On the Formalization of Higher Inductive Types and Synthetic Homotopy Theory

Floris van Doorn

Carnegie Mellon University

May 12, 2017

Homotopy Type Theory

Homotopy type theory (HoTT) refers to the homotopical interpretation of dependent type theory by Awodey, Warren and Voevodsky.

In HoTT we can do *Synthetic Homotopy Theory*:

Study types in type theory as spaces in homotopy theory.

Advantages over regular homotopy theory:

- More general
- Constructive
- Feasible to formalize
- Novel ways of reasoning

Homotopy Type Theory

Homotopy Type Theory combines Type Theory with Homotopy Theory.

	Type Theory	**Logic**	**Homotopy Theory**
A	Type	Formula	Space*
$a : A$	Term/Element	Proof	Point
$A \times B$	Product Type	Conjunction	Binary Product sp.
$A \to B$	Function Type	Implication	Mapping space
$P : A \to \mathcal{U}$	Dependent Type	Predicate	Fibration
$(x : A) \times P(x)$	Sigma Type	Ex. Quantifier	Total space
$(x : A) \to P(x)$	Dep. Fn. Type	Un. Quantifier	Product space
$a =_A b$	Identity Type	Equality	Path space

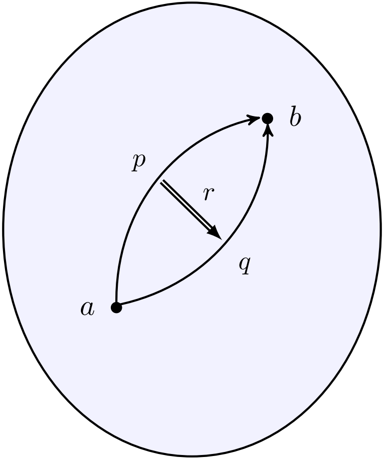
I will use these notions interchangably.

Types as spaces

A type A can have

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

$$\vdots$$



Identity Type

Different ways to think about the identity type:

- **Type theory**: The identity type $a =_A (-)$ is generated by $\mathrm{refl}_a : a =_A a$.
- **Logic**: Equality is the least (free) reflexive relation.
- **Homotopy theory**: The path space with one point fixed is contractible.

Formalization in Lean

It is feasible to formalize Homotopy Type Theory in a proof assistant.

To show this, we are formalizing all obtained results in *Lean*.

Lean is a new open source proof assistant with support for HoTT, similar to Coq and Agda.

Lean implements dependent type theory with a hierarchy of (non-cumulative) universes and inductive types (à la Dybjer).

Lean has a small kernel, smaller than Coq or Agda.

Lean has a HoTT mode to do Homotopy Type Theory.

Formalization in Lean

The HoTT library (including spectral sequence project) has ∼40k lines of code, compared to ∼30k in HoTT-Coq and ∼20k in HoTT-Agda.

It contains:

- Most of the results in chapter 1-8 from the HoTT-book

```
definition whitehead_principle (n : ℕ₋₂) {A B : Type}
  [HA : is_trunc n A] [HB : is_trunc n B] (f : A → B)
  (H₁ : is_equiv (trunc_functor 0 f))
  (H₂ : Πa k, is_equiv
              (π→*[k + 1] (pmap_of_map f a))) :
  is_equiv f
```

Formalization in Lean

The HoTT library (including spectral sequence project) has ∼40k lines of code, compared to ∼30k in HoTT-Coq and ∼20k in HoTT-Agda.

It contains:

- Most of the results in chapter 1-8 from the HoTT-book
- A library of squares, cubes, pathovers, squareovers, cubeovers (based on [Licata-Brunerie, 2015])

```
definition circle.rec {P : S¹ → Type}
  (Pbase : P base) (Ploop : Pbase =[loop] Pbase)
  (x : S¹) : P x
```

Formalization in Lean

The HoTT library (including spectral sequence project) has ∼40k lines of code, compared to ∼30k in HoTT-Coq and ∼20k in HoTT-Agda.

