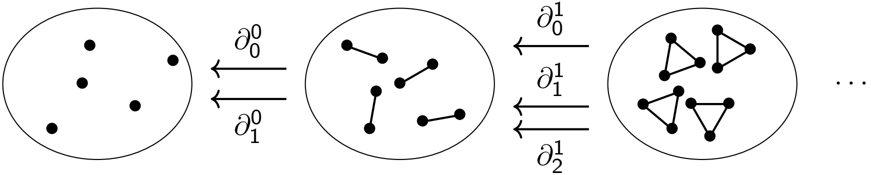
A semi-simplicial type is the data of



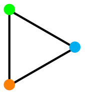
With an additional constraint:

What is a semi-simplicial type?

A semi-simplicial type is the data of

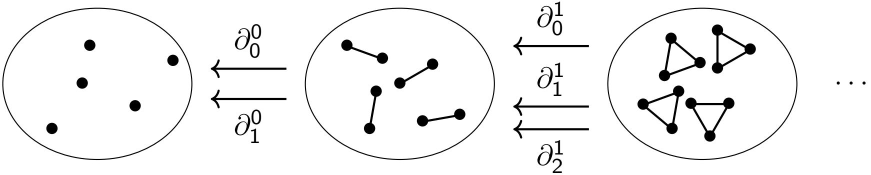


With an additional constraint:

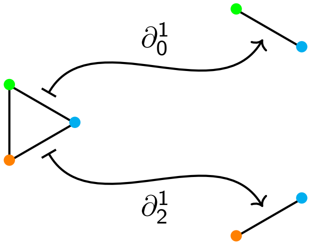


What is a semi-simplicial type?

A semi-simplicial type is the data of

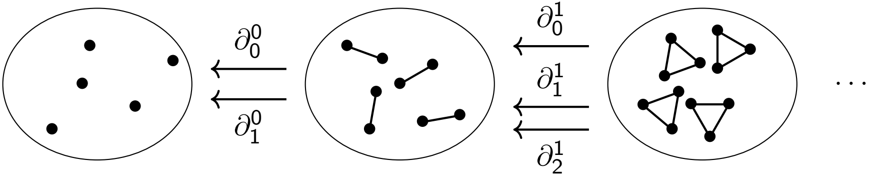


With an additional constraint:

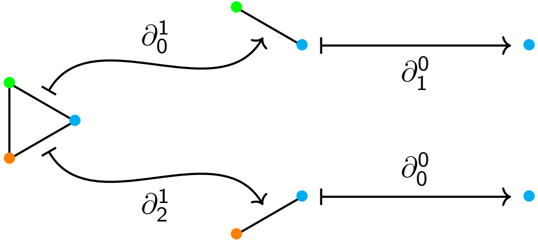


What is a semi-simplicial type?

A semi-simplicial type is the data of

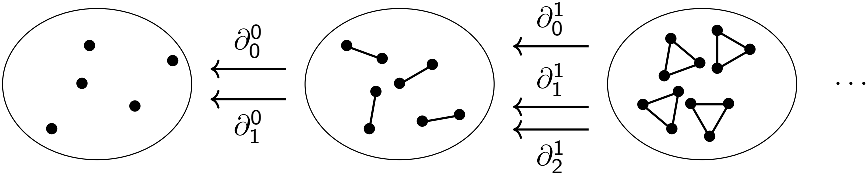


With an additional constraint:

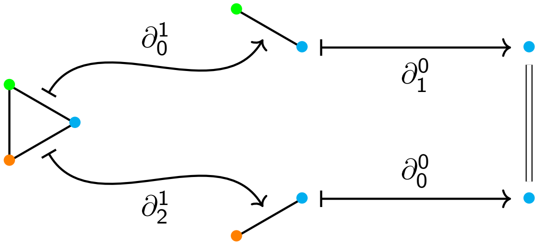


What is a semi-simplicial type?

