Categorical Semantics for Logic-Enriched Type Theories

Robin Adams

Royal Holloway, University of London

TYPES 2011, 8 September 2011

The Curry-Howard Isomorphism

There are two facts that are both sometimes referred to as the *Curry-Howard isomorphism*. One is trivial, one is not.

The Curry-Howard Isomorphism

There are two facts that are both sometimes referred to as the *Curry-Howard isomorphism*. One is trivial, one is not.

Trivial Fact

It is possible to write a linear syntax for natural deduction proofs, and then write $\Gamma \vdash P : \phi$ for 'P is a proof of ϕ (that depends on the free variables and hypotheses Γ)'

The Curry-Howard Isomorphism

There are two facts that are both sometimes referred to as the *Curry-Howard isomorphism*. One is trivial, one is not.

Non-trivial Fact

When we do so:

- the rules for conjunction are the rules for product type;
- the rules for impliciation are the rules for non-dependent function type;
- the rules for universal quantification are (almost) the rules for dependent function type;
- the rules for classical logic are the rules for control operators (usually);
- the rules for modal logic are the rules for metavariables;
- etc.

The Curry-Howard Isomorphism

There are two facts that are both sometimes referred to as the *Curry-Howard isomorphism*. One is trivial, one is not.

Non-trivial Fact

When we do so:

- the rules for conjunction are the rules for product type;
- the rules for impliciation are the rules for non-dependent function type;
- the rules for universal quantification are (almost) the rules for dependent function type;
- the rules for classical logic are the rules for control operators (usually);
- the rules for modal logic are the rules for metavariables;
- etc.

In this talk, 'Curry-Howard' shall mean the second.

My Beliefs on Curry-Howard

My Beliefs on Curry-Howard

I believe:

1. Curry-Howard is surprising,

My Beliefs on Curry-Howard

I believe:

1. Curry-Howard is surprising,
2. There is something there to be explained. (Why do propositions behave like types?)

My Beliefs on Curry-Howard

I believe:

1. Curry-Howard is surprising,
2. There is something there to be explained. (Why do propositions behave like types?)
3. We do not have a good explanation yet. (Propositions are not literally types.)

My Beliefs on Curry-Howard

I believe:

1. Curry-Howard is surprising,
2. There is something there to be explained. (Why do propositions behave like types?)
3. We do not have a good explanation yet. (Propositions are not literally types.)
4. We are having problems because we tacetly *assume* propositions-as-types.

My Beliefs on Curry-Howard

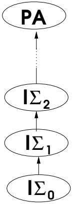
I believe:

1. Curry-Howard is surprising,
2. There is something there to be explained. (Why do propositions behave like types?)
3. We do not have a good explanation yet. (Propositions are not literally types.)
4. We are having problems because we tacetly *assume* propositions-as-types.
5. We should instead turn Curry-Howard into a mathematical object.

Introduction

We have:

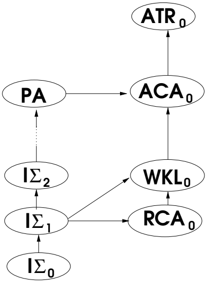
- systems of first-order arithmetic



Introduction

We have:

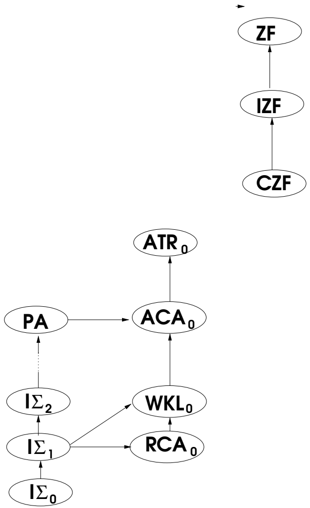
- systems of first-order arithmetic
- systems of second-order arithmetic



Introduction

We have:

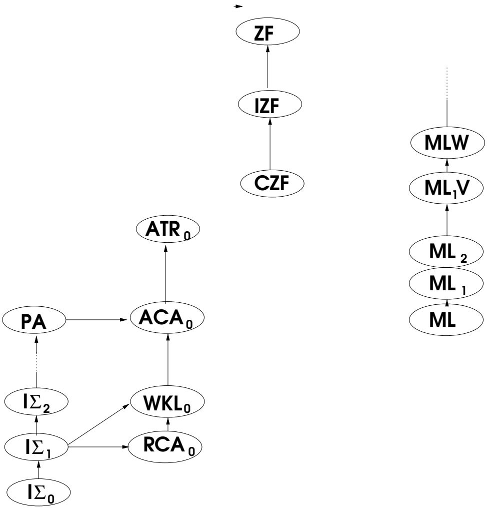
- systems of first-order arithmetic
- systems of second-order arithmetic
- set theories



Introduction

We have:

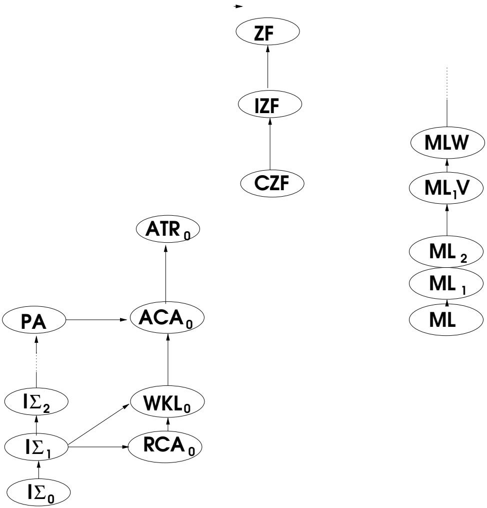
- systems of first-order arithmetic
- systems of second-order arithmetic
- set theories
- type theories



