Abstract type theories

Taichi Uemura

June 17, 2020. HoTTEST Conference
Goal

Define a general notion of a type theory to give a unified account of (CwF-)semantics of type theories.
**Goal**

Define a general notion of a type theory to give a unified account of (CwF-)semantics of type theories.

- We define a type theory to be a mathematical structure (category with certain structures) rather than a set of inference rules.
- (A set of inference rules is a presentation of a type theory.)
Scope

We only consider type theories with *single-layered* contexts and inference rules stable under *change of context* (substitution).

**Examples**

Martin-Löf type theory, Book HoTT, two-level type theory, CCHM cubical type theory

**Non-examples**

- Spatial type theory (Shulman 2017): contexts are split into two layers $\Delta \mid \Gamma$
- Modal type theories: inference rules may have restrictions on the form of context, so they are not stable under change of context.

Roughly, our type theories admit semantics based on CwFs.
Key concepts

Type theory presented by inference rules
Model of a type theory mathematical structure that can interpret the inference rules
Theory over a type theory presented by type symbols, term symbols and axioms written in the type theory.
For a type theory $\mathbb{T}$, theories over $\mathbb{T}$ and models of $\mathbb{T}$ are in adjunction.

\[
\text{Th}(\mathbb{T}) \perp \text{Mod}(\mathbb{T})
\]
Example: Basic dependent type theory

Definition

We call the dependent type theory without any type constructors the *basic dependent type theory* (DTT for short).

The only inference rules of DTT are the structural rules of weakening, projection and substitution.
Example: Basic dependent type theory

- A category with families (CwF) (Dybjer 1996) is a model of DTT.
- A generalized algebraic theory (GAT) (Cartmell 1978) is a theory over DTT.
- An example of a GAT is the theory of a category.

\[
\begin{align*}
O &: () \rightarrow \text{Type} \\
M &: (x : O, y : O) \rightarrow \text{Type} \\
i &: (x : O) \rightarrow M(x, x) \\
c &: (x : O, y : O, z : O, f : M(y, z), g : M(x, y)) \rightarrow M(x, z) \\
\_ &: (x : O, y : O, f : M(x, y)) \rightarrow c(x, y, y, i(y), f) = f \\
\_ &: (x : O, y : O, f : M(x, y)) \rightarrow c(x, x, y, f, i(x)) = f \\
\_ &: \{\text{equation for associativity}\}
\end{align*}
\]
Example: Basic dependent type theory

\[ \text{GAT} \quad \perp \quad \text{CwF} \]
Goal

We define the following notions:

- a *type theory*;
- a *model of a type theory*;
- a *theory over a type theory*

and then establish

- theory-model correspondence.
More precisely, we develop functorial semantics of type theories.

- A type theory is defined to be a category equipped with certain structures.
- A model of $\mathbb{T}$ is a structure-preserving functor from $\mathbb{T}$ to a presheaf category.
- A theory over $\mathbb{T}$ is defined in some way.

We then establish

- theory-model correspondence.
Abstract type theories
Taichi Uemura

Introduction
Models of a type theory
Theories over a type theory
Theory-model correspondence

References
Abstract type theories

Taichi Uemura

Introduction

Models of a type theory

Theories over a type theory

Theory-model correspondence

References
Natural models

An alternative definition of a category with families.

Definition (Awodey (2018))

A natural model consists of:

- a category $\mathcal{C}$ with a terminal object;
- a map $\partial : E \to U$ of presheaves over $\mathcal{C}$ that is representable: for any object $\Gamma \in \mathcal{C}$ and section $A : y(\Gamma) \to U$, the pullback $A^*E$ is representable. In other words, we have an object $\{A\} \in \mathcal{C}$ and a pullback of the form

\[
\begin{array}{ccc}
  y(\{A\}) & \xrightarrow{q} & E \\
  y(p) & \downarrow & \downarrow \partial \\
  y(\Gamma) & \xrightarrow{A} & U.
\end{array}
\]
Natural model semantics