A semi-simplicial type is the data of



With an additional constraint:



Formal specification

A semi-simplicial type is the data of

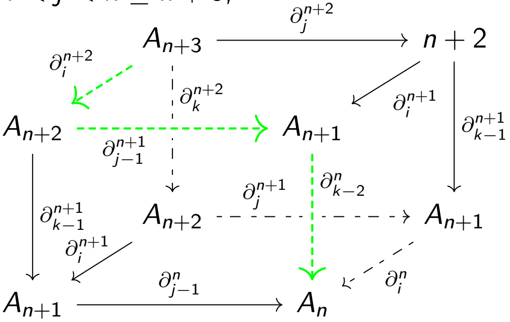
$$(A_n)_{n:\mathbb{N}} : \mathbb{N} \to \mathsf{Type}$$
$$\partial_i^n : A_{n+1} \to A_n \qquad \text{for } n : \mathbb{N} \text{ and } i \leq n+1$$

such that, for $n : \mathbb{N}$ and $i < j \leq n+1$,

$$\begin{array}{ccc} A_{n+2} & \xrightarrow{\partial_j^{n+1}} & A_{n+1} \\ \Big\downarrow{\scriptstyle \partial_i^{n+1}} & & \Big\downarrow{\scriptstyle \partial_i^{n}} \\ A_{n+1} & \xrightarrow{\partial_{j-1}^{n}} & A_n \end{array}$$