Introduction

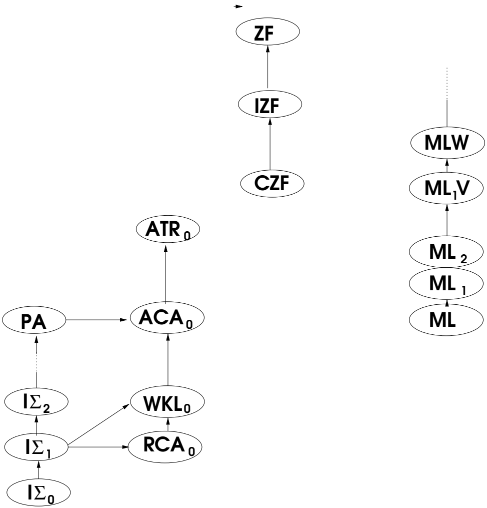
We have:

- systems of first-order arithmetic
- systems of second-order arithmetic
- set theories
- type theories
- ...



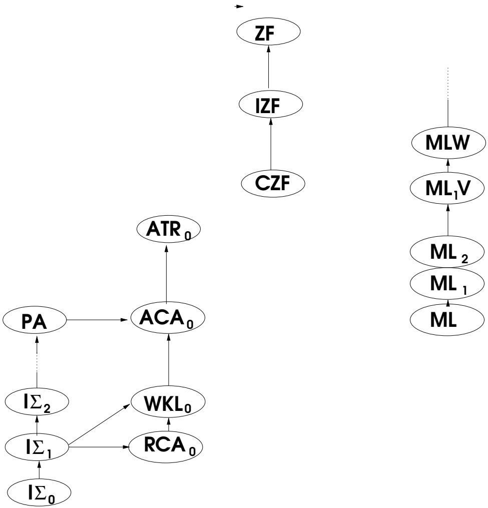
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.



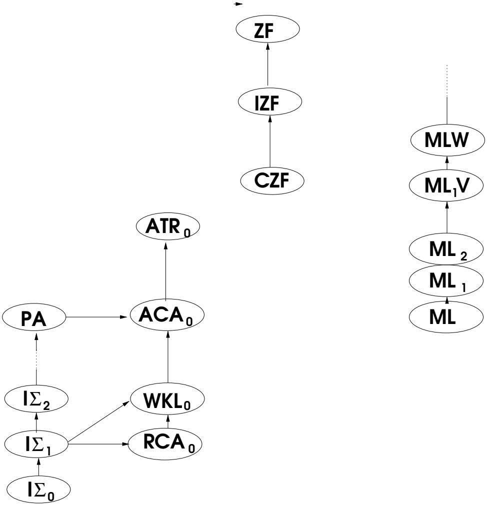
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.



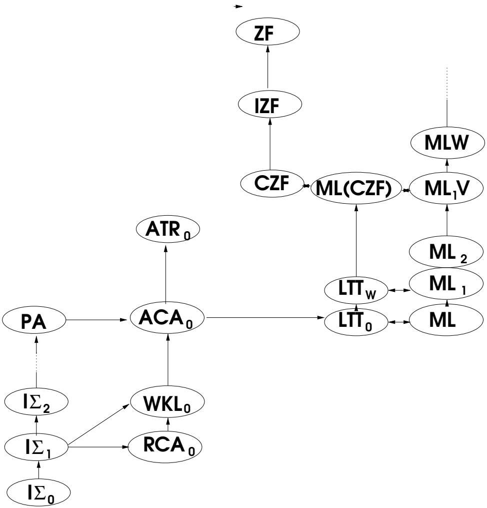
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
  - If propositions really were types, it should be easy.



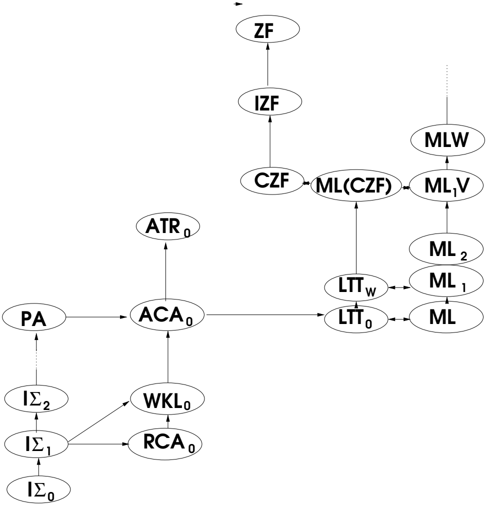
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and *semantics* are both very different.



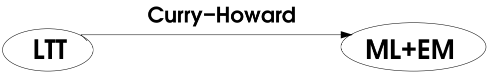
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and *semantics* are both very different.
- *Logic-enriched type theories* (*LTTs*) help.



Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and *semantics* are both very different.
- *Logic-enriched type theories* (*LTTs*) help.

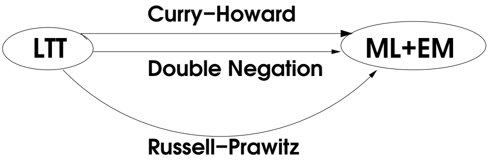


Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and *semantics* are both very different.
- *Logic-enriched type theories* (*LTTs*) help.
- Syntactic translations are possible.

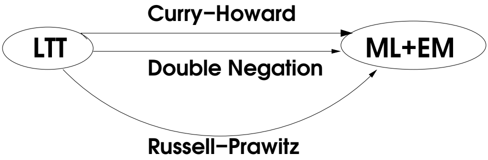
Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and *semantics* are both very different.
- *Logic-enriched type theories* (*LTTs*) help.
- Syntactic translations are possible.
- Curry-Howard becomes just one of a family.



Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and *semantics* are both very different.
- *Logic-enriched type theories* (*LTTs*) help.
- Syntactic translations are not enough.

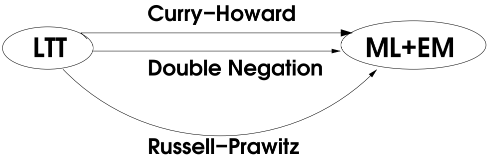


Introduction

- It is very difficult to translate between the systems on the left, and the systems on the right.
- Syntax and *semantics* are both very different.

- *Logic-enriched type theories* (*LTTs*) help.

- We need a semantics for LTTs.



1. Logic-Enriched Type Theories
   - Syntax

2. Categorical Semantics
   - Introduction to Categorical Semantics
   - Categorical Semantics for Logic-Enriched Type Theories
   - Soundness and Completeness Theorems

3. Applications
   - Conservativity of ACA_0 over PA
   - Bounded Quantification

Syntax of an LTT

LTT_0 is a system with:
Judgement forms:

$$\Gamma \vdash A \ \mathrm{Type} \quad \Gamma \vdash M : A$$
$$\Gamma \vdash \phi \ \mathrm{Prop} \quad \Gamma \vdash P : \phi$$

and associated equality judgements.

Syntax of an LTT

LTT_0 is a system with:

- arrow types $A \to B$
  with objects $\lambda x : A.M$

Syntax of an LTT

LTT_0 is a system with:

- arrow types $A \to B$
- product types $A \times B$
  with objects (M, M)

Syntax of an LTT

LTT_0 is a system with:

- arrow types $A \to B$
- product types $A \times B$
- natural numbers $\mathbb{N}$
  with objects 0 and $S(M)$

Syntax of an LTT

LTT_0 is a system with:

- arrow types $A \to B$
- product types $A \times B$
- natural numbers $\mathbb{N}$
- a type universe U
  with objects $\hat{\mathbb{N}}$ and $M \hat{\times} M$

Syntax of an LTT

LTT_0 is a system with:

- arrow types $A \to B$
- product types $A \times B$
- natural numbers $\mathbb{N}$
- a type universe U
- classical predicate logic
  with propositions $M =_A M$, $\neg\phi$, $\phi \wedge \psi$, $\forall x : A.\phi$, ...

Syntax of an LTT

LTT_0 is a system with:

- arrow types $A \to B$
- product types $A \times B$
- natural numbers $\mathbb{N}$
- a type universe U
- classical predicate logic
- a *propositional universe* prop Type

Syntax of an LTT

LTT_0 is a system with:

- arrow types $A \to B$
- product types $A \times B$
- natural numbers $\mathbb{N}$
- a type universe U
- classical predicate logic
- a *propositional universe* $\mathrm{prop}\ \mathrm{Type}$

We write $\mathrm{Set}\,(A)$ for $A \to \mathrm{prop}$.

The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe U contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}$, ...

The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe U contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}$, ...
- A type is *small* iff it has a name in U, *large* otherwise.

The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe U contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}$, ...
- A type is *small* iff it has a name in U, *large* otherwise.

The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe U contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}$, ...
- A type is *small* iff it has a name in U, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe prop contains $\hat{\neg} 0 \hat{=}_{\hat{\mathbb{N}}} S(0)$, $\hat{\forall} x : \hat{\mathbb{N}} . x \hat{=}_{\hat{\mathbb{N}}} x$, etc.

The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe U contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}$, ...
- A type is *small* iff it has a name in U, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe prop contains $\hat{\neg} 0 \hat{=}_{\hat{\mathbb{N}}} S(0)$, $\hat{\forall} x : \hat{\mathbb{N}}. x \hat{=}_{\hat{\mathbb{N}}} x$, etc.
- In LTT_0, prop contains the propositions that do not involve quantification over large types.

The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe U contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}$, ...
- A type is *small* iff it has a name in U, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe prop contains $\hat{\neg} 0 \hat{=}_{\hat{\mathbb{N}}} S(0)$, $\hat{\forall} x : \hat{\mathbb{N}}. x \hat{=}_{\hat{\mathbb{N}}} x$, etc.
- In LTT_0, prop contains the propositions that do not involve quantification over large types.
- A proposition is *small* iff it has a name in prop, *large* otherwise.

The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe U contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}$, ...
- A type is *small* iff it has a name in U, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe prop contains $\hat{\neg} 0 \hat{=}_{\hat{\mathbb{N}}} S(0)$, $\hat{\forall} x : \hat{\mathbb{N}}. x \hat{=}_{\hat{\mathbb{N}}} x$, etc.
- In LTT_0, prop contains the propositions that do not involve quantification over large types.
- A proposition is *small* iff it has a name in prop, *large* otherwise.

The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe U contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}$, ...
- A type is *small* iff it has a name in U, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe prop contains $\hat{\neg} 0 \hat{=}_{\hat{\mathbb{N}}} S(0)$, $\hat{\forall} x : \hat{\mathbb{N}}. x \hat{=}_{\hat{\mathbb{N}}} x$, etc.
- In LTT_0, prop contains the propositions that do not involve quantification over large types.
- A proposition is *small* iff it has a name in prop, *large* otherwise.

The strength of an LTT is determined by which types and which propositions we can *eliminate* over.

- We can only eliminate $\mathbb{N}$ over small types.

