# Abstract type theories

Taichi Uemura

June 17, 2020. HoTTEST Conference

# Goals

## Goal

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

# Goals

## 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.

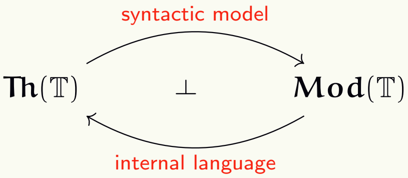
# Key concepts

**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.



# Example: Basic dependent type theory

## Definition

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

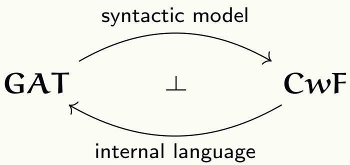
The only inference rules of DTT are the structural rules of weakening, projection and substitution.

# 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.

$$
\begin{aligned}
O &: () \Rightarrow \mathtt{Type} \\
M &: (x : O, y : O) \Rightarrow \mathtt{Type} \\
i &: (x : O) \Rightarrow M(x, x) \\
c &: (x : O, y : O, z : O, f : M(y, z), g : M(x, y)) \Rightarrow M(x, z) \\
\_ &: (x : O, y : O, f : M(x, y)) \Rightarrow c(x, y, y, i(y), f) = f \\
\_ &: (x : O, y : O, f : M(x, y)) \Rightarrow c(x, x, y, f, i(x)) = f \\
\_ &: \{\text{equation for associativity}\}
\end{aligned}
$$

# Example: Basic dependent type theory



# 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.

# Goal

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.

# Outline

# Outline

# 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 $A : \mathbf{y}(\Gamma) \to U$, the pullback $A^*E$ is representable. In other words, we have an object $\{A\} \in \mathcal{C}$ and a pullback of the form

$$\begin{array}{ccc} \mathbf{y}(\{A\}) & \xrightarrow{q} & E \\ {\scriptstyle \mathbf{y}(p)}\downarrow & & \downarrow{\scriptstyle \partial} \\ \mathbf{y}(\Gamma) & \xrightarrow[A]{} & U. \end{array}$$

# Natural model semantics

$$\begin{array}{ccc} \mathbf{y}(\{A\}) & \xrightarrow{q} & E \\ {\scriptstyle \mathbf{y}(p)}\downarrow & & \downarrow{\scriptstyle \partial} \\ \mathbf{y}(\Gamma) & \xrightarrow[A]{} & U \end{array}$$

| Type theory | Natural model |
|---|---|
| $\Gamma \vdash \mathtt{Ctx}$ | $\Gamma \in \mathcal{C}$ |
| $\Gamma \vdash A : \mathtt{Type}$ | $A : \mathbf{y}(\Gamma) \to U$ |
| $\Gamma, x : A \vdash \mathtt{Ctx}$ | $\{A\} \in \mathcal{C}$ |
| $(\Gamma, x : A) \to \Gamma$ | $p : \{A\} \to \Gamma$ |
| $\Gamma, x : A \vdash x : A$ | $q : \mathbf{y}(\{A\}) \to E$ |

# Type constructors on natural models

Type constructors are modeled by maps between presheaves.

## Example

An *extensional* Id-*type structure on* $\partial$ is a pullback of the form

$$\begin{array}{ccc} E & \overset{\mathsf{refl}}{\dashrightarrow} & E \\ {\scriptstyle \Delta}\downarrow & & \downarrow{\scriptstyle \partial} \\ E \times_U E & \underset{\mathsf{Id}}{\dashrightarrow} & U. \end{array}$$

How to model Π-types which bind a variable?

# Polynomial functors

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

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

# Polynomial functors

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

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

## Proposition

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

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

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 $\mathrm{P}_\partial$ or $\partial_*$.

## Example

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

$$\begin{array}{ccc} \mathrm{P}_\partial \mathrm{E} & \overset{\lambda}{\dashrightarrow} & \mathrm{E} \\ {\scriptstyle \mathrm{P}_\partial \partial}\downarrow & & \downarrow{\scriptstyle \partial} \\ \mathrm{P}_\partial \mathrm{U} & \underset{\Pi}{\dashrightarrow} & \mathrm{U}. \end{array}$$

- $\Pi$ sends a pair $(A_1, A_2)$ of types $A_1 : \mathbf{y}(\Gamma) \to \mathrm{U}$ and $A_2 : \mathbf{y}(\{A_1\}) \to \mathrm{U}$ to a type $\Pi(A_1, A_2) : \mathbf{y}(\Gamma) \to \mathrm{U}$.
- Sections $\mathbf{y}(\Gamma) \to \mathrm{E}$ over $\Pi(A_1, A_2)$ are equivalent to sections $\mathbf{y}(\{A_1\}) \to \mathrm{E}$ over $A_2$.

