Localization in Homotopy Type Theory

Morgan Opie

March 22, 2019

- Joint work with J.D. Christensen, Egbert Rijke, and Luis Scoccola.
- A result of a Summer 2017 AMS MRC program.
- Preprint: *Localization in Homotopy Type Theory*, arXiv:1807.04155. Accepted.

Introduction

- Homotopy type theory (HoTT): "alternative foundation," conducive to doing homotopy theory.

Introduction

- Homotopy type theory (HoTT): “alternative foundation,” conducive to doing homotopy theory.
- Basic objects (types) interpreted as spaces. Constructions are automatically homotopy invariant.

Introduction

- Homotopy type theory (HoTT): “alternative foundation,” conducive to doing homotopy theory.
- Basic objects (types) interpreted as spaces. Constructions are automatically homotopy invariant.
- To do math in HoTT: need to “import” basic results.

Introduction

- Homotopy type theory (HoTT): "alternative foundation," conducive to doing homotopy theory.
- Basic objects (types) interpreted as spaces. Constructions are automatically homotopy invariant.
- To do math in HoTT: need to "import" basic results.
- Goal: develop a theory of localization of types. Show that it enjoys the desireable properties of localization of spaces.

Basics of HoTT

- Syntax: rules for writing down and manipulating symbols. Synthetic approach to math.

Basics of HoTT

- Syntax: rules for writing down and manipulating symbols. Synthetic approach to math.
- Semantics: seek models in various categories (e.g. the homotopy category of spaces).
- Basic objects: types.
- Basic statements: for a, A types,

$$a : A$$

("a is a term of type A").

Basics of HoTT

- Syntax: rules for writing down and manipulating symbols. Synthetic approach to math.
- Semantics: seek models in various categories (e.g. the homotopy category of spaces).
- Basic objects: types.
- Basic statements: for a, A types,

  $$a : A$$

  ("a is a term of type A").
- To avoid pathologies: fix a universe $\mathcal{U}$, a "large" type of which all other types are terms.

Type constructors

To construct new types from old: use "type constructors."
Generally, the following data might be included:

- formation rules: what input is needed. For example:

$$\frac{A, B : \mathcal{U}}{A \times B : \mathcal{U}}$$

Type constructors

To construct new types from old: use "type constructors."
Generally, the following data might be included:

- formation rules: what input is needed. For example:

$$\frac{A, B : \mathcal{U}}{A \times B : \mathcal{U}}$$

- introduction rules: how to produce terms in the new type. For example:

$$\frac{A, B : \mathcal{U}, a : A, b : B}{(a, b) : A \times B}$$

Type constructors

To construct new types from old: use "type constructors."
Generally, the following data might be included:

- formation rules: what input is needed. For example:

$$\frac{A, B : \mathcal{U}}{A \times B : \mathcal{U}}$$

- introduction rules: how to produce terms in the new type. For example:

$$\frac{A, B : \mathcal{U}, a : A, b : B}{(a, b) : A \times B}$$

- elimination rules: how to "use" terms.

$$\frac{A, B : \mathcal{U}}{(A \to B) : \mathcal{U}}$$

$$\frac{f : (A \to B), x : A}{f(x) : B}.$$

Type constructors

To construct new types from old: use "type constructors."
Generally, the following data might be included:

- formation rules: what input is needed. For example:

$$\frac{A, B : \mathcal{U}}{A \times B : \mathcal{U}}$$

- introduction rules: how to produce terms in the new type. For example:

$$\frac{A, B : \mathcal{U}, a : A, b : B}{(a, b) : A \times B}$$

- elimination rules: how to "use" terms.

$$\frac{A, B : \mathcal{U}}{(A \to B) : \mathcal{U}}$$

$$\frac{f : (A \to B), x : A}{f(x) : B}.$$

- uniqueness: "universal properties."

Dependant types

- Dependant function types, called Π-types:

$$\frac{B : A \to \mathcal{U}}{\Pi_{a \in A} B(a)},$$

$$\frac{f : \Pi_{a \in A} B(a), a : A}{f(a) : B(a)}.$$

Intuition: spaces of sections.

