

Constructing the Propositional Truncation using Non-recursive HITs

Floris van Doorn

Carnegie Mellon University

January 19, 2016

In *Homotopy Type Theory* (HoTT) there are *Higher Inductive Types*, generalizing Inductive Types.

Goal: Reduce complicated Higher Inductive Types to simpler ones.

Analogue: In Extensional Type Theory, we can reduce all inductive types to W -types and Σ -types.

Homotopy Type Theory

Homotopy Type Theory combines Type Theory with Homotopy Theory.

	Type Theory	Logic
A	Type	Formula
$a : A$	Term	Proof
$A + B$	Sum Type	Disjunction
$A \rightarrow B$	Function Type	Implication
$P : A \rightarrow \text{Type}$	Dependent Type	Predicate
$\prod(x : A), P(x)$	Dep. Fn. Type	U. Quantifier
$a =_A b$	Identity	Equality

Homotopy Type Theory

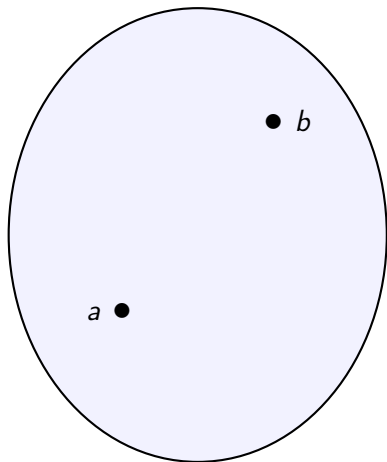
Homotopy Type Theory combines Type Theory with Homotopy Theory.

	Type Theory	Logic	Homotopy Theory
A	Type	Formula	Space*
$a : A$	Term	Proof	Point
$A + B$	Sum Type	Disjunction	Coproduct of spaces
$A \rightarrow B$	Function Type	Implication	Mapping space
$P : A \rightarrow \text{Type}$	Dependent Type	Predicate	Fibration
$\prod(x : A), P(x)$	Dep. Fn. Type	U. Quantifier	Dep. product space
$a =_A b$	Identity	Equality	Path space

Types as spaces

A type A can have

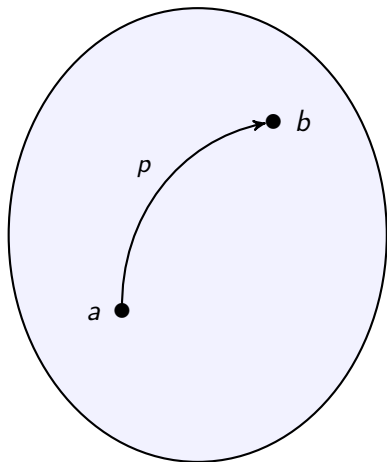
• points $a, b : A$



Types as spaces

A type A can have

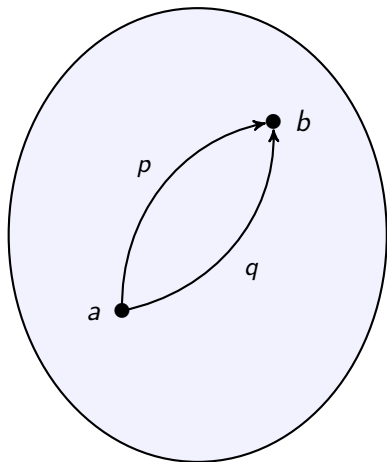
- paths $p : a =_A b$
- points $a, b : A$