The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe U contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}}\hat{\times}\hat{\mathbb{N}}$, ...
- A type is *small* iff it has a name in U, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe prop contains $\hat{\neg}0\hat{=}_{\hat{\mathbb{N}}}S(0)$, $\hat{\forall}x : \hat{\mathbb{N}}.x\hat{=}_{\hat{\mathbb{N}}}x$, etc.
- In LTT_0, prop contains the propositions that do not involve quantification over large types.
- A proposition is *small* iff it has a name in prop, *large* otherwise.

The strength of an LTT is determined by which types and which propositions we can *eliminate* over.

- We can only eliminate $\mathbb{N}$ over small types.
- We can only use proof by induction with small propositions.

The Propositional Universe

A *type universe* is a type whose objects are names of types:

- The universe U contains $\hat{\mathbb{N}}$, $\hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}$, ...
- A type is *small* iff it has a name in U, *large* otherwise.

A *propositional universe* is a type whose objects are names of propositions:

- The universe prop contains $\hat{\neg} 0 \hat{=}_{\hat{\mathbb{N}}} S(0)$, $\hat{\forall} x : \hat{\mathbb{N}}. x \hat{=}_{\hat{\mathbb{N}}} x$, etc.
- In LTT_0, prop contains the propositions that do not involve quantification over large types.
- A proposition is *small* iff it has a name in prop, *large* otherwise.

The strength of an LTT is determined by which types and which propositions we can *eliminate* over.

- We can only eliminate $\mathbb{N}$ over small types.
- We can only use proof by induction with small propositions.
- Adding a new type or connective is conservative. Adding it to the universes is not.

Categorical Semantics

We can give semantics to a type theory in a variety of ways:

Map types to sets, ω-sets, PERs, sheaves, domains, ...

To save repeating work, we:

- define the properties a category must have for us to build a semantics from its objects;
- give semantics to the theory in an *arbitrary* category with those properties.

Categorical Semantics for a Dependent Type Theory

To give semantics to a dependent type theory, we need:

- a category $\mathbb{B}$ (whose objects interpret *contexts* Γ);

$\mathbb{B}$

Categorical Semantics for a Dependent Type Theory

To give semantics to a dependent type theory, we need:

- a category $\mathbb{B}$ (whose objects interpret *contexts* Γ);
- a category $\mathbb{E}$ (whose objects interpret *types-in-context* $\Gamma \vdash A \ \mathrm{Type}$);

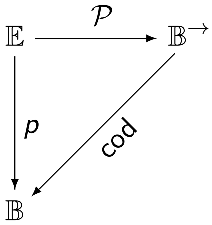
$\mathbb{E}$

$\mathbb{B}$

Categorical Semantics for a Dependent Type Theory

To give semantics to a dependent type theory, we need:

- a category $\mathbb{B}$ (whose objects interpret *contexts* Γ);
- a category $\mathbb{E}$ (whose objects interpret *types-in-context* $\Gamma \vdash A \ \mathrm{Type}$);
- a functor $\mathcal{P} : \mathbb{E} \to \mathbb{B}^{\to}$ (mapping $\Gamma \vdash A$ to $(\Gamma, x : A) \to \Gamma$).



Categorical Semantics for a Dependent Type Theory

To give semantics to a dependent type theory, we need:

- a category $\mathbb{B}$ (whose objects interpret *contexts* Γ);
- a category $\mathbb{E}$ (whose objects interpret *types-in-context* $\Gamma \vdash A\ \mathrm{Type}$);
- a functor $\mathcal{P} : \mathbb{E} \to \mathbb{B}^{\to}$ (mapping $\Gamma \vdash A$ to $(\Gamma, x : A) \to \Gamma$).

such that

- $p = \mathrm{cod} \circ \mathcal{P}$ is a fibration
- $\mathbb{B}$ has a terminal object

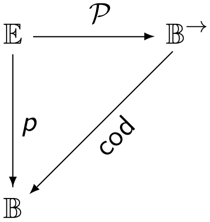
$$\begin{array}{ccc} \mathbb{E} & \xrightarrow{\mathcal{P}} & \mathbb{B}^{\to} \\ {\scriptstyle p}\downarrow & \swarrow_{\mathrm{cod}} & \\ \mathbb{B} & & \end{array}$$

Categorical Semantics for a LTT

To give semantics to a LTT,
we need, in addition:

- a category $\mathbb{P}$ (whose objects interpret *propositions-in-context* $\Gamma \vdash \phi$ Prop);

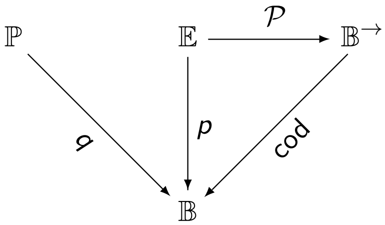
$\mathbb{P}$



Categorical Semantics for a LTT

To give semantics to a LTT, we need, in addition:

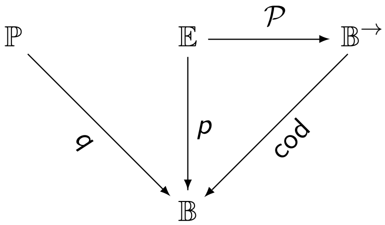
- a category $\mathbb{P}$ (whose objects interpret *propositions-in-context* $\Gamma \vdash \phi \ \mathrm{Prop}$);
- a fibration $q : \mathbb{P} \to \mathbb{B}$ (mapping $\Gamma \vdash \phi \ \mathrm{Prop}$ to Γ)



Categorical Semantics for a LTT

To give semantics to a LTT, we need, in addition:

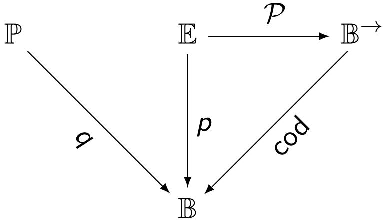
- a category $\mathbb{P}$ (whose objects interpret *propositions-in-context* $\Gamma \vdash \phi \ \mathrm{Prop}$);
- a fibration $q : \mathbb{P} \to \mathbb{B}$ (mapping $\Gamma \vdash \phi \ \mathrm{Prop}$ to Γ)
- for every object $\Gamma \vdash A \ \mathrm{Type}$ in $\mathbb{E}$, a right adjoint $\pi^* \dashv \forall$ and a left adjoint $\exists \dashv \pi^*$



Categorical Semantics for a LTT

To give semantics to a LTT, we need, in addition:

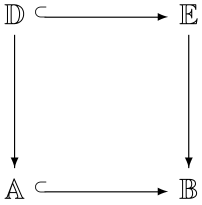
- a category $\mathbb{P}$ (whose objects interpret *propositions-in-context* $\Gamma \vdash \phi\ \mathrm{Prop}$);
- a fibration $q : \mathbb{P} \to \mathbb{B}$ (mapping $\Gamma \vdash \phi\ \mathrm{Prop}$ to Γ)
- for every object $\Gamma \vdash A\ \mathrm{Type}$ in $\mathbb{E}$, a right adjoint $\pi^* \dashv \forall$ and a left adjoint $\exists \dashv \pi^*$
- such that $\mathbb{P}$ is a locally Cartesian closed category.



Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe (U, T), we need:

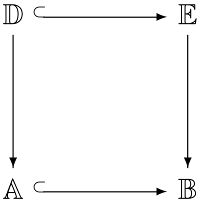
- a substructure (intended to represent the small types and small contexts)



Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe (U, T), we need:

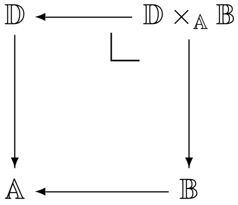
- a substructure (intended to represent the small types and small contexts)
- an object U in $\mathbb{E}$ over the empty context (terminal object)



Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe (U, T), we need:

- a substructure (intended to represent the small types and small contexts)
- an object U in $\mathbb{E}$ over the empty context (terminal object)
- a generic object $x : U \vdash T(x)\ \mathrm{Type}$ in $\mathbb{E}$ over $\mathrm{dom}\,\mathcal{P}(\vdash U\ \mathrm{Type})$;



Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe (U, T), we need:

- a substructure (intended to represent the small types and small contexts)
- an object U in $\mathbb{E}$ over the empty context (terminal object)
- a generic object $x : U \vdash T(x)\ \mathrm{Type}$ in $\mathbb{E}$ over $\mathrm{dom}\,\mathcal{P}(\vdash U\ \mathrm{Type})$;

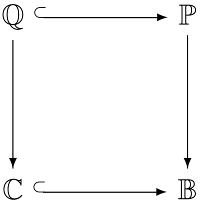
Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe (U, T), we need:

- a substructure (intended to represent the small types and small contexts)
- an object U in $\mathbb{E}$ over the empty context (terminal object)
- a generic object $x : U \vdash T(x)\ \mathrm{Type}$ in $\mathbb{E}$ over dom $\mathcal{P}(\vdash U\ \mathrm{Type})$;

To give semantics to (prop, V), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)



Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe (U, T), we need:

- a substructure (intended to represent the small types and small contexts)
- an object U in $\mathbb{E}$ over the empty context (terminal object)
- a generic object $x : U \vdash T(x)\ \mathrm{Type}$ in $\mathbb{E}$ over $\mathrm{dom}\,\mathcal{P}(\vdash U\ \mathrm{Type})$;

To give semantics to (prop, V), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)
- an object prop in $\mathbb{E}$ over the terminal object

$$\begin{array}{ccc} \mathbb{Q} & \hookrightarrow & \mathbb{P} \\ \downarrow & & \downarrow \\ \mathbb{C} & \hookrightarrow & \mathbb{B} \end{array}$$

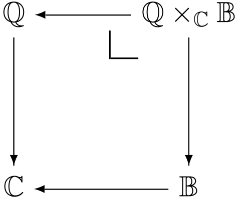
Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe (U, T), we need:

- a substructure (intended to represent the small types and small contexts)
- an object U in $\mathbb{E}$ over the empty context (terminal object)
- a generic object $x : U \vdash T(x)\ \mathrm{Type}$ in $\mathbb{E}$ over dom $\mathcal{P}(\vdash U\ \mathrm{Type})$;

To give semantics to (prop, V), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)
- an object prop in $\mathbb{E}$ over the terminal object
- a generic object V in $\mathbb{E}$ over dom $\mathcal{P}\mathrm{prop}$.



Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe (U, T), we need:

- a substructure (intended to represent the small types and small contexts)
- an object U in $\mathbb{E}$ over the empty context (terminal object)
- a generic object $x : U \vdash T(x)\ \mathrm{Type}$ in $\mathbb{E}$ over $\mathrm{dom}\,\mathcal{P}(\vdash U\ \mathrm{Type})$;

To give semantics to (prop, V), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)
- an object prop in $\mathbb{E}$ over the terminal object
- a generic object V in $\mathbb{E}$ over $\mathrm{dom}\,\mathcal{P}\mathrm{prop}$.