Dependant types

- Dependant function types, called Π-types:

$$\frac{B : A \to \mathcal{U}}{\Pi_{a \in A} B(a)},$$

$$\frac{f : \Pi_{a \in A} B(a), a : A}{f(a) : B(a)}.$$

  Intuition: spaces of sections.
- Similarly: dependant sums, or Σ-types, thought of as fibrations.

Dependant types

- Dependant function types, called Π-types:

$$\frac{B : A \to \mathcal{U}}{\Pi_{a \in A} B(a)},$$

$$\frac{f : \Pi_{a \in A} B(a), a : A}{f(a) : B(a)}.$$

  Intuition: spaces of sections.
- Similarly: dependant sums, or Σ-types, thought of as fibrations.
- Why should these be thought of as *spaces*, rather than sets? We'll return to this.

Propositions as types

Key difference from traditional foundations: any proposition (statement with truth value) is itself a type.

- To say that two elements are equal, we exhibit an element in the *identity type* $(a = b)$.

Propositions as types

Key difference from traditional foundations: any proposition (statement with truth value) is itself a type.

- To say that two elements are equal, we exhibit an element in the *identity type* $(a = b)$.
- For any $a : A$, we have a term $id_a : (a = a)$, guaranteeing $a = a$.

Propositions as types

Key difference from traditional foundations: any proposition (statement with truth value) is itself a type.

- To say that two elements are equal, we exhibit an element in the *identity type* $(a = b)$.
- For any $a : A$, we have a term $id_a : (a = a)$, guaranteeing $a = a$.
- 1. Suppose $P(x)$ and $Q(x)$ are propositions (already thought of as types).

Propositions as types

Key difference from traditional foundations: any proposition (statement with truth value) is itself a type.

- To say that two elements are equal, we exhibit an element in the *identity type* $(a = b)$.
- For any $a : A$, we have a term $id_a : (a = a)$, guaranteeing $a = a$.
- 1. Suppose $P(x)$ and $Q(x)$ are propositions (already thought of as types).
  2. (Classical) tautology "If for all $x : A$ $P(x)$ and $Q(x)$, then for all $x : A$ $P(x)$ and for all $x : AQ(x)$.
- It's not just whether something is true, but *how* it is true.

Propositions as types

Key difference from traditional foundations: any proposition (statement with truth value) is itself a type.

- To say that two elements are equal, we exhibit an element in the *identity type* $(a = b)$.
- For any $a : A$, we have a term $id_a : (a = a)$, guaranteeing $a = a$.
- 1. Suppose $P(x)$ and $Q(x)$ are propositions (already thought of as types).
  2. (Classical) tautology "If for all $x : A\ P(x)$ and $Q(x)$, then for all $x : A\ P(x)$ and for all $x : AQ(x)$.
  3. Truth comes from that fact that

$$\Big((\Pi_{a:A} P(a) \times Q(a) \to (\Pi_{x:A} P(x)) \times (\Pi_{y:A} Q(y)\Big)$$

     is inhabited (has a term). Existence of such a term can be deduced from rules for product and function types.
- It's not just whether something is true, but *how* it is true.

Higher Inductive Types and homotopy analogies

HITs: type constructors that depend not only on terms in types, but also on paths (terms in identity types).

- $S^1 : \mathcal{U}$ is the HIT type freely generated by $\mathsf{base} : S^1$ and $\mathsf{loop} : (\mathsf{base} = \mathsf{base})$.

Higher Inductive Types and homotopy analogies

HITs: type constructors that depend not only on terms in types, but also on paths (terms in identity types).

- $S^1 : \mathcal{U}$ is the HIT type freely generated by $\mathsf{base} : S^1$ and $\mathsf{loop} : (\mathsf{base} = \mathsf{base})$.
- $I : \mathcal{U}$ is the HIT freely generated by $0, 1 : I$ and $\mathsf{seg} : (0 = 1)$.

Higher Inductive Types and homotopy analogies

HITs: type constructors that depend not only on terms in types, but also on paths (terms in identity types).