It contains:

- Most of the results in chapter 1-8 from the HoTT-book
- A library of squares, cubes, pathovers, squareovers, cubeovers (based on [Licata-Brunerie, 2015])
- A library of pointed types, pointed maps, pointed homotopies, pointed equivalences

```
definition loopn_ptrunc_pequiv
  (n : ℕ₋₂) (k : ℕ) (A : Type*) :
  Ω[k] (ptrunc (n+k) A) ≃* ptrunc n (Ω[k] A)
```

Formalization in Lean

The HoTT library (including spectral sequence project) has ∼40k lines of code, compared to ∼30k in HoTT-Coq and ∼20k in HoTT-Agda.

It contains:

- Most of the results in chapter 1-8 from the HoTT-book
- A library of squares, cubes, pathovers, squareovers, cubeovers (based on [Licata-Brunerie, 2015])
- A library of pointed types, pointed maps, pointed homotopies, pointed equivalences
- Category theory, with e.g. limits, yoneda lemma and exponential laws:

```
definition functor_functor_iso (C D E : Precategory) :
  (C ^c D) ^c E ≅c C ^c (E ×c D)
```

Formalization in Lean

The HoTT library (including spectral sequence project) has ∼40k lines of code, compared to ∼30k in HoTT-Coq and ∼20k in HoTT-Agda.

It contains:

- Most of the results in chapter 1-8 from the HoTT-book
- A library of squares, cubes, pathovers, squareovers, cubeovers (based on [Licata-Brunerie, 2015])
- A library of pointed types, pointed maps, pointed homotopies, pointed equivalences
- Category theory, with e.g. limits, yoneda lemma and exponential laws:
- Various results in synthetic homotopy theory

```
variables (G : AbGroup) (X : Type) (n : ℕ)
definition K_pequiv (e : πg[n+1] X ≃g G)
  [H1 : is_conn n X] [H2 : is_trunc (n.+1) X]
  : K G (n+1) ≃* X
```

Why formalization?

1. To establish that a result is correct

Formalization in Lean

Why formalization?

1. To establish that a result is correct
2. To show that formalization of more complicated results is feasible

Formalization in Lean

Why formalization?

1. To establish that a result is correct
2. To show that formalization of more complicated results is feasible
3. In principle it is possible to give a formalized algorithm to compute algebraic invariants of spaces

Formalization in Lean

Why formalization?

1. To establish that a result is correct
2. To show that formalization of more complicated results is feasible
3. In principle it is possible to give a formalized algorithm to compute algebraic invariants of spaces
4. To force the use of convenient definitions and reusable theorems

Outline

1. Introduction
2. Higher Inductive Types
    1. Constructing the propositional truncation
    2. Constructing the localization
3. Synthetic Homotopy Theory
    1. Long exact sequence of homotopy groups
    2. The Serre spectral sequence

Not in talk

Constructing 2-HITs
Eilenberg-MacLane spaces
The smash product

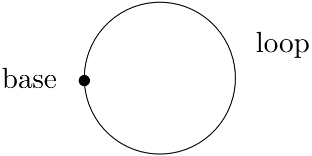
Higher Inductive Types

In HoTT there are *higher inductive types*, which combine inductive types from type theory with cell complexes from homotopy theory [Shulman-Lumsdaine, 2012].

Example. The circle $\mathbb{S}^1$

`HIT` $\mathbb{S}^1 :=$

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



Using univalence, we can prove $\mathrm{loop} \neq \mathrm{refl}$.

Quotients

The *quotient* (or graph quotient or typal quotient) is the following HIT given $A : \mathcal{U}$, $R : A \to A \to \mathcal{U}$.

`HIT` $\mathsf{quotient}_A(R) :=$

- $i : A \to \mathsf{quotient}_A(R)$
- $\mathsf{glue} : (a\ a' : A) \to R(a, a') \to i(a) = i(a')$

Question. Which HITs can we define in using quotients (in MLTT + UA)?

Quotients

Question. Which HITs can we define using quotients?
(in MLTT + UA)?