Categorical Semantics for Universes

To give semantics to a dependent type theory with a universe (U, T), we need:

- a substructure (intended to represent the small types and small contexts)
- an object U in $\mathbb{E}$ over the empty context (terminal object)
- a generic object $x : U \vdash T(x)\ \mathrm{Type}$ in $\mathbb{E}$ over $\mathrm{dom}\,\mathcal{P}(\vdash U\ \mathrm{Type})$;

To give semantics to (prop, V), we need in addition:

- a substructure (intended to represent the small propositions and contexts consisting solely of small propositions)
- an object prop in $\mathbb{E}$ over the terminal object
- a generic object V in $\mathbb{E}$ over $\mathrm{dom}\,\mathcal{P}\mathrm{prop}$.

We require $\top \to \langle x : \mathbb{N} \rangle \to \langle x : \mathbb{N} \rangle$ to be a weak fibred natural number object in both of these right-hand-sides.

Interpretation

Given an $\mathrm{LTT_W}$-category $\mathcal{C}$, define:

- for every valid context Γ, an object $[\![\Gamma]\!]$ of $\mathbb{B}$;
- for every type A such that $\Gamma \vdash A \ \mathrm{Type}$, an object $[\![\Gamma \vdash A]\!]$ of $\mathbb{E}$ such that $p\,[\![\Gamma \vdash A]\!] = [\![\Gamma]\!]$
- for every term M such that $\Gamma \vdash M : A$, an arrow $[\![\Gamma \vdash M]\!] : [\![\Gamma]\!] \to \mathrm{dom}\,\mathcal{P}\,[\![\Gamma \vdash A]\!]$
- for every proposition ϕ such that $\Gamma \vdash \phi \mathrm{prop}$, an object $[\![\Gamma \vdash \phi]\!]$ of $\mathbb{P}$ over $[\![\Gamma]\!]$

Soundness Theorem

Theorem

Every judgement is true in any $\mathrm{LTT_W}$*-category. That is:*

1. *If* $\Gamma \vdash A = B$ *then* $[\![\Gamma \vdash A]\!] = [\![\Gamma \vdash B]\!]$
2. *If* $\Gamma \vdash M = N : A$ *then* $[\![\Gamma \vdash M]\!] = [\![\Gamma \vdash N]\!]$
3. *If* $\Gamma \vdash \phi = \psi$ *then* $[\![\Gamma \vdash \phi]\!] = [\![\Gamma \vdash \psi]\!]$
4. *If there is a proof* $\Gamma \vdash P : \phi$ *then there is a vertical arrow* $\top \to [\![\Gamma \vdash \phi]\!]$ *in the fibre* $\mathbb{P}/ [\![\Gamma]\!]$.

Proof.

Induction on derivations. □

Completeness Theorem

Theorem

If a judgement is true in every category $\mathcal{C}$, then it is derivable in T.

Proof.

Define the category $\mathrm{Cl}(T)$, the *classifying category* of T, thus:

- the objects of $\mathbb{B}$ are the valid contexts;
- the objects of $\mathbb{E}$ are the pairs (Γ, A) such that $\Gamma \vdash A\ \mathrm{Type}$, quotiented by equality;
- ...

If a judgement is true in $\mathrm{Cl}(T)$, then it is derivable in T. □

Completeness Theorem

Theorem

If a judgement is true in every category $\mathcal{C}$, then it is derivable in T.

Proof.

Define the category $\mathrm{Cl}(T)$, the *classifying category* of T, thus:

- the objects of $\mathbb{B}$ are the valid contexts;
- the objects of $\mathbb{E}$ are the pairs (Γ, A) such that $\Gamma \vdash A \ \mathrm{Type}$, quotiented by equality;
- ...

If a judgement is true in $\mathrm{Cl}(T)$, then it is derivable in T. □

In fact, $\mathrm{Cl}(T)$ is an initial object in the metacategory of $\mathrm{LTT_W}$-categories. The interpretation given earlier is the unique functor $\mathrm{Cl}(T) \to \mathbb{C}$.

Completeness Theorem

Theorem

If a judgement is true in every category $\mathcal{C}$, then it is derivable in T.

Proof.

Define the category $\mathrm{Cl}(T)$, the *classifying category* of T, thus:

- the objects of $\mathbb{B}$ are the valid contexts;
- the objects of $\mathbb{E}$ are the pairs (Γ, A) such that $\Gamma \vdash A\ \mathrm{Type}$, quotiented by equality;
- ...

If a judgement is true in $\mathrm{Cl}(T)$, then it is derivable in T. □

In fact, $\mathrm{Cl}(T)$ is an initial object in the metacategory of $\mathrm{LTT_W}$-categories. The interpretation given earlier is the unique functor $\mathrm{Cl}(T) \to \mathbb{C}$. This is the sort of thing that gets category theorists excited.

Conservativity of LTT_0 over PA

I have previously given *syntactic* proofs that LTT_0 is conservative over PA. We can now give a *semantic* proof of the same result.

Theorem

LTT_0 *is conservative over* PA.

Proof.

From any model $\mathcal{M}$ of PA, we construct a model of LTT_0.

Conservativity of LTT_0 over PA

I have previously given *syntactic* proofs that LTT_0 is conservative over PA. We can now give a *semantic* proof of the same result.

Theorem

LTT_0 *is conservative over* PA.

Proof.

From any model $\mathcal{M}$ of PA, we construct a model of LTT_0.
Define the *higher-order recursive* (hor) functions to be those built up from $0^{\mathcal{M}}$ and $S^{\mathcal{M}}$ by composition, primitive recursion, pairing, projection, lambda-abstraction and application.

Conservativity of LTT_0 over PA

I have previously given *syntactic* proofs that LTT_0 is conservative over PA. We can now give a *semantic* proof of the same result.