- $S^1 : \mathcal{U}$ is the HIT type freely generated by $\mathsf{base} : S^1$ and $\mathsf{loop} : (\mathsf{base} = \mathsf{base})$.
- $I : \mathcal{U}$ is the HIT freely generated by $0, 1 : I$ and $\mathsf{seg} : (0 = 1)$.
- More from HITs: suspensions, loop spaces, a notion of equivalence of types ($\simeq$), Eilenberg–Mac Lane spaces, correspondence between groups and Eilenberg–Mac Lane spaces, homotopy groups...

The Univalence Axiom

There are many "typed" theories with dependant types, dating back to the mid-20th century (work of Martin-Löf). The key is the following: [1]

- **(Univalence Axiom.)** For $A, B : \mathcal{U}$,

$$(A =_{\mathcal{U}} B) \simeq (A \simeq_{\mathcal{U}} B).$$

[1] For a detailed discussion of HoTT, see *Homotopy Type Theory: Univalent Foundations of Mathematics*, produced during the IAS year on HoTT.

The Univalence Axiom

There are many "typed" theories with dependant types, dating back to the mid-20th century (work of Martin-Löf). The key is the following: [1]

- **(Univalence Axiom.)** For $A, B : \mathcal{U}$,

$$(A =_{\mathcal{U}} B) \simeq (A \simeq_{\mathcal{U}} B).$$

- This can be thought of as giving a way to identify equivalent things.

[1] For a detailed discussion of HoTT, see *Homotopy Type Theory: Univalent Foundations of Mathematics*, produced during the IAS year on HoTT.

The Univalence Axiom

There are many "typed" theories with dependant types, dating back to the mid-20th century (work of Martin-Löf). The key is the following: [1]

- **(Univalence Axiom.)** For $A, B : \mathcal{U}$,

$$(A =_{\mathcal{U}} B) \simeq (A \simeq_{\mathcal{U}} B).$$

- This can be thought of as giving a way to identify equivalent things.
- Any reasoning in HoTT which holds for A holds for all equivalent B. I.e. **all reasoning and constructions are homotopy invariant.**

[1] For a detailed discussion of HoTT, see *Homotopy Type Theory: Univalent Foundations of Mathematics*, produced during the IAS year on HoTT.

What else is different in HoTT?

- No law of the excluded middle. Classical Whitehead theorem fails.

[2] due to Rezk, following the HoTT proof of Lumsdaine–Finster–Licata.

What else is different in HoTT?

- No law of the excluded middle. Classical Whitehead theorem fails.
- Proof verification: easier than for first order logic. Proof assistants can help come up with proofs.

[2] due to Rezk, following the HoTT proof of Lumsdaine–Finster–Licata.

What else is different in HoTT?

- No law of the excluded middle. Classical Whitehead theorem fails.
- Proof verification: easier than for first order logic. Proof assistants can help come up with proofs.
- Synthetic theory: meant to be interpreted in different contexts, so results should be more general. Awodey, Warren, and Voevodsky (2006): homotopical models, including homotopy category of spaces.

[2] due to Rezk, following the HoTT proof of Lumsdaine–Finster–Licata.

What else is different in HoTT?

- No law of the excluded middle. Classical Whitehead theorem fails.
- Proof verification: easier than for first order logic. Proof assistants can help come up with proofs.
- Synthetic theory: meant to be interpreted in different contexts, so results should be more general. Awodey, Warren, and Voevodsky (2006): homotopical models, including homotopy category of spaces.
- The *initiality conjecture* roughly states that homotopy type theory should have models in (appropriate) infinity topoi, meaning that results proved in type theory hold for many interesting categories besides spaces.

[2] due to Rezk, following the HoTT proof of Lumsdaine–Finster–Licata.

What else is different in HoTT?