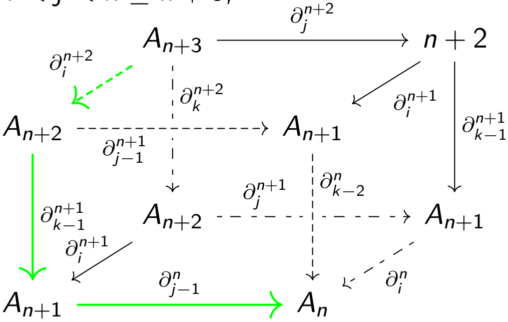
Higher problems

Given $n : \mathbb{N}$, and $i < j < k \leq n + 3$,



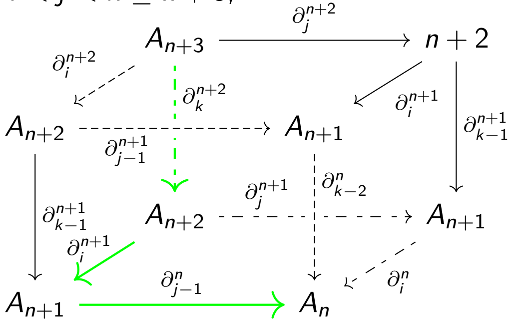
Higher problems

Given $n : \mathbb{N}$, and $i < j < k \leq n + 3$,



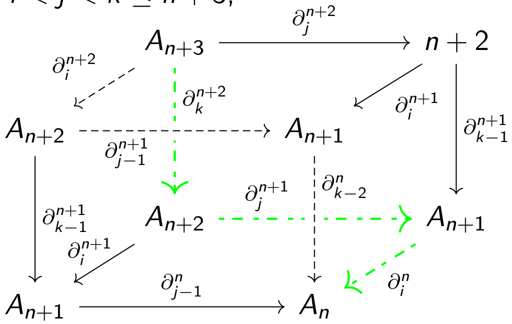
Higher problems

Given $n : \mathbb{N}$, and $i < j < k \leq n + 3$,



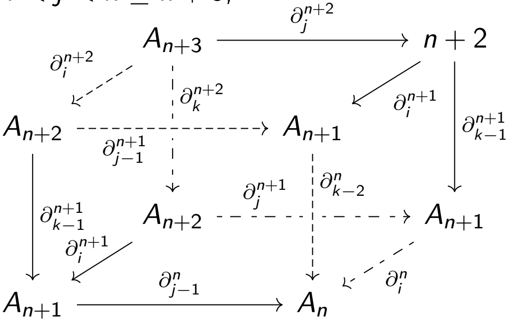
Higher problems

Given $n : \mathbb{N}$, and $i < j < k \leq n + 3$,



Higher problems

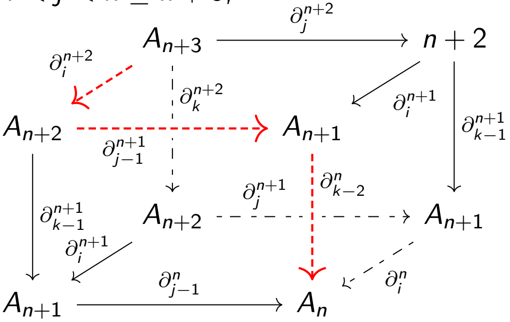
Given $n : \mathbb{N}$, and $i < j < k \leq n+3$,



$$p : \partial_{k-2}^{n} \circ \partial_{j-1}^{n+1} \circ \partial_i^{n+2} = \partial_i^{n} \circ \partial_j^{n+1} \circ \partial_k^{n+2}$$

Higher problems

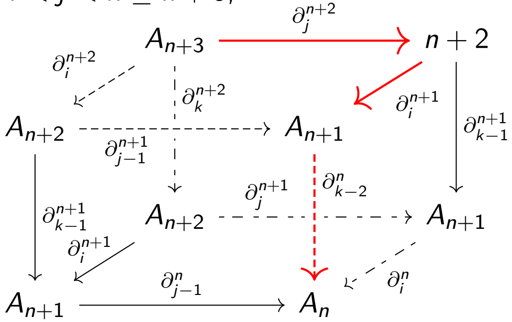
Given $n : \mathbb{N}$, and $i < j < k \leq n + 3$,



$$p : \partial_{k-2}^{n} \circ \partial_{j-1}^{n+1} \circ \partial_i^{n+2} = \partial_i^{n} \circ \partial_j^{n+1} \circ \partial_k^{n+2}$$

Higher problems

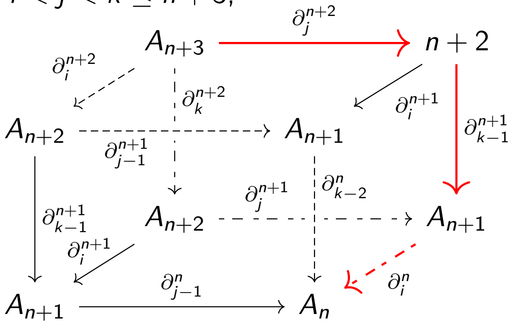
Given $n : \mathbb{N}$, and $i < j < k \leq n+3$,



$$p : \partial_{k-2}^{n} \circ \partial_{j-1}^{n+1} \circ \partial_i^{n+2} = \partial_i^{n} \circ \partial_j^{n+1} \circ \partial_k^{n+2}$$

Higher problems

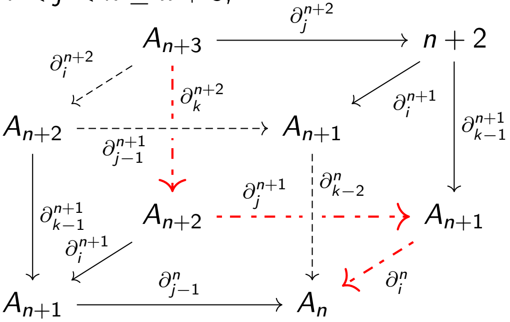
Given $n : \mathbb{N}$, and $i < j < k \leq n + 3$,



$$p : \partial_{k-2}^{n} \circ \partial_{j-1}^{n+1} \circ \partial_i^{n+2} = \partial_i^{n} \circ \partial_j^{n+1} \circ \partial_k^{n+2}$$