Theorem

LTT_0 *is conservative over* PA.

Proof.

From any model $\mathcal{M}$ of PA, we construct a model of LTT_0.
Define the *higher-order recursive* (hor) functions to be those built up from $0^{\mathcal{M}}$ and $S^{\mathcal{M}}$ by composition, primitive recursion, pairing, projection, lambda-abstraction and application.
Define the *arithmetic* predicates to be those built up from equality by Boolean operations and quantification over $|\mathcal{M}|$.

Conservativity of LTT_0 over PA

I have previously given *syntactic* proofs that LTT_0 is conservative over PA. We can now give a *semantic* proof of the same result.

Theorem

LTT_0 *is conservative over* PA.

Proof.

From any model $\mathcal{M}$ of PA, we construct a model of LTT_0.
The objects of $\mathbb{E}$ are the sets built up from $|\mathcal{M}|$ by $\times$, $\rightarrow$ and P,
where $A \rightarrow B$ is the set of hor functions from A to B,
and PA is the set of arithmetic subsets of A.

Conservativity of LTT_0 over PA

I have previously given *syntactic* proofs that LTT_0 is conservative over PA. We can now give a *semantic* proof of the same result.

Theorem

LTT_0 *is conservative over* PA*.*

Proof.

From any model $\mathcal{M}$ of PA, we construct a model of LTT_0.
The objects of $\mathbb{E}$ are the sets built up from $|\mathcal{M}|$ by $\times$, $\rightarrow$ and P.
The objects of $\mathbb{B}$ are the sets of all sequences of objects of $\mathbb{E}$. The arrows are the hor functions.

Conservativity of LTT_0 over PA

I have previously given *syntactic* proofs that LTT_0 is conservative over PA. We can now give a *semantic* proof of the same result.

Theorem

LTT_0 *is conservative over* PA.

Proof.

From any model $\mathcal{M}$ of PA, we construct a model of LTT_0.
The objects of $\mathbb{E}$ are the sets built up from $|\mathcal{M}|$ by $\times$, $\rightarrow$ and P.
The objects of $\mathbb{B}$ are the sets of all sequences of objects of $\mathbb{E}$. The arrows are the hor functions.
The objects of $\mathbb{P}$ over $b \in \mathbb{B}$ are all subsets of b. □

Note that $\mathbb{E}$ and $\mathbb{P}$ are radically different.

Conservativity of LTT_0 over PA

I have previously given *syntactic* proofs that LTT_0 is conservative over PA. We can now give a *semantic* proof of the same result.

Theorem

LTT_0 *is conservative over* PA.

Proof.

From any model $\mathcal{M}$ of PA, we construct a model of LTT_0.
The objects of $\mathbb{E}$ are the sets built up from $|\mathcal{M}|$ by $\times$, $\to$ and P.
The objects of $\mathbb{B}$ are the sets of all sequences of objects of $\mathbb{E}$. The arrows are the hor functions.
The objects of $\mathbb{P}$ over $b \in \mathbb{B}$ are all subsets of b. □

We can similarly prove LTT_0 conservative over ACA_0.

Corollary

ACA_0 *is conservative over* PA.

Bounded Quantification

Problem: How do we turn prop into the set of Σ_0-propositions?

Bounded Quantification

Problem: How do we turn prop into the set of Σ_0-propositions?
Just close it under bounded quantification? Categorical semantics are horrible.

Bounded Quantification

Problem: How do we turn prop into the set of Σ_0-propositions?
Just close it under bounded quantification? Categorical semantics are horrible.
Answer: Put (a name of) prop in U.

Bounded Quantification

Problem: How do we turn prop into the set of Σ_0-propositions?
Just close it under bounded quantification? Categorical semantics are horrible.
Answer: Put (a name of) prop in U.
We can define bounded quantification by elimination $\mathbb{N}$ over prop:

$$\begin{aligned} \forall x < 0.\phi(x) &= \top \\ \forall x < S(n).\phi(x) &= \forall x < n.\phi(x) \wedge \phi(n) \end{aligned}$$

Bounded Quantification

Problem: How do we turn prop into the set of Σ_0-propositions?
Just close it under bounded quantification? Categorical semantics are horrible.
Answer: Put (a name of) prop in U.
We can define bounded quantification by elimination $\mathbb{N}$ over prop:

$$\begin{aligned} \forall x < 0.\phi(x) &= \top \\ \forall x < S(n).\phi(x) &= \forall x < n.\phi(x) \wedge \phi(n) \end{aligned}$$

Conversely, any formula in prop in the new LTT corresponds to a Σ_0-formula in $I\Sigma_0(exp)$.

Bounded Quantification

Problem: How do we turn prop into the set of Σ_0-propositions?
Just close it under bounded quantification? Categorical semantics are horrible.
Answer: Put (a name of) prop in U.
We can define bounded quantification by elimination $\mathbb{N}$ over prop:

$$\begin{aligned} \forall x < 0.\phi(x) &= \top \\ \forall x < S(n).\phi(x) &= \forall x < n.\phi(x) \wedge \phi(n) \end{aligned}$$

Conversely, any formula in prop in the new LTT corresponds to a Σ_0-formula in $I\Sigma_0(exp)$.
(Show that the functions in $\mathbb{N} \to \mathbb{N}$ are all defined by a Σ_0-formula in $I\Sigma_0(exp)$. Use the fact that the Σ_0-definable functions are closed under primitive recursion.)

Conclusion

Logic-enriched type theories are the right setting for investigating many foundational questions in type theory, and in orthodox logic.