Type theory

<table>
<thead>
<tr>
<th>Context</th>
<th>Type</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma \vdash \text{Ctx}$</td>
<td>$\Gamma \in \mathcal{C}$</td>
<td></td>
</tr>
<tr>
<td>$\Gamma \vdash A : \text{Type}$</td>
<td>$A : \mathbf{y}(\Gamma) \rightarrow \mathcal{U}$</td>
<td></td>
</tr>
<tr>
<td>$\Gamma, x : A \vdash \text{Ctx}$</td>
<td>${A} \in \mathcal{C}$</td>
<td></td>
</tr>
<tr>
<td>$(\Gamma, x : A) \rightarrow \Gamma$</td>
<td>$p : {A} \rightarrow \Gamma$</td>
<td></td>
</tr>
<tr>
<td>$\Gamma, x : A \vdash x : A$</td>
<td>$q : \mathbf{y}{{A}} \rightarrow \mathcal{E}$</td>
<td></td>
</tr>
</tbody>
</table>
Type constructors on natural models

Type constructors are modeled by maps between presheaves.

Example

An *extensional* \( \text{Id-type structure on} \ \partial \) is a pullback of the form

\[
\begin{array}{ccc}
E & \xrightarrow{\text{refl}} & E \\
\Delta \downarrow & & \partial \\
E \times_{U} E & \xrightarrow{\text{Id}} & U.
\end{array}
\]

How to model \( \Pi \)-types which bind a variable?
Polynomial functors

The pullback functor $\partial^* : X/U \to X/E$, where $X = [\text{C}^{\text{op}}, \text{Set}]$, has a right adjoint $\partial_*$ called the pushforward along $\partial$. The polynomial functor $P_{\partial}$ associated with $\partial$ is the composite

$$
X \xrightarrow{(- \times E)} X/E \xrightarrow{\partial_*} X/U \xrightarrow{\text{dom}} X.
$$
Polynomial functors

The pullback functor $\partial^* : \mathcal{X}/U \to \mathcal{X}/E$, where $\mathcal{X} = [\text{Cop}, \text{Set}]$, has a right adjoint $\partial_*$ called the pushforward along $\partial$. The polynomial functor $P_{\partial}$ associated with $\partial$ is the composite

$$\mathcal{X} \xrightarrow{(- \times E)} \mathcal{X}/E \xrightarrow{\partial_*} \mathcal{X}/U \xrightarrow{\text{dom}} \mathcal{X}.$$

**Proposition**

*When $\partial$ is representable, we have for any presheaf $\mathcal{X}$*

$$\{y(\Gamma) \to P_{\partial}\mathcal{X}\} \simeq \{(A, x) \mid A : y(\Gamma) \to U, x : y(\{A\}) \to \mathcal{X}\}.$$

*In particular, $P_{\partial}U$ classifies families of types, and $P_{\partial}E$ classifies families of terms.*
Variable binding

Type and term constructors that bind some variables are modeled using $P_\partial$ or $\partial_\ast$.

Example

A $\Pi$-type structure on $\partial$ is a pullback of the form

$$
\begin{array}{ccc}
P_\partial E & \xrightarrow{\lambda} & E \\
P_\partial \partial \downarrow & & \downarrow \partial \\
P_\partial U & \xrightarrow{\Pi} & U.
\end{array}
$$

- $\Pi$ sends a pair $(A_1, A_2)$ of types $A_1 : \text{y}(\Gamma) \rightarrow U$ and $A_2 : \text{y}([A_1]) \rightarrow U$ to a type $\Pi(A_1, A_2) : \text{y}(\Gamma) \rightarrow U$.
- Sections $\text{y}(\Gamma) \rightarrow E$ over $\Pi(A_1, A_2)$ are equivalent to sections $\text{y}([A_1]) \rightarrow E$ over $A_2$. 
A natural model is a diagram in a presheaf category written in the language of
- representable maps;
- finite limits;
- pushforwards along representable maps.
A natural model is a diagram in a presheaf category written in the language of

- representable maps;
- finite limits;
- pushforwards along representable maps.