Types as spaces

A type A can have

- paths $q, p : a =_A b$
- points $a, b : A$

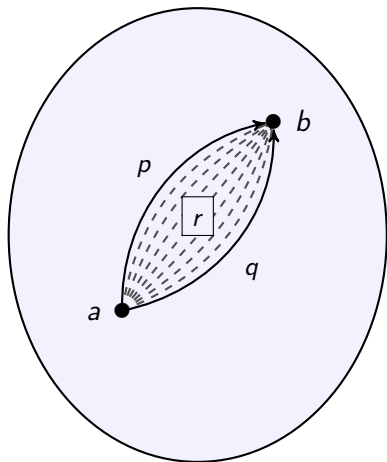


Types as spaces

A type A can have

⋮

- paths between paths $r : p = q$
- paths $q, p : a =_A b$
- points $a, b : A$

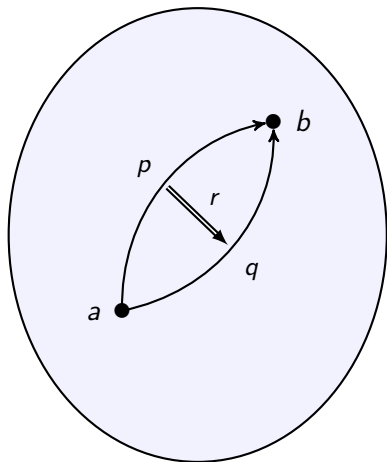


Types as spaces

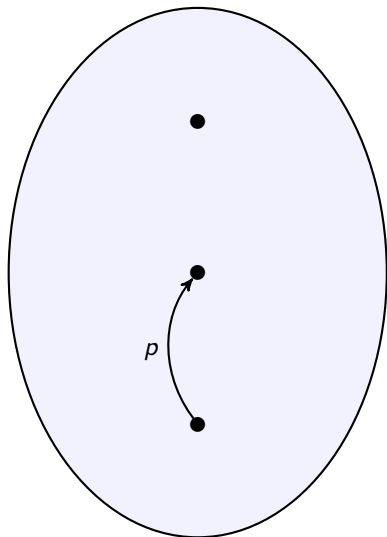
A type A can have

:

- paths between paths $r : p = q$
- paths $q, p : a =_A b$
- points $a, b : A$



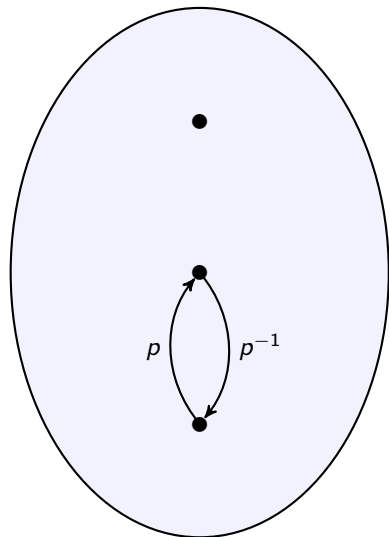
Operations on paths



Operations on paths

We can

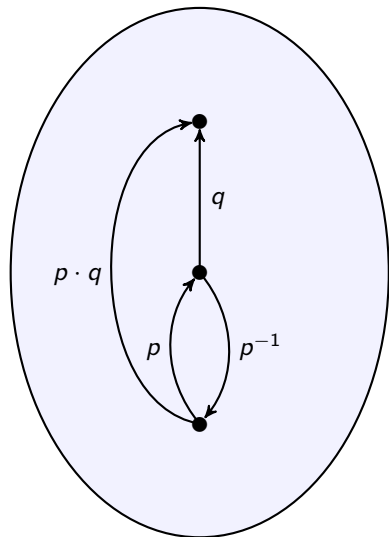
- invert paths (symmetry);



Operations on paths

We can

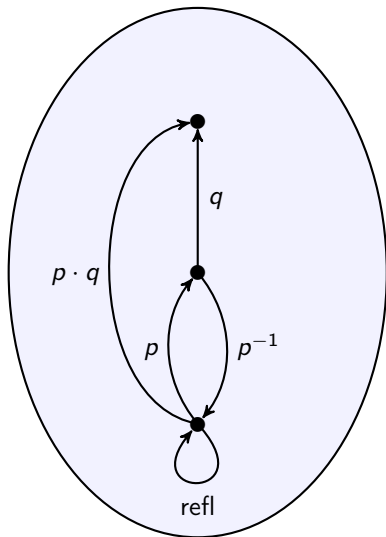
- invert paths (symmetry);
- concatenate paths (transitivity);



Operations on paths

We can

- invert paths (symmetry);
- concatenate paths (transitivity);
- make identity paths (reflexivity).
(and more)



Some types have trivial higher structure.

Type

⋮

- paths between paths $r, s : p = q$
- paths $q, p : a =_A b$
- points $a, b : A$

Some types have trivial higher structure.