Higher problems

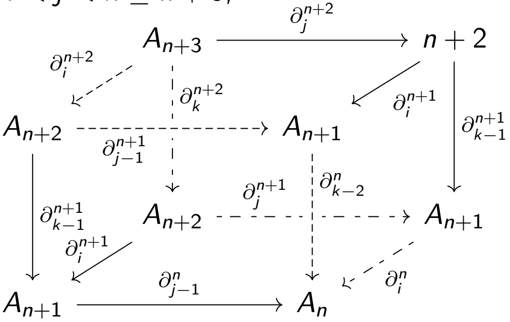
Given $n : \mathbb{N}$, and $i < j < k \leq n + 3$,



$$p : \partial_{k-2}^{n} \circ \partial_{j-1}^{n+1} \circ \partial_i^{n+2} = \partial_i^{n} \circ \partial_j^{n+1} \circ \partial_k^{n+2}$$

Higher problems

Given $n : \mathbb{N}$, and $i < j < k \leq n+3$,

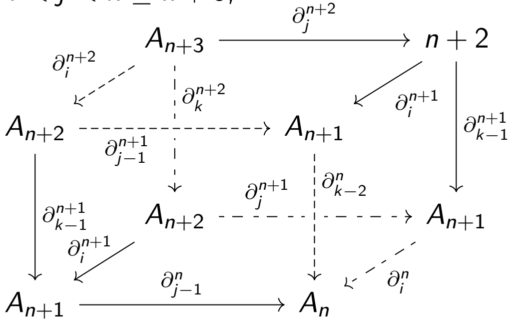


$$p : \partial^n_{k-2} \circ \partial^{n+1}_{j-1} \circ \partial^{n+2}_i = \partial^n_i \circ \partial^{n+1}_j \circ \partial^{n+2}_k$$

$$q : \partial^n_{k-2} \circ \partial^{n+1}_{j-1} \circ \partial^{n+2}_i = \partial^n_i \circ \partial^{n+1}_j \circ \partial^{n+2}_k$$

Higher problems

Given $n : \mathbb{N}$, and $i < j < k \leq n + 3$,



$$p : \partial_{k-2}^{n} \circ \partial_{j-1}^{n+1} \circ \partial_i^{n+2} = \partial_i^{n} \circ \partial_j^{n+1} \circ \partial_k^{n+2}$$

$$q : \partial_{k-2}^{n} \circ \partial_{j-1}^{n+1} \circ \partial_i^{n+2} = \partial_i^{n} \circ \partial_j^{n+1} \circ \partial_k^{n+2}$$

$$p \stackrel{?}{=} q$$

The impossible conjecture

So far, no one has managed to define semi-simplicial types in homotopy type theory.

The impossible conjecture

So far, no one has managed to define semi-simplicial types in homotopy type theory.
We would like to conjecture that it is impossible to define semi-simplicial types in homotopy type theory. How do you state this conjecture, in homotopy type theory?

The impossible conjecture

So far, no one has managed to define semi-simplicial types in homotopy type theory.
We would like to conjecture that it is impossible to define semi-simplicial types in homotopy type theory. How do you state this conjecture, in homotopy type theory?

It is impossible to define semi-simplicial types.

The impossible conjecture

So far, no one has managed to define semi-simplicial types in homotopy type theory.
We would like to conjecture that it is impossible to define semi-simplicial types in homotopy type theory. How do you state this conjecture, in homotopy type theory?

It is impossible to define semi-simplicial types.

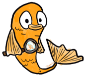
But this is self-contradictory!

The meta trick

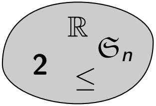
Let's step back.

The meta trick

Let's step back.