- No law of the excluded middle. Classical Whitehead theorem fails.
- Proof verification: easier than for first order logic. Proof assistants can help come up with proofs.
- Synthetic theory: meant to be interpreted in different contexts, so results should be more general. Awodey, Warren, and Voevodsky (2006): homotopical models, including homotopy category of spaces.
- The *initiality conjecture* roughly states that homotopy type theory should have models in (appropriate) infinity topoi, meaning that results proved in type theory hold for many interesting categories besides spaces.
- This philosophy has been fruitful: alternative proof of Blakers–Massey theorem (not relying on properties of the category of spaces). [2]

[2] due to Rezk, following the HoTT proof of Lumsdaine–Finster–Licata.

How do we work in HoTT?

- In classical math (ZFC + first order logic), we work informally. All correct proofs *could* be formalized.

How do we work in HoTT?

- In classical math (ZFC + first order logic), we work informally. All correct proofs *could* be formalized.
- Same approach in HoTT: just what can and cannot be formalized differs.

How do we work in HoTT?

- In classical math (ZFC + first order logic), we work informally. All correct proofs *could* be formalized.
- Same approach in HoTT: just what can and cannot be formalized differs.
- Proof verification is a real possibility.

Algebraic Localization

Definition

Let G be an abelian group, and p a prime. G is **uniquely p-divisible** if the p-th power map $G \to G$ is an isomorphism. A p-localization G' of a group G away from p is the universal approximation $G \to G'$ by a uniquely p-divisible group.

Algebraic Localization

Definition

Let G be an abelian group, and p a prime. G is **uniquely** p-**divisible** if the p-th power map $G \to G$ is an isomorphism. A p-localization G' of a group G away from p is the universal approximation $G \to G'$ by a uniquely p-divisible group.

- L_p is a right adjoint to the inclusion of uniquely p-divisible groups into all abelian groups.
- The unit $G \to L_p G$ is a localization of G away from p.

Algebraic Localization

Definition

Let G be an abelian group, and p a prime. G is **uniquely p-divisible** if the p-th power map $G \to G$ is an isomorphism. A p-localization G' of a group G away from p is the universal approximation $G \to G'$ by a uniquely p-divisible group.

- L_p is a right adjoint to the inclusion of uniquely p-divisible groups into all abelian groups.
- The unit $G \to L_p G$ is a localization of G away from p.
- This is very concrete. $L_p G = G \otimes_{\mathbb{Z}} \mathbb{Z}[\frac{1}{p}]$.

Algebraic Localization

Definition

Let G be an abelian group, and p a prime. G is **uniquely p-divisible** if the p-th power map $G \to G$ is an isomorphism. A p-localization G' of a group G away from p is the universal approximation $G \to G'$ by a uniquely p-divisible group.

- L_p is a right adjoint to the inclusion of uniquely p-divisible groups into all abelian groups.
- The unit $G \to L_p G$ is a localization of G away from p.
- This is very concrete. $L_p G = G \otimes_{\mathbb{Z}} \mathbb{Z}[\frac{1}{p}]$.
- More generally, we can localize away from families of numbers.

Localizing spaces

Spaces have associated algebraic invariants: homotopy groups, homology and cohomology groups, etc.

[3] for non-abelian π_1, we simply require the *p*-th power map to be bijective.

[4] Caution: sometimes *p*-local means something else!

[5] See for e.g. May–Ponto, *More Concise Algebraic Topology*.

Localizing spaces

Spaces have associated algebraic invariants: homotopy groups, homology and cohomology groups, etc.

- Rather than just localizing invariants, we seek a localization at the level of spaces.
- Classical theory: p-local spaces are those whose homotopy groups are uniquely p-divisible[3] [4].

[3] for non-abelian π_1, we simply require the p-th power map to be bijective.

[4] Caution: sometimes p-local means something else!

[5] See for e.g. May–Ponto, *More Concise Algebraic Topology*.

Localizing spaces

Spaces have associated algebraic invariants: homotopy groups, homology and cohomology groups, etc.

- Rather than just localizing invariants, we seek a localization at the level of spaces.
- Classical theory: p-local spaces are those whose homotopy groups are uniquely p-divisible[3] [4].
- Classical theory: reflect onto subcategory of p-local spaces. Works for nilpotent spaces [5]:

Theorem

Given a nilpotent space X, there is a p-local space X_p and a map $X \to X_p$ which induces algebraic localization on algebraic invariants, and such that for any other p-local Y and $X \to Y$, there is a unique factorization through the localization map $X \to X_p$.