# Language of natural models

A natural model is a diagram in a presheaf category written in the language of

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

# Language of natural models

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 $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$ for an arbitrary category $\mathcal{C}$.

# Type theories

## Definition

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

## Definition

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

- a category $\mathcal{M}(\star)$ with a terminal object;
- a structure-preserving functor $\mathcal{M} : \mathbb{T} \to [\mathcal{M}(\star)^{\mathrm{op}}, \boldsymbol{Set}]$ (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) $\mathbb{G}$ freely generated by a representable map $\partial : \mathrm{E} \to \mathrm{U}$.

## Universal property of $\mathbb{G}$

The morphisms $\mathbb{G} \to \mathcal{C}$ of categories with representable maps are equivalent to the representable maps in $\mathcal{C}$.

So, a model of $\mathbb{G}$ consists of:

- a category $\mathcal{M}(\star)$ with a terminal object;
- a representable map $\mathcal{M}(\partial) : \mathcal{M}(\mathrm{E}) \to \mathcal{M}(\mathrm{U})$ of presheaves over $\mathcal{M}(\star)$,

that is, a natural model.

# Example: Π-types

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

$$\begin{array}{ccc} \mathrm{P}_{\partial}\mathrm{E} & \xrightarrow{\lambda} & \mathrm{E} \\ {\scriptstyle \mathrm{P}_{\partial}\partial}\downarrow & & \downarrow{\scriptstyle \partial} \\ \mathrm{P}_{\partial}\mathrm{U} & \xrightarrow[\Pi]{} & \mathrm{U}. \end{array}$$

A model of $\mathbb{G}^{\Pi}$ consists of:

- a category $\mathcal{M}(\star)$ with a terminal object;
- a representable map $\mathcal{M}(\partial) : \mathcal{M}(\mathrm{E}) \to \mathcal{M}(\mathrm{U})$ of presheaves over $\mathcal{M}(\star)$;
- a Π-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 $\Pi : P_{\partial}U \to U$ in $\mathbb{G}^{\Pi}$ corresponds to the inference rule

$$\frac{\vdash A : \mathtt{Type} \qquad x : A \vdash B : \mathtt{Type}}{\vdash \prod_{x:A} B : \mathtt{Type}}$$

# Strategy for encoding type theories

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

## Example

The morphism $\Pi : P_\partial U \to U$ in $\mathbb{G}^\Pi$ corresponds to the inference rule

$$\frac{\vdash A : \mathtt{Type} \qquad x : A \vdash B : \mathtt{Type}}{\vdash \prod_{x:A} B : \mathtt{Type}}$$

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

## Example

The object $U \in \mathbb{G}$ corresponds to the judgment form $\vdash \_ : \mathtt{Type}$.

# Strategy for encoding type theories

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

## Example

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

# Strategy for encoding type theories

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

## Example

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

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

## Example

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

# 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 $(\vdash \_ : \mathtt{Type})$ and $(\vdash \_ : A)_{A:\mathtt{Type}}$);
- a representable map $t : 1 \to \Omega$ (corresponding to $(\vdash \_ : \mathtt{Cof})$ and $(\vdash \varphi)_{\varphi:\mathtt{Cof}}$);
- 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.

# Outline

# Theories as algebras

## Definition (informal)

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

# Theories as algebras

## Definition (informal)

A *theory over* $\mathbb{T}$ 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 $\mathbb{T}$ 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.

# Theories as algebras

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

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

# Theories as algebras

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

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

**Definition (still informal)**

A *theory over* $\mathbb{T}$ 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 $\mathcal{C} \simeq \mathbf{Lex}(\Sigma, \mathbf{Set})$, the category of functors preserving finite limits.*

# 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 $\mathcal{C} \simeq \mathbf{Lex}(\Sigma, \mathbf{Set})$, 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 $\Sigma_{\mathbb{T}} \to \mathbf{Set}$ preserving finite limits.