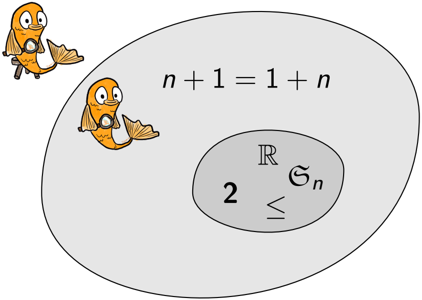


$n + 1 = 1 + n$



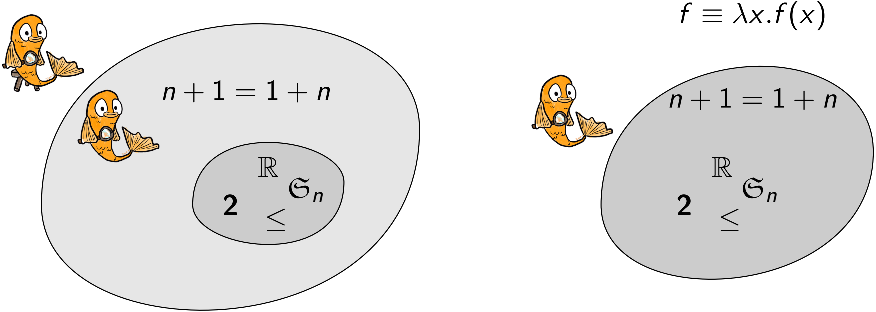
The meta trick

Let's step back.



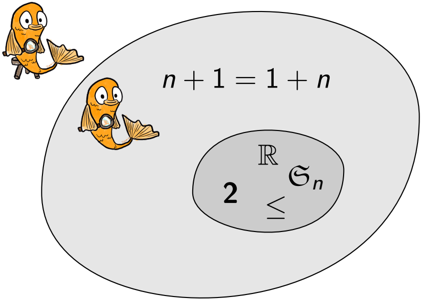
The meta trick

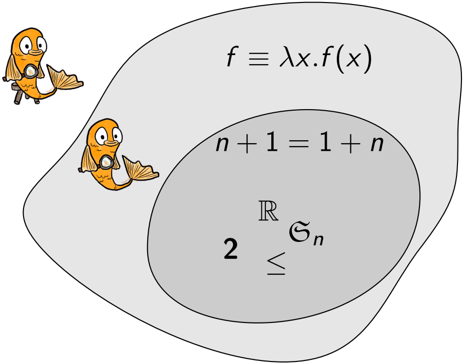
Let's step back.



The meta trick

Let's step back.



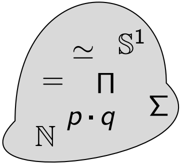


Lifting HoTT to an outer theory

The “trick” is to embed our meta tools used to speak of semi-simplicial types into the theory itself. This results into a strengthening of HoTT, called two-level type theory.

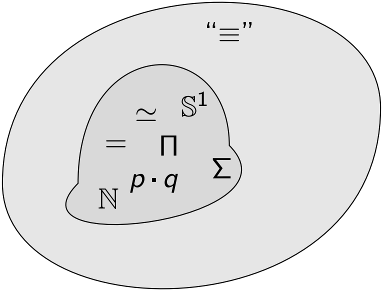
Lifting HoTT to an outer theory

The "trick" is to embed our meta tools used to speak of semi-simplicial types into the theory itself. This results into a strengthening of HoTT, called two-level type theory.



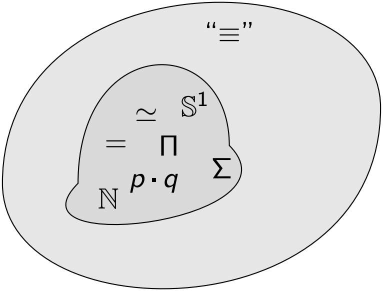
Lifting HoTT to an outer theory

The "trick" is to embed our meta tools used to speak of semi-simplicial types into the theory itself. This results into a strengthening of HoTT, called two-level type theory.



Lifting HoTT to an outer theory

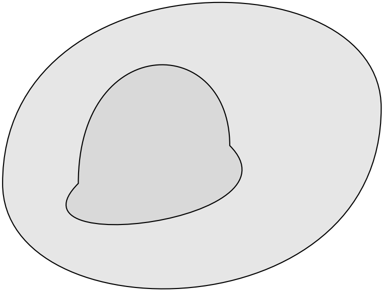
The "trick" is to embed our meta tools used to speak of semi-simplicial types into the theory itself. This results into a strengthening of HoTT, called two-level type theory.