It is easy to define nonrecursive 1-HITs: pushouts, suspensions, ...

We can also construct nonrecursive 2-HITs: torus, groupoid quotient, $K(G,1)$, ...

Progress on recursive HITs:

- Formalized construction of the propositional truncation [v.D., 2016]
- Construction of the n-truncation [Egbert Rijke, 2017]
- Sketch of construction of ω-compact localizations (j.w.w. Egbert Rijke, Kristina Sojakova)

Construction of the Propositional truncation

We want to define the propositional truncation using quotients.

$\texttt{HIT}\ \|A\| :=$

- $|-| : A \to \|A\|$
- $(x\ y : \|A\|) \to x = y$

We will define the Propositional Truncation as a colimit.

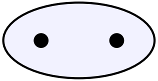
At every step we will apply the *one-step truncation*, which is the following HIT.

$\texttt{HIT}\ \{A\} :=$

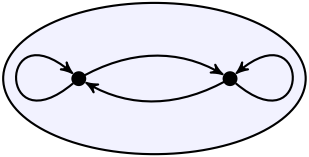
- $f : A \to \{A\}$
- $e : (x\ y : A) \to f(x) = f(y)$

Construction of the Propositional truncation

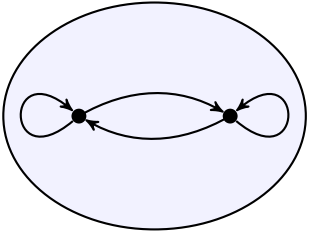
$A = \mathbf{2}$



$\{A\}$

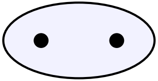


$\{\{A\}\}$
(partial structure)

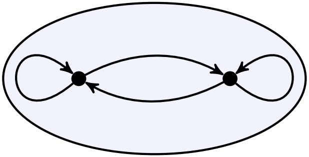


Construction of the Propositional truncation

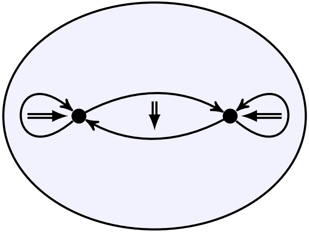
$A = \mathbf{2}$



$\{A\}$

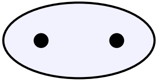


$\{\{A\}\}$
(partial structure)

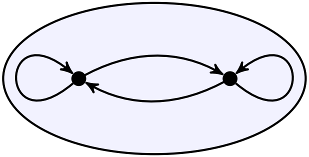


Construction of the Propositional truncation

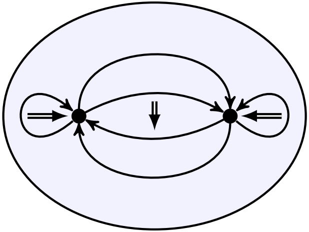
$A = \mathbf{2}$



$\{A\}$



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



$\vdots$

Construction of the Propositional truncation

We obtain the diagram

$$A \xrightarrow{f} \{A\} \xrightarrow{f} \{\{A\}\} \xrightarrow{f} \{\{\{A\}\}\} \xrightarrow{f} \cdots \tag{1}$$

Theorem

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

Construction of the Propositional truncation

```
parameter (A : Type)
definition An : ℕ → Type
| An 0          := A
| An (succ n) := one_step_tr (An n)
definition fn {n : ℕ} : An n → An (succ n) := f
definition truncA : Type := @seq_colim An fn

theorem is_prop_truncA : is_prop truncA
definition truncA.rec {P : truncA → Type}
  [Π(x : truncA), is_prop (P x)] (H : Π(a : A), P (i a)) :
  Π(x : truncA), P x
example : truncA.rec H (i a) = H a := by reflexivity
```

Local types

Given families $P, Q : A \to \mathcal{U}$ and $F : (a : A) \to P(a) \to Q(a)$.