# Theories over $\mathbb{G}$

In fact, we can simply put $\Sigma_{\mathbb{T}} := \mathbb{T}$. For example:

## Theorem

$$\mathbf{GAT} \simeq \mathbf{Lex}(\mathbb{G}, \mathcal{S}et).$$

## Idea of proof.

Given a functor $K : \mathbb{G} \to \mathcal{S}et$ preserving finite limits, one can think of:

- $K(P^n_\partial U)$ as the set of types over contexts of length $n$;
- $K(P^n_\partial E)$ as the set of terms over contexts of length $n$,

and then $K(P^n_\partial U)$'s and $K(P^n_\partial E)$'s form an algebra of types and terms. □

# Theories over a type theory

## Definition

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

# Summary

## Definition

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

- A *model of* $\mathbb{T}$ is a pair $(\mathcal{M}(\star), \mathcal{M})$ consisting of a category $\mathcal{M}(\star)$ with a terminal object and a morphism $\mathcal{M} : \mathbb{T} \to [\mathcal{M}(\star)^{\mathsf{op}}, \boldsymbol{Set}]$ of categories with representable maps.
- A *theory over* $\mathbb{T}$ is a functor $\mathbb{T} \to \boldsymbol{Set}$ preserving finite limits.

# 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.

# Outline

# Theory-model correspondence

We construct an adjunction.

$$\mathbf{Th}(\mathbb{T}) \underset{\mathsf{L}}{\overset{\mathcal{F}}{\rightleftarrows}} \mathbf{Mod}(\mathbb{T}) \qquad (\mathcal{F} \dashv \mathsf{L})$$

- The left adjoint $\mathcal{F}$ assigns a syntactic model to each theory over $\mathbb{T}$;
- The right adjoint $\mathsf{L}$ assigns an internal language to each model of $\mathbb{T}$.

All constructions and proofs are purely category-theoretic.

# Internal languages

Let $\mathbb{T}$ be a type theory.

## Definition

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

$$\mathbb{T} \xrightarrow{\mathcal{M}} [\mathcal{M}(\star)^{\mathrm{op}}, \mathbf{Set}] \xrightarrow{\mathrm{ev}_1} \mathbf{Set}$$

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

The internal languages define a functor

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

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

# Syntactic models

**Theorem**

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

# Democratic models

## Definition

Let $\mathcal{M}$ be a model of $\mathbb{T}$. 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. $\mathbf{Mod}^{\mathsf{dem}}(\mathbb{T})$ denotes the full subcategory of $\mathbf{Mod}(\mathbb{T})$ spanned by the democratic models.

## Theorem

*The essential image of $\mathcal{F} : \mathbf{Th}(\mathbb{T}) \to \mathbf{Mod}(\mathbb{T})$ is $\mathbf{Mod}^{\mathsf{dem}}(\mathbb{T})$. Therefore, we have an equivalence*

$$\mathbf{Mod}^{\mathsf{dem}}(\mathbb{T}) \simeq \mathbf{Th}(\mathbb{T}).$$

# 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.

# $\infty$-type theories

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* $\mathbb{E}_\infty$ *such that*

$$\mathbf{Th}(\mathbb{E}_\infty) \simeq \mathbf{Lex}_\infty.$$

- *We find an* $\infty$*-type theory* $\mathbb{E}^\Pi_\infty$ *such that*

$$\mathbf{Th}(\mathbb{E}^\Pi_\infty) \simeq \mathbf{LCCC}_\infty.$$

- *and more...*

# References I

Jiří Adámek and Jiří Rosický (1994). *Locally Presentable and Accessible Categories*. Vol. 189. London Mathematical Society Lecture Note Series. Cambridge University Press.

Steve Awodey (2018). "Natural models of homotopy type theory". In: *Mathematical Structures in Computer Science* 28.2, pp. 241–286. DOI: 10.1017/S0960129516000268.

J. W. Cartmell (1978). "Generalised algebraic theories and contextual categories". PhD thesis. Oxford University.

Peter Dybjer (1996). "Internal Type Theory". In: *Types for Proofs and Programs: International Workshop, TYPES '95 Torino, Italy, June 5–8, 1995 Selected Papers*. Ed. by Stefano Berardi and Mario Coppo. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 120–134. DOI: 10.1007/3-540-61780-9_66.

Peter Gabriel and Friedrich Ulmer (1971). *Lokal präsentierbare Kategorien*. Vol. 221. Lecture Notes in Mathematics. Springer-Verlag, Berlin, Heidelberg. DOI: 10.1007/BFb0059396.

# References II

Richard Garner (2015). "Combinatorial structure of type dependency". In: *Journal of Pure and Applied Algebra* 219.6, pp. 1885–1914. DOI: 10.1016/j.jpaa.2014.07.015.

Valery Isaev (2018). *Algebraic Presentations of Dependent Type Theories*. arXiv: 1602.08504v3.

Michael Shulman (2017). "Brouwer's fixed-point theorem in real-cohesive homotopy type theory". In: *Mathematical Structures in Computer Science*, pp. 1–86. DOI: 10.1017/S0960129517000147.

Vladimir Voevodsky (2014). *B-systems*. arXiv: 1410.5389v1.