Homotopy type theory can be lifted in the two-level type theory.

Stating the conjecture

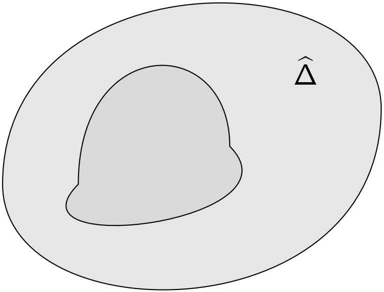
We now have a strategy to state the conjecture:



Stating the conjecture

We now have a strategy to state the conjecture:

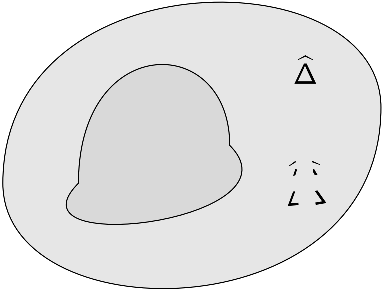
- define the type of semi-simplicial types in 2LTT



Stating the conjecture

We now have a strategy to state the conjecture:

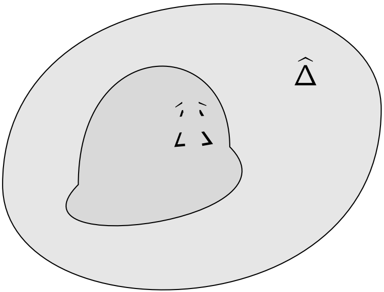
- define the type of semi-simplicial types in 2LTT
- break it into fragments



Stating the conjecture

We now have a strategy to state the conjecture:

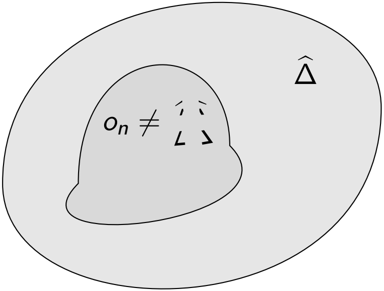
- define the type of semi-simplicial types in 2LTT
- break it into fragments
- push them into HoTT



Stating the conjecture

We now have a strategy to state the conjecture:

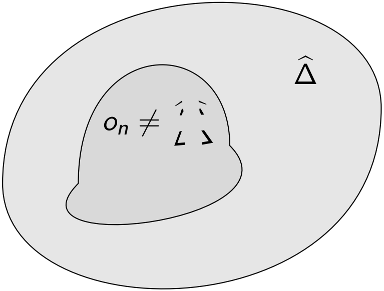
- define the type of semi-simplicial types in 2LTT
- break it into fragments
- push them into HoTT
- state "there is no collection of objects in HoTT that is pointwise equal to the collection of fragments"



Stating the conjecture

We now have a strategy to state the conjecture:

- define the type of semi-simplicial types in 2LTT
- break it into fragments
- push them into HoTT
- state “there is no collection of objects in HoTT that is pointwise equal to the collection of fragments”



If there was such a think such as semi-simplicial types directly in HoTT, we could also break it

The Reedy way (1/2)

We now have to break semi-simplicial types into fragments that fit into HoTT. To do so, we present semi-simplicial types in the Reedy fashion.

The Reedy way (1/2)

We now have to break semi-simplicial types into fragments that fit into HoTT. To do so, we present semi-simplicial types in the Reedy fashion.