Set

⋮

- [at most one path between paths]
- [at most one path between points]
- points $a, b : A$

Type

⋮

- paths between paths $r, s : p = q$
- paths $q, p : a =_A b$
- points $a, b : A$

Some types have trivial higher structure.

Mere Proposition

⋮

- [at most one path between paths]
- [at most one path between points]
- [at most one point]

Set

⋮

- [at most one path between paths]
- [at most one path between points]
- points $a, b : A$

Type

⋮

- paths between paths $r, s : p = q$
- paths $q, p : a =_A b$
- points $a, b : A$

Propositional Truncation

Given A , we can form the *Propositional truncation* $\|A\|$.

$\|A\|$ is the **mere proposition** specifying whether A is inhabited.

Propositional Truncation

Given A , we can form the *Propositional truncation* $\|A\|$.

$\|A\|$ is the **mere proposition** specifying whether A is inhabited.

A
 \vdots

- paths between paths
- paths
- points

\Rightarrow

$\|A\|$
 \vdots

- [at most one path between paths]
- [at most one path between points]
- [at most one point]

Propositional Truncation (usage)

An element of $A + B$ specifies whether A is inhabited or B is inhabited.

Using the propositional truncation we can define a proof irrelevant disjunction which does not reveal its witness.

$$A \vee B := \parallel A + B \parallel$$

$$\exists x, P(x) := \parallel \Sigma x, P(x) \parallel$$

Higher Inductive Types

In HoTT we use *Higher Inductive Types* to construct types with nontrivial paths.

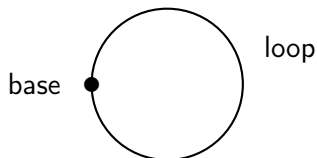
Higher Inductive Types

In HoTT we use *Higher Inductive Types* to construct types with nontrivial paths.

Example: the circle

HIT $S^1 :=$

- $\text{base} : S^1$
- $\text{loop} : \text{base} = \text{base}$



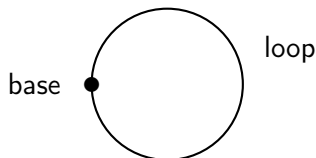
Higher Inductive Types

In HoTT we use *Higher Inductive Types* to construct types with nontrivial paths.

Example: the circle

HIT $S^1 :=$

- $\text{base} : S^1$
- $\text{loop} : \text{base} = \text{base}$



In the circle we have $\text{loop} \neq \text{refl}$. Hence we have paths

$\dots \text{loop}^{-1}, \text{refl}, \text{loop}, \text{loop} \cdot \text{loop}, \text{loop} \cdot \text{loop} \cdot \text{loop}, \dots$

which are all different.

Sequential Colimits

Given

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} A_3 \xrightarrow{f_3} \dots$$

we can define the colimit of this sequence.

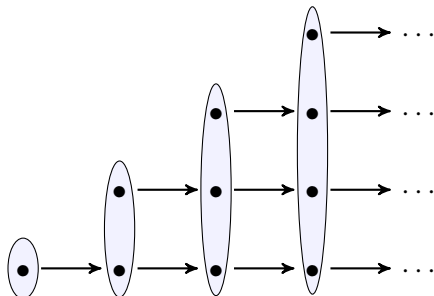
Sequential Colimits

Given

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} A_3 \xrightarrow{f_3} \dots$$

we can define the colimit of this sequence.

⋮



The colimit of this diagram is \mathbb{N} .

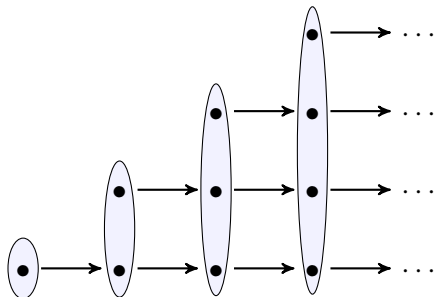
Sequential Colimits

Given

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} A_3 \xrightarrow{f_3} \dots$$

we can define the colimit of this sequence.