Idea

A natural model is a structure-preserving functor from a category equipped with such structures.
Categories with representable maps

**Definition**

A *category with representable maps* consists of:

- a category $\mathcal{C}$;
- a class of maps in $\mathcal{C}$ called *representable maps*;
- finite limits in $\mathcal{C}$;
- *pushforwards* along representable maps

satisfying certain closure properties. A *morphism of categories with representable maps* is a functor preserving these structures.

**Example**

The presheaf category $[\mathcal{C}^{\text{op}}, \textbf{Set}]$ for an arbitrary category $\mathcal{C}$.
### Type theories

A *type theory* is a (small) category with representable maps.

Let $\mathbb{T}$ be a type theory. A *model of* $\mathbb{T}$ consists of:

- a category $\mathbb{M}(\ast)$ with a terminal object;
- a **structure-preserving functor** $\mathbb{M} : \mathbb{T} \to [\mathbb{M}(\ast)^{\text{op}}, \text{Set}]$ (morphism of categories with representable maps).
Example: Basic dependent type theory

**Definition**

We define the *basic dependent type theory* to be the type theory (category with representable maps) \( G \) freely generated by a representable map \( \partial : E \to U \).

**Universal property of \( G \)**

The morphisms \( G \to C \) of categories with representable maps are equivalent to the representable maps in \( C \).

So, a model of \( G \) consists of:

- a category \( M(\star) \) with a terminal object;
- a representable map \( M(\partial) : M(E) \to M(U) \) of presheaves over \( M(\star) \), that is, a natural model.
Example: \( \Pi \)-types

Consider a type theory \( \mathcal{G}^{\Pi} \) freely generated by a representable map \( \partial : E \to U \) and a pullback of the form

\[
\begin{array}{ccc}
P\partial E & \xrightarrow{\lambda} & E \\
P\partial U & \xrightarrow{\Pi} & U.
\end{array}
\]

A model of \( \mathcal{G}^{\Pi} \) consists of:

- a category \( \mathcal{M}(\ast) \) with a terminal object;
- a representable map \( \mathcal{M}(\partial) : \mathcal{M}(E) \to \mathcal{M}(U) \) of presheaves over \( \mathcal{M}(\ast) \);
- a \( \Pi \)-type structure on \( \mathcal{M}(\partial) \).
Strategy for encoding type theories

In general, we represent *inference rules as morphisms* in a category with representable maps $\mathbb{T}$.

**Example**

The morphism $\Pi : P_0 U \to U$ in $G^\Pi$ corresponds to the inference rule

$$
\begin{array}{c}
\vdash A : \text{Type} \\
\vdash B : \text{Type}
\end{array}
\xRightarrow{\pi} \\
\vdash \prod_x A \vdash B : \text{Type}
$$
Strategy for encoding type theories

In general, we represent *inference rules as morphisms* in a category with representable maps $\mathbb{T}$.

**Example**

The morphism $\Pi : P_0 U \to U$ in $\mathcal{G}^\Pi$ corresponds to the inference rule

$$
\frac{\vdash A : \text{Type} \quad \forall x : A \vdash B : \text{Type}}{\vdash \prod_{x:A} B : \text{Type}}
$$

Objects in $\mathbb{T}$ are then *judgment forms*.

**Example**

The object $U \in \mathcal{G}$ corresponds to the judgment form $\vdash \_ : \text{Type}$. 
Strategy for encoding type theories

A morphism \( \partial : E \rightarrow U \) in \( \mathbb{T} \), regarded as an object of \( \mathbb{T}/U \), is a family of judgment forms.

Example

The object \( E \in G/U \) corresponds to the family of judgment forms \( ( \vdash _{-} : A )_{A : \text{Type}} \).
A morphism $\partial : E \rightarrow U$ in $T$, regarded as an object of $T/U$, is a family of judgment forms.

**Example**

The object $E \in G/U$ corresponds to the family of judgment forms $(\vdash \_ : A)_{A:\text{Type}}$.

We make a morphism $\partial : E \rightarrow U$ representable when judgments of the type theory can have hypotheses of the form $(x : E(A))$.