A_0 : Type

The Reedy way (1/2)

We now have to break semi-simplicial types into fragments that fit into HoTT. To do so, we present semi-simplicial types in the Reedy fashion.

$$\left[\begin{array}{l} A_0 : \text{Type} \\ A_1 : A_0 \to A_0 \to \text{Type} \end{array}\right.$$

The Reedy way (1/2)

We now have to break semi-simplicial types into fragments that fit into HoTT. To do so, we present semi-simplicial types in the Reedy fashion.

$$
\begin{aligned}
&A_0 : \mathsf{Type} \\
&A_1 : A_0 \to A_0 \to \mathsf{Type} \\
&A_2 : (x : A_0) \to (y : A_0) \to (z : A_0) \\
&\qquad\qquad \to A_1(x, y) \to A_1(x, z) \to A_1(y, z) \to \mathsf{Type}
\end{aligned}
$$

The Reedy way (1/2)

We now have to break semi-simplicial types into fragments that fit into HoTT. To do so, we present semi-simplicial types in the Reedy fashion.

$$
\begin{aligned}
&A_0 : \mathrm{Type}\\
&A_1 : A_0 \to A_0 \to \mathrm{Type}\\
&A_2 : (x : A_0) \to (y : A_0) \to (z : A_0)\\
&\qquad\quad \to A_1(x, y) \to A_1(x, z) \to A_1(y, z) \to \mathrm{Type}\\
&A_3 : (x, y, z, w : A_0) \to (s_1 : A_1(x, y)) \to (s_2 : A_1(x, z)) \to (s_3 : A_1(x, w))\\
&\qquad\quad \to (s_4 : A_1(y, z)) \to (s_5 : A_1(y, w)) \to (s_6 : A_1(z, w))\\
&\qquad\quad \to A_2(x, y, z, s_1, s_2, s_4) \to A_2(x, y, w, s_1, s_3, s_5)\\
&\qquad\quad \to A_2(x, z, w, s_2, s_3, s_6) \to A_2(y, z, w, s_4, s_5, s_6)\\
&\qquad\quad \to \mathrm{Type}
\end{aligned}
$$

The Reedy way (1/2)

We now have to break semi-simplicial types into fragments that fit into HoTT. To do so, we present semi-simplicial types in the Reedy fashion.

$$
\left[
\begin{array}{l}
A_0 : \mathrm{Type} \\
A_1 : A_0 \to A_0 \to \mathrm{Type} \\
A_2 : (x : A_0) \to (y : A_0) \to (z : A_0) \\
\qquad\quad \to A_1(x, y) \to A_1(x, z) \to A_1(y, z) \to \mathrm{Type} \\
A_3 : (x, y, z, w : A_0) \to (s_1 : A_1(x, y)) \to (s_2 : A_1(x, z)) \to (s_3 : A_1(x, w)) \\
\qquad\quad \to (s_4 : A_1(y, z)) \to (s_5 : A_1(y, w)) \to (s_6 : A_1(z, w)) \\
\qquad\quad \to A_2(x, y, z, s_1, s_2, s_4) \to A_2(x, y, w, s_1, s_3, s_5) \\
\qquad\quad \to A_2(x, z, w, s_2, s_3, s_6) \to A_2(y, z, w, s_4, s_5, s_6) \\
\qquad\quad \to \mathrm{Type} \\
\qquad \vdots
\end{array}
\right.
$$

The Reedy way (2/2)

These objects are called *very dependent types*. It is unknown whether they can be formulated directly in HoTT, and even whether they are consistent in it!

The Reedy way (2/2)

These objects are called *very dependent types*. It is unknown whether they can be formulated directly in HoTT, and even whether they are consistent in it!
But we can generate the signature of each level using the meta language.
The fragments of the semi-simplicial types are simply the truncation up to A_n, for every $n : \mathbb{N}$.

The Reedy way (2/2)

These objects are called *very dependent types*. It is unknown whether they can be formulated directly in HoTT, and even whether they are consistent in it!
But we can generate the signature of each level using the meta language.
The fragments of the semi-simplicial types are simply the truncation up to A_n, for every $n : \mathbb{N}$.
The bulk of the work is showing that these fragments "fit" in HoTT.