⋮



This colimit is a HIT:

HIT $\text{colimit}(A, f) :=$

- $i : \prod(n : \mathbb{N}), A_n \rightarrow \text{colimit}(A, f)$
- $g : \prod(n : \mathbb{N}), \prod(a : A_n), i_{n+1}(f_n(a)) = i_n(a)$

The colimit of this diagram is \mathbb{N} .

Propositional Truncation as HIT

The Propositional Truncation $\|-\|$ is also a HIT:

HIT $\|A\| :=$

- $|-| : A \rightarrow \|A\|$
- $\varepsilon : \Pi(x, y : \|A\|), x = y$

Propositional Truncation as HIT

The Propositional Truncation $\|-\|$ is also a HIT:

HIT $\|A\| :=$

- $|-| : A \rightarrow \|A\|$
- $\varepsilon : \Pi(x, y : \|A\|), x = y$

ε is a *recursive path constructor*: the domain of the recursor is the type $\|A\|$ being constructed.

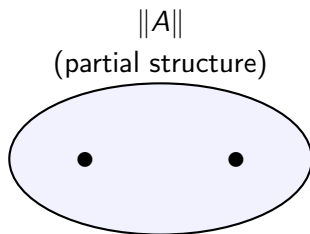
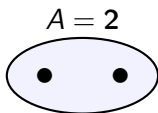
Propositional Truncation as HIT

The Propositional Truncation $\|- \|$ is also a HIT:

HIT $\|A\| :=$

- $|-| : A \rightarrow \|A\|$
- $\varepsilon : \Pi(x, y : \|A\|), x = y$

ε is a *recursive path constructor*: the domain of the recursor is the type $\|A\|$ being constructed.



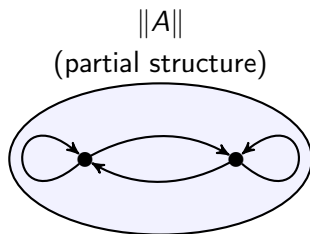
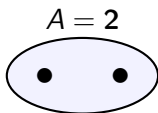
Propositional Truncation as HIT

The Propositional Truncation $\|-\|$ is also a HIT:

HIT $\|A\| :=$

- $|-| : A \rightarrow \|A\|$
- $\varepsilon : \prod(x, y : \|A\|), x = y$

ε is a *recursive path constructor*: the domain of the recursor is the type $\|A\|$ being constructed.



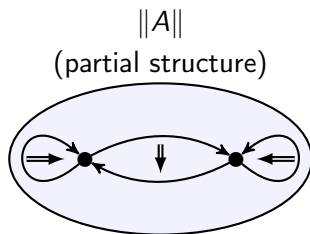
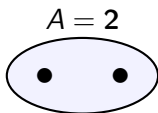
Propositional Truncation as HIT

The Propositional Truncation $\|- \|$ is also a HIT:

HIT $\|A\| :=$

- $|-| : A \rightarrow \|A\|$
- $\varepsilon : \prod(x, y : \|A\|), x = y$

ε is a *recursive path constructor*: the domain of the recursor is the type $\|A\|$ being constructed.



HITs are not well understood

- There is no general theory describing which HITs are allowed.
- Which restrictions are needed for the constructors?
- It will help if we can reduce complicated HITs to basic HITs.
- The definition of the propositional truncation is impredicative. Can we give a predicative construction?

Construction of the Propositional truncation

We will define the Propositional Truncation as a colimit.

At every step we will apply the *one-step truncation*, which is the following HIT.

HIT $\{A\} :=$

- $f : A \rightarrow \{A\}$
- $e : \Pi(x, y : A), f(x) = f(y)$

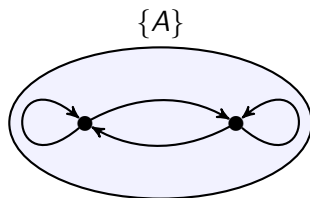
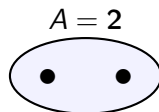
Construction of the Propositional truncation

We will define the Propositional Truncation as a colimit.

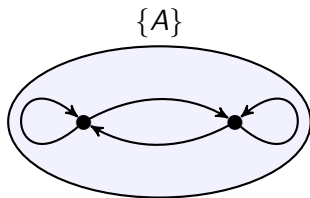
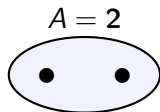
At every step we will apply the *one-step truncation*, which is the following HIT.