A type X is F-*local* if for all $a : A$ the map

$$\psi_X(a) :\equiv \lambda f. f \circ F(a) : (Q(a) \to X) \to (P(a) \to X)$$

is an equivalence.

$$\begin{tikzcd} P(a) \arrow[d] \arrow[dr] & \\ Q(a) \arrow[r, dashed] & X \end{tikzcd}$$

Example A is n-truncated iff A is local w.r.t. $\mathbb{S}^{n+1} \to 1$.

Localizations

The F-*localization* $L_F X$ of X turns X into a F-local type in a universal way.

$$\begin{tikzcd}
P(a) \arrow[d] \arrow[dr] & X \arrow[d] \\
Q(a) \arrow[r, dashed] & L_F X
\end{tikzcd}$$

`HIT` $L_F X :=$

- $i : X \to L_F X$
- $r : \{a\} \to (P(a) \to L_F X) \to Q(a) \to L_F X$
- $\ell : \{a\} \to (P(a) \to L_F X) \to Q(a) \to L_F X$
- $\{a\} \to (f : P(a) \to L_F X) \to (x : P(a)) \to r_f(F(x)) = f(x)$
- $\{a\} \to (f : P(a) \to L_F X) \to (x : Q(a)) \to \ell_{f \circ F}(x) = f(x)$

We can try to define localizations as a sequential colimits of *one-step F-localization* $L^1_F X$.

$$\begin{tikzcd} P(a) \arrow[r] \arrow[d] & X \arrow[d] \\ Q(a) \arrow[r, dashed] & L^1_F X \end{tikzcd}$$

`HIT` $L^1_F X :=$

- $i : X \to L^1_F X$
- $r : \{a\} \to (P(a) \to X) \to Q(a) \to L^1_F X$
- $\ell : \{a\} \to (P(a) \to X) \to Q(a) \to L^1_F X$
- $\{a\} \to (f : P(a) \to X) \to (x : P(a)) \to r_f(F(x)) = i(f(x))$
- $\{a\} \to (f : P(a) \to X) \to (x : Q(a)) \to \ell_{f \circ F}(x) = i(f(x))$

Constructing the localization

We can ask whether iterating this construction

$$X \to L_F^1 X \to L_F^2 X \to L_F^3 X \to \cdots \qquad (2)$$

gives the localization $L_F X$ in the colimit.

We cannot hope this is true in general.

If $P(a)$ and $Q(a)$ are ω-*compact*, then it is true.

Theorem

Assume that for all $a : A$ the types $P(a)$ and $Q(a)$ are ω-compact. Then the colimit of (2) is the localization of X.

A type X is ω-*compact* if for all sequences $(A_n, f_n)_{n:\mathbb{N}}$ the canonical map

$$\mathrm{colim}(X \to A_n)_n \to (X \to \mathrm{colim}(A))$$

is an equivalence.

Long exact sequence of homotopy groups

Given a pointed map $f : X \to Y$ with $F :\equiv \mathsf{fib}_f :\equiv (x : X) \times f(x) = y_0$. Then we have the following long exact sequence.

$$\begin{tikzcd}
 & \vdots & \\
\pi_2(F) \arrow[r, "\pi_2(p_1)"'] & \pi_2(X) \arrow[r, "\pi_2(f)"] & \pi_2(Y) \arrow[dll, "\pi_1(\delta)"'] \\
\pi_1(F) \arrow[r, "\pi_1(p_1)"'] & \pi_1(X) \arrow[r, "\pi_1(f)"] & \pi_1(Y) \arrow[dll, "\pi_0(\delta)"'] \\
\pi_0(F) \arrow[r, "\pi_0(p_1)"'] & \pi_0(X) \arrow[r, "\pi_0(f)"] & \pi_0(Y)
\end{tikzcd}$$

Spectral Sequences

Overview of the construction of spectral sequences:

1. Start with a sequence of maps
2. Construct an exact couple
3. Repeatedly derive this exact couple to get a spectral sequence
4. Prove the convergence theorem

Input: sequence of pointed maps OR sequence of spectrum maps.

We want to prove the convergence theorem once for both cases.