---

[3] for non-abelian π_1, we simply require the p-th power map to be bijective.

[4] Caution: sometimes p-local means something else!

[5] See for e.g. May–Ponto, *More Concise Algebraic Topology*.

Remarks

- Why useful? Fracture theorems: can reconstruct a space from appropriate localizations, each of which is simpler.

Remarks

- Why useful? Fracture theorems: can reconstruct a space from appropriate localizations, each of which is simpler.
- Approach in May–Ponto: localize Eilenberg–Mac Lane spaces, then localize nilpotent spaces using Postnikov towers.

Remarks

- Why useful? Fracture theorems: can reconstruct a space from appropriate localizations, each of which is simpler.
- Approach in May–Ponto: localize Eilenberg–Mac Lane spaces, then localize nilpotent spaces using Postnikov towers.
- Doesn't work in HoTT: algebraic invariants are not "strong enough."

Remarks

- Why useful? Fracture theorems: can reconstruct a space from appropriate localizations, each of which is simpler.
- Approach in May–Ponto: localize Eilenberg–Mac Lane spaces, then localize nilpotent spaces using Postnikov towers.
- Doesn't work in HoTT: algebraic invariants are not "strong enough."
- Need to focus on the "structural" aspects of the problem: reflection onto subcategories.

Reflective subuniverses

Definition

A **subuniverse** of a universe $\mathcal{U}$ is a family $\mathsf{isLocal}_L : \mathcal{U} \to \mathsf{Prop}$. X is said to be L-local if $\mathsf{isLocal}_L(X)$ is inhabited. Notation:

$$\mathcal{U}_L := \Sigma_{x:\mathcal{U}} \mathsf{isLocal}_L(X).$$

A **reflective subuniverse** consists of a subuniverse $\mathcal{U}_L$ of $\mathcal{U}$, a **reflector** $L : \mathcal{U} \to \mathcal{U}_L$, and a **unit** $\eta : \Pi_{x:\mathcal{U}}(X \to LX)$.

Reflective subuniverses

Definition

A **subuniverse** of a universe $\mathcal{U}$ is a family $\mathsf{isLocal}_L : \mathcal{U} \to \mathsf{Prop}$. X is said to be L-local if $\mathsf{isLocal}_L(X)$ is inhabited. Notation:

$$\mathcal{U}_L := \Sigma_{x:\mathcal{U}} \mathsf{isLocal}_L(X).$$

A **reflective subuniverse** consists of a subuniverse $\mathcal{U}_L$ of $\mathcal{U}$, a **reflector** $L : \mathcal{U} \to \mathcal{U}_L$, and a **unit** $\eta : \Pi_{x:\mathcal{U}}(X \to LX)$.

- Much of our work on reflective subuniverses is an extension of Rijke–Shulman–Spitters, *Modalities in Homotopy Type Theory*, arXiv:1706.07526.

Reflective subuniverses: examples

- Given $f : (A \to B)$, a type X is f-local if the map $f^* : (X \to B) \to (X \to A)$ is an equivalence of types. For any f, f-local types form a reflective subuniverse with reflector denoted L_f. [6]

[6] see Rijke–Shulman–Spitters arXiv:1706.07526.

Reflective subuniverses: examples

- Given $f : (A \to B)$, a type X is f-local if the map $f^* : (X \to B) \to (X \to A)$ is an equivalence of types. For any f, f-local types form a reflective subuniverse with reflector denoted L_f. [6]
- Main example: take $f = \deg(p) : (S^1 \to S^1)$. We take $X \to L_{\deg p}X$ as our definition of the p-localization of a type.

[6] see Rijke–Shulman–Spitters arXiv:1706.07526.