**Example**

- The morphism $\partial : E \rightarrow U$ in $G$ should be representable because judgments in DTT can have hypotheses of the form $(x : A)$ for $A : \text{Type}$.
- But $U \rightarrow 1$ should not be representable, because judgments in DTT cannot have hypotheses of the form $(X : \text{Type})$. 

References
More complicated example: Cubical type theory

One can define cubical type theory to be the category with representable maps freely generated by:

- a representable map $\partial : E \to U$ (corresponding to $(\vdash \_ : Type)$ and $(\vdash \_ : A)_{A:Type}$);
- a representable map $t : 1 \to \Omega$ (corresponding to $(\vdash \_ : Cof)$ and $(\vdash \varphi)_{\varphi:Cof}$);
- a representable map $\mathbb{I} \to 1$ (corresponding to $(\vdash \_ : \mathbb{I})$);
- morphisms corresponding to inference rules.
Summary

A type theory $\mathbb{T}$ is a category with

- representable maps;
- finite limits;
- pushforwards along representable maps.

A model of $\mathbb{T}$ is a structure-preserving functor into a presheaf category.
Goal

We define the following notions:

- a *type theory*;
- a *model of a type theory*;
- a *theory over a type theory*

and then establish

- theory-model correspondence.
Abstract type theories

Introduction

Models of a type theory

Theories over a type theory

Theory-model correspondence

References
Theories as algebras

Definition (informal)

A *theory over* $\mathcal{T}$ is something presented by type symbols, term symbols and axioms.
Theories as algebras

**Definition (informal)**

A *theory over* $T$ is something presented by type symbols, term symbols and axioms.

Given such symbols and axioms, the sets of types and terms generated by them under the type constructors of $T$ form an algebra (a model of an essentially algebraic theory).

**Example**

Given a GAT, we have
- the set $U_n$ of types over contexts of length $n$;
- the set $E_n$ of terms over contexts of length $n$;
- (partial) operators between $U_n$’s and $E_n$’s defined by the structural rules.
Theorem (Garner (2015). See also Isaev (2018) and Voevodsky (2014).)

The category \textbf{GAT} of GATs and equivalence classes of their interpretations is equivalent to a category of algebras whose underlying sets are $\mathbb{U}_n$'s and $\mathbb{E}_n$'s.
Theories as algebras

Theorem (Garner (2015). See also Isaev (2018) and Voevodsky (2014).)

The category GAT of GATs and equivalence classes of their interpretations is equivalent to a category of algebras whose underlying sets are $U_n$'s and $E_n$'s.

Definition (still informal)

A theory over $T$ is an algebra of types and terms.
Algebras = Left exact functors

Theorem (Adámek and Rosický (1994) and Gabriel and Ulmer (1971))

Let $\mathcal{C}$ be a category of algebras. Then $\mathcal{C}$ is locally finitely presentable. Consequently, one can find a (small) category $\Sigma$ with finite limits such that $\mathcal{C} \simeq \text{Lex}(\Sigma, \text{Set})$, the category of functors preserving finite limits.
Abstract type theories

Introduction

Models of a type theory

Theories over a type theory

Theory-model correspondence

References

Algebras = Left exact functors

Theorem (Adámek and Rosický (1994) and Gabriel and Ulmer (1971))

Let $\mathcal{C}$ be a category of algebras. Then $\mathcal{C}$ is locally finitely presentable. Consequently, one can find a (small) category $\Sigma$ with finite limits such that $\mathcal{C} \simeq \text{Lex}(\Sigma, \text{Set})$, the category of functors preserving finite limits.