To avoid duplication we prove convergence theorem for an arbitrary exact couple.

Graded modules

An I*-graded* R*-module* M is a family of R-modules indexed over a set I.

A *graded morphism* $f : M \to M'$ *of degree* $e \equiv \mathsf{deg}_f : I \simeq I$ is a term of type

$$(x\ y : I) \to e(x) = y \to M_x \to M'_y$$

(Equivalently: $(x : I) \to M_x \to M'_{e(x)}$)

Contrast with the more convential definition: $f : (g : G) \to M_g \to M'_{g+h}$

Graded modules

Conventional Definition	This definition
$(g : G) \to M_g \to M'_{g+h}$	$(x\ y : I) \to e(x) = y \to M_x \to M'_y$
$\begin{tikzcd} M_g \arrow[r, "f"] \arrow[drr, "f' \circ f"'] & M_{g+h} \arrow[r, "f'"] & M_{(g+h)+h'} \arrow[d, "\sim"] \\ & & M_{g+(h+h')} \end{tikzcd}$	$\begin{tikzcd} M_x \arrow[r, "f"] \arrow[drr, "f' \circ f"'] & M_{e(x)} \arrow[r, "f'"] & M_{e'(e(x))} \arrow[d, equal] \\ & & M_{e'(e(x))} \end{tikzcd}$
$\begin{tikzcd} & M_{(g-h)+h} \arrow[d, "\sim"] & \\ M_{g-h} \arrow[ur, "f"] \arrow[r, dashed] & M_g \arrow[r, "f'"] & M_{g+h} \end{tikzcd}$	$\begin{tikzcd} M_{e^{-1}(x)} \arrow[r, "f"] & M_x \arrow[r, "f'"] & M_{e'(x)} \end{tikzcd}$ $e(e^{-1}(x)) = x$

Exact couples

An exact couple is a pair of graded R-modules (D, E) with the following graded homomorphisms which are exact at each vertex:

$$\begin{tikzcd} D \arrow[rr, "i"] & & D \arrow[dl, "j"] \\ & E \arrow[ul, "k"] & \end{tikzcd}$$

Constructing exact couples

Given a sequence of maps

$$\cdots \to Y_s \xrightarrow{f_s} Y_{s-1} \xrightarrow{f_{s-1}} Y_{s-2} \to \cdots$$

Let $X_s :\equiv \mathsf{fib}_{f_s}$. Every map gives a long exact sequence

$$\cdots \to \pi_n(X_s) \xrightarrow{\pi_n(p_1)} \pi_n(Y_s) \xrightarrow{\pi_n(f_s)} \pi_n(Y_{s-1}) \xrightarrow{\pi_n(\delta)} \pi_{n-1}(X_s) \to \cdots$$

which we can put together in an $\mathbb{Z} \times \mathbb{Z}$-graded exact couple:

$$\begin{tikzcd}
(\pi_n(Y_s))_{n,s} \arrow[rr, "\pi_n(f_s)"] & & (\pi_n(Y_s))_{n,s} \arrow[dl, "\pi_n(\delta)"] \\
& (\pi_n(X_s))_{n,s} \arrow[ul, "\pi_n(p_1)"] &
\end{tikzcd}$$

Derived Exact couple

From an exact couple

$$\begin{tikzcd} D \arrow[rr, "i"] & & D \arrow[dl, "j"] \\ & E \arrow[ul, "k"] & \end{tikzcd}$$

we build a *derived exact couple*

$$\begin{tikzcd} D' \arrow[rr, "i'"] & & D' \arrow[dl, "j'"] \\ & E' \arrow[ul, "k'"] & \end{tikzcd}$$

with $E' = H(E, d) = \mathrm{ker}(d)/\mathrm{im}(d)$ with differential $d :\equiv j \circ k : E \to E$.

Spectral Sequence

We iterate this process, and construct the exact couple $(E^{r+1}, D^{r+1}, i^{r+1}, j^{r+1}, k^{r+1})$ as the derived couple of $(E^r, D^r, i^r, j^r, k^r)$.