Another example: truncation

Definition

For a type $A : \mathcal{U}$, the type $||A||_n$, called the *n*-**truncation of** A, is the HIT given by:

- $|-|_n : (A \to ||A||_n)$
- $\frac{r:(S^{n+1} \to ||A||_n)}{h(r):||A||_n}$
- $\frac{r:(S^{n+1} \to ||A||_n),\, x:S^{n+1}}{s_r(x):r(x)=h(r)}$.

Truncation

- By univalence: all homotopy groups above level n are trivial.

Truncation

- By univalence: all homotopy groups above level n are trivial.
- A type A is said to be n-**truncated** if the map $A \to ||A||_n$ is an equivalence. n-truncated types form a reflective subuniverse, with reflector $|-|_n$.

Truncation

- By univalence: all homotopy groups above level n are trivial.
- A type A is said to be n-**truncated** if the map $A \to ||A||_n$ is an equivalence. n-truncated types form a reflective subuniverse, with reflector $| - |_n$.
- Truncations: analogue for Postnikov sections.

Main theorem

Theorem (Christensen–O.–Rijke–Scoccola)

Let X be a simply connected type. Then the localization map $X \to L_{\mathrm{deg}(p)}X$ induces algebraic localization away from p on all homotopy groups.

Main theorem

Theorem (Christensen–O.–Rijke–Scoccola)

Let X be a simply connected type. Then the localization map $X \to L_{\deg(p)}X$ induces algebraic localization away from p on all homotopy groups.

- The proof relies on 4 facts:
  1. $\deg(p)$-localization preserves connectedness.
  2. $\deg(p)$-localization commutes with truncation for simply connected types.
  3. $\deg(p)$-localization preserves certain fiber sequences
  4. The theorem holds for Eilenberg–Mac Lane spaces $K(G, n)$.

Proof of the main theorem

The facts assemble as follows:

- fact 1 implies that π_1 is localized since we restrict to simply connected spaces.

Proof of the main theorem

The facts assemble as follows:

- fact 1 implies that π_1 is localized since we restrict to simply connected spaces.
- Now, inductively:

Proof of the main theorem

The facts assemble as follows:

- fact 1 implies that π_1 is localized since we restrict to simply connected spaces.
- Now, inductively:
- consider the fiber sequence

$$K(\pi_{n+1}X, n+2) \to ||X||_{n+1} \to ||X||_n.$$

Let $f = \deg(p)$.

Proof of the main theorem

The facts assemble as follows:

- fact 1 implies that π_1 is localized since we restrict to simply connected spaces.
- Now, inductively:
- consider the fiber sequence

$$K(\pi_{n+1}X, n+2) \to ||X||_{n+1} \to ||X||_n.$$

  Let $f = \deg(p)$.
- Using facts 3, when we apply L_f to the previous, we get a fiber sequence

$$L_f K(\pi_{n+1}X, n+2) \to L_f ||X||_{n+1} \to L_f ||X||_n.$$

Proof of theorem continued

- Using facts 1,2, and 4, we get a map of fibrations:

$$\begin{array}{ccccc} K(\pi_{n+1}X, n+2) & \to & ||X||_{n+1} & \to & ||X||_n \\ \downarrow & & \downarrow & & \downarrow \\ K((\pi_{n+1}X)_p, n+2) & \to & ||L_f X||_{n+1} & \to & ||L_f X||_n \end{array}$$

Key ideas

- The key: focus on theory of reflective subuniverses.

Key ideas

- The key: focus on theory of reflective subuniverses.
- To analyze reflective subuniverses, we develop the theory of **separated types**:

Key ideas

- The key: focus on theory of reflective subuniverses.
- To analyze reflective subuniverses, we develop the theory of **separated types**:
- Given a reflective subuniverse L, the subuniverse of separated types are those types whose identity types (loop spaces) are local.

Key ideas

- The key: focus on theory of reflective subuniverses.
- To analyze reflective subuniverses, we develop the theory of **separated types**:
- Given a reflective subuniverse L, the subuniverse of separated types are those types whose identity types (loop spaces) are local.
- This analysis allows us to understand key relationships between suspensions and localization, which is classically understood using delooping machinery.

Thank you!