Idea

Given a type theory $\mathbb{T}$, find a suitable category $\Sigma_\mathbb{T}$ with finite limits and define a theory over $\mathbb{T}$ to be a functor $\Sigma_\mathbb{T} \to \text{Set}$ preserving finite limits.
In fact, we can simply put $\Sigma_T := T$. For example:

**Theorem**

\[
GAT \simeq \text{Lex}(G, \text{Set}).
\]

**Idea of proof.**

Given a functor $K : G \to \text{Set}$ preserving finite limits, one can think of:

- $K(P^n_u)$ as the set of types over contexts of length $n$;
- $K(P^n_e)$ as the set of terms over contexts of length $n$,

and then $K(P^n_u)$’s and $K(P^n_e)$’s form an algebra of types and terms.
Theories over a type theory

Definition

A theory over $T$ is a functor $T \to \mathbf{Set}$ preserving finite limits.
Summary

Definition

Let $T$ be a type theory (i.e. a category with representable maps).

A model of $T$ is a pair $(\mathcal{M}(\star), M)$ consisting of a category $\mathcal{M}(\star)$ with a terminal object and a morphism $M : T \to [\mathcal{M}(\star)^{\text{op}}, \text{Set}]$ of categories with representable maps.

A theory over $T$ is a functor $T \to \text{Set}$ preserving finite limits.
We define the following notions:

- a *type theory*;
- a *model of a type theory*;
- a *theory over a type theory*

and then establish

- theory-model correspondence.
Abstract type theories

Introduction
Models of a type theory
Theories over a type theory
Theory-model correspondence

References
We construct an adjunction.

\[ \text{Th}(T) \cong \text{Mod}(T) \]

- The left adjoint \( F \) assigns a syntactic model to each theory over \( T \);
- The right adjoint \( L \) assigns an internal language to each model of \( T \).

All constructions and proofs are purely category-theoretic.
Let $\mathbb{T}$ be a type theory.

**Definition**

For a model $\mathbb{M}$ of $\mathbb{T}$, we have a theory over $\mathbb{T}$

$$\mathbb{T} \xrightarrow{\mathbb{M}} \left[\mathbb{M}(\star)^{\text{op}}, \mathbb{Set}\right] \xrightarrow{\text{ev}_1} \mathbb{Set}$$

which we call the *internal language* of $\mathbb{M}$.

The internal languages define a functor

$$L : \text{Mod}(\mathbb{T}) \to \text{Th}(\mathbb{T})$$

from a category of models of $\mathbb{T}$ to a category of theories over $\mathbb{T}$. 
Theorem

The functor $L : \text{Mod}(\mathbb{T}) \to \text{Th}(\mathbb{T})$ has a fully faithful left adjoint $\mathcal{F} : \text{Th}(\mathbb{T}) \to \text{Mod}(\mathbb{T})$. We call $\mathcal{F}(\mathbb{K})$ the syntactic model generated by $\mathbb{K}$. 
Democratic models

**Definition**

Let $\mathcal{M}$ be a model of $\mathcal{T}$. The class of *contextual objects* is the smallest class of objects of $\mathcal{M}(\star)$ containing the terminal object and closed under context comprehension. We say $\mathcal{M}$ is *democratic* if every object of $\mathcal{M}(\star)$ is contextual. $\text{Mod}^{\text{dem}}(\mathcal{T})$ denotes the full subcategory of $\text{Mod}(\mathcal{T})$ spanned by the democratic models.

**Theorem**

The essential image of $\mathcal{F} : \text{Th}(\mathcal{T}) \rightarrow \text{Mod}(\mathcal{T})$ is $\text{Mod}^{\text{dem}}(\mathcal{T})$. Therefore, we have an equivalence

$$\text{Mod}^{\text{dem}}(\mathcal{T}) \simeq \text{Th}(\mathcal{T}).$$
Goal

We define the following notions:

☑ a type theory;
☑ a model of a type theory;
☑ a theory over a type theory

and then establish

☑ theory-model correspondence.
Most of our results can be translated into the language of $\infty$-categories, leading us to a notion of an $\infty$-type theory (joint work with Hoang Kim Nguyen).

**Theorem**

- We find an $\infty$-type theory $E_\infty$ such that
  \[ \text{Th}(E_\infty) \simeq \text{Lex}_\infty. \]
- We find an $\infty$-type theory $E^\Pi_\infty$ such that
  \[ \text{Th}(E^\Pi_\infty) \simeq \text{LCCC}_\infty. \]
- and more...