Then $(E^r, d^r)_r$ forms an *spectral sequence*.

When does such a spectral sequence converge?

Bounded exact couple

We call an exact couple (E, D, i, j, k) *bounded* if for every $x : I$ there is a bound B_x such that for all $s \geq B_x$ we have

- $D_{\mathsf{deg}_i^s(x)}$ is contractible
- $E_{\mathsf{deg}_i^s(x)}$ is contractible
- $i_{\mathsf{deg}_i^{-s}(x)}$ is surjective
- $E_{\mathsf{deg}_i^{-s}(x)}$ is contractible
- deg_i, deg_j and deg_k commute

We say that x is *stable* if $i_{\mathsf{deg}_i^{-s}(x)}$ is surjective for *all* $s \geq 1$

Convergence Theorem

Theorem

If (E, D, i, j, k) is a bounded exact couple, then

- *for each $x : I$, E_x^r and D_x^r stabilize for large enough r to E_x^∞ and D_x^∞*
- *For any $x : I$ there is a sequence of short exact sequences (assuming $\mathsf{deg}_k = \mathsf{id}$)*

$$
\begin{array}{ccccccccc}
0 & \longrightarrow & E_x^\infty & \longrightarrow & D_x^\infty & \longrightarrow & D^\infty_{\mathsf{deg}_i(x)} & \longrightarrow & 0 \\
 & & & & \vdots & & & & \\
0 & \longrightarrow & E^\infty_{\mathsf{deg}_i^s(x)} & \longrightarrow & D^\infty_{\mathsf{deg}_i^s(x)} & \longrightarrow & D^\infty_{\mathsf{deg}_i^{s+1}(x)} & \longrightarrow & 0 \\
0 & \longrightarrow & E^\infty_{\mathsf{deg}_i^{s+1}(x)} & \longrightarrow & D^\infty_{\mathsf{deg}_i^{s+1}(x)} & \longrightarrow & D^\infty_{\mathsf{deg}_i^{s+2}(x)} & \longrightarrow & 0 \\
 & & & & \vdots & & & & \\
0 & \longrightarrow & 0 & \longrightarrow & 0 & \longrightarrow & 0 & \longrightarrow & 0
\end{array}
$$

- *If $x : I$ is stable, then $D_x^\infty \cong D_x$.*

Convergence Theorem

Remarks:

- The result for stable x is denoted
$$E_x \Rightarrow D_x.$$
- From this we can get the convergence theorem in Mike Shulman's blog posts on spectral sequences
  - For both spectrum maps and pointed maps
- This theorem works for any way of indexing the spectral sequence

Serre Spectral Sequence

Theorem

Given a pointed map $f : X \to B$ with fiber F where B is simply connected. For a spectrum Y we get

$$H^p(B; H^q(F; Y)) \Rightarrow H^{p+q}(X; Y).$$

Here $H^n(X; Y) :\equiv \|X \to Y_n\|_0$ and $H^n(X; G) :\equiv H^n(X; K(G, -))$ for an abelian group G.

The Serre Spectral Sequence has many corollaries, which usually require other tools. The following also needs Hurewicz Theorem:

Corollary

$$\pi_4(\mathbb{S}^3) = \mathbb{Z}_2$$

Progress

Progress on spectral sequences:

Construct an exact couple	Mostly done
Derive an exact couple	Only maps defined
Convergence theorem	Done
Serre spectral sequence	
Applications	

Constructing spectral sequences is joint work with Jeremy Avigad, Steve Awodey, Ulrik Buchholtz, Egbert Rijke and Mike Shulman.

Future work

Main project:

- Formalize the Serre spectral sequence and its applications
    - It would be particularly nice to formalize $\pi_4(\mathbb{S}^3) = \mathbb{Z}_2$

Other projects:

- Define ω-compact localizations
- Relate higher groups to connected truncated pointed types
- Prove that the smash product forms a 1-coherent symmetric monoidal product on pointed types and do homology theory

Thank you