HIT $\{A\} :=$

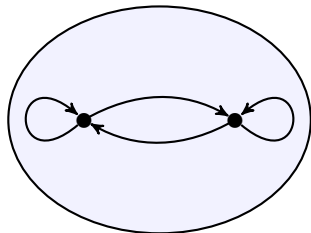
- $f : A \rightarrow \{A\}$
- $e : \Pi(x, y : A), f(x) = f(y)$



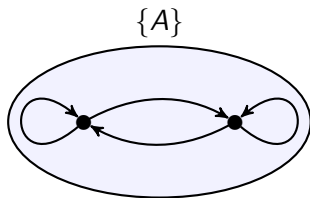
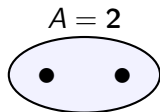
Construction of the Propositional truncation



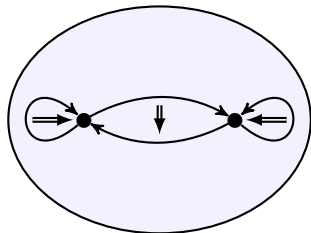
$\{\{A\}\}$
(partial structure)



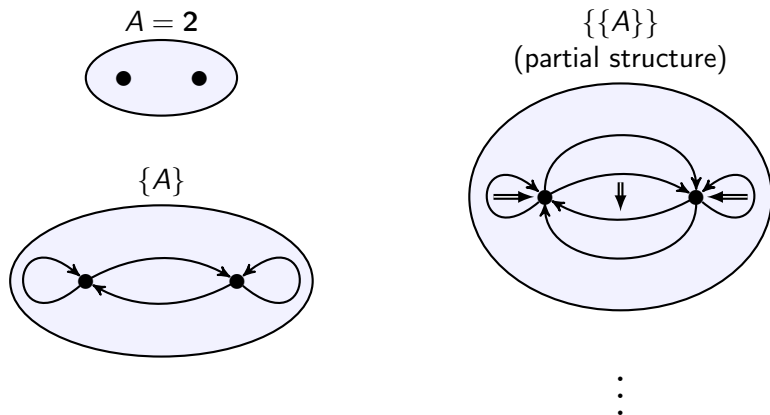
Construction of the Propositional truncation



$\{\{A\}\}$
(partial structure)



Construction of the Propositional truncation



Construction of the Propositional truncation

We obtain the diagram

$$A \xrightarrow{f} \{A\} \xrightarrow{f} \{\{A\}\} \xrightarrow{f} \{\{\{A\}\}\} \xrightarrow{f} \dots \quad (1)$$

Theorem

The colimit of diagram (1) is the propositional truncation $\|A\|$.

Corollary

A function in $\|A\| \rightarrow B$ is the same as a cocone over (1), for any type B .

This result is formally proven in the proof assistant Lean.

In Lean, the HoTT mode is (mostly) just the standard mode, without Prop.

There is no proof assistant with good support for HITs.

In Lean we added two HITs as a primitive types, and we can define most of the commonly used HITs in terms of these.

```
definition trunc.{u} (A : Type.{u}) : Type.{u}
definition tr (A : Type) : A → trunc A
definition is_hprop_trunc (A : Type) : is_hprop (trunc A)
definition trunc.rec {A : Type} {P : trunc A → Type}
  [Pt :  $\prod$ (x : trunc A), is_hprop (P x)]
  (H :  $\prod$ (a : A), P (tr a)) :  $\prod$ (x : trunc A), P x

definition elim2_equiv (A P : Type) : (trunc A → P)  $\simeq$ 
   $\sum$ (h :  $\prod$ {n}, n_step_tr A n → P),
   $\prod$ (n :  $\mathbb{N}$ ) (a : n_step_tr A n),
  @h (succ n) (one_step_tr.tr a) = h a
```

- HITs are generally not very well understood.
- We can construct the propositional truncation (a recursive HIT) using simpler (nonrecursive) HITs.
- Conjecture: A large class of HITs can be reduced to a single HIT, the *homotopy coequalizer*.

Thank you

The Lean formalization is available at:

<https://github.com/fpvandoorn/leansnippets/blob/master/cpp.hlean>