Conclusion

Logic-enriched type theories are the right setting for investigating many foundational questions in type theory, and in orthodox logic. At the moment, they are a solution looking for a problem.

I don't want to carry on finding new proofs of old results.

Conclusion

Logic-enriched type theories are the right setting for investigating many foundational questions in type theory, and in orthodox logic. At the moment, they are a solution looking for a problem.

I don't want to carry on finding new proofs of old results.

Questions I plan to investigate:

- What is the proof-theoretic ordinal of this LTT?
- What is the set of functions definable in this LTT?
- Some logical features work nicely in LTTs (Σ_0-induction, Σ_1-induction)
- Some do not (Σ_2-induction)
- What's the difference between these?

Conclusion

Logic-enriched type theories are the right setting for investigating many foundational questions in type theory, and in orthodox logic. At the moment, they are a solution looking for a problem.

I don't want to carry on finding new proofs of old results.

Questions I plan to investigate:

- What is the proof-theoretic ordinal of this LTT?
- What is the set of functions definable in this LTT?
- Some logical features work nicely in LTTs (Σ_0-induction, Σ_1-induction)
- Some do not (Σ_2-induction)
- What's the difference between these?

Please bring me some more.

Syntax of an LTT

LTT_0 is a system with:

Judgement forms:

$$\begin{array}{ll} \Gamma \vdash A \text{ type} & \Gamma \vdash M : A \\ \Gamma \vdash \phi \text{ Prop} & \Gamma \vdash P : \phi \end{array}$$

and associated equality judgements.

Type	A	$::=$	
Term	M	$::=$	x
Proposition	ϕ	$::=$	
Proof	P	$::=$	

Syntax of an LTT

LTT_0 is a system with:

- arrow types

$$
\begin{array}{lrcl}
\text{Type} & A & ::= & A \to A \\
\text{Term} & M & ::= & x \mid \lambda x : A.M \mid MM \\
\\
\text{Proposition} & \phi & ::= & \\
\\
\text{Proof} & P & ::= &
\end{array}
$$

Syntax of an LTT

LTT_0 is a system with:

- arrow types
- product types

$$
\begin{array}{lrcl}
\text{Type} & A & ::= & A \to A \mid A \times A \\
\text{Term} & M & ::= & x \mid \lambda x : A.M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M)
\end{array}
$$

$$
\begin{array}{lrc}
\text{Proposition} & \phi & ::=
\end{array}
$$

$$
\begin{array}{lrc}
\text{Proof} & P & ::=
\end{array}
$$

Syntax of an LTT

LTT_0 is a system with:

- arrow types
- product types
- natural numbers

$$
\begin{array}{llcl}
\text{Type} & A & ::= & A \to A \mid A \times A \mid \mathbb{N} \\
\text{Term} & M & ::= & x \mid \lambda x : A.M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid \\
 & & & 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \\
\text{Proposition} & \phi & ::= & \\
\text{Proof} & P & ::= &
\end{array}
$$

Syntax of an LTT

LTT_0 is a system with:

- arrow types
- product types
- natural numbers
- a type universe closed under $\mathbb{N}$ and $\times$

$$
\begin{array}{llll}
\text{Type} & A & ::= & A \to A \mid A \times A \mid \mathbb{N} \mid U \mid T(M) \\
\text{Term} & M & ::= & x \mid \lambda x : A.M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid \\
 & & & 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \mid \hat{\mathbb{N}} \mid M \hat{\times} M \mid \\
 & & & \\
\text{Proposition} & \phi & ::= & \\
 & & & \\
\text{Proof} & P & ::= &
\end{array}
$$

Syntax of an LTT

LTT_0 is a system with:

- arrow types
- product types
- natural numbers
- a type universe closed under $\mathbb{N}$ and $\times$
- classical predicate logic

$$
\begin{array}{llll}
\text{Type} & A & ::= & A \to A \mid A \times A \mid \mathbb{N} \mid U \mid T(M) \\
\text{Term} & M & ::= & x \mid \lambda x : A.M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid \\
 & & & 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \mid \hat{\mathbb{N}} \mid M \hat{\times} M \mid \\
 & & & \\
\text{Proposition} & \phi & ::= & M =_A M \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \to \phi \mid \\
 & & & \forall x : A.\phi \mid \exists x : A.\phi \\
\text{Proof} & P & ::= & \cdots
\end{array}
$$

Syntax of an LTT

LTT_0 is a system with:

- arrow types
- product types
- natural numbers
- a type universe closed under $\mathbb{N}$ and $\times$
- classical predicate logic
- a *propositional universe*

$$
\begin{array}{lrcl}
\text{Type} & A & ::= & A \to A \mid A \times A \mid \mathbb{N} \mid U \mid T(M) \\
\text{Term} & M & ::= & x \mid \lambda x : A.M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid \\
& & & 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \mid \hat{\mathbb{N}} \mid M \hat{\times} M \mid \\
\\
\text{Proposition} & \phi & ::= & M =_A M \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \to \phi \mid \\
& & & \forall x : A.\phi \mid \exists x : A.\phi \mid \mathrm{prop} \mid V(P) \\
\text{Proof} & P & ::= & \cdots \mid M \hat{=}_M M \mid \hat{\neg}\phi \mid \phi \hat{\wedge} \phi \mid \cdots
\end{array}
$$

Syntax of an LTT

LTT_0 is a system with:

- arrow types
- product types
- natural numbers
- a type universe closed under $\mathbb{N}$ and $\times$
- classical predicate logic
- a *propositional universe*
- *typed sets*

$$\begin{array}{llcl}
\text{Type} & A & ::= & A \to A \mid A \times A \mid \mathbb{N} \mid U \mid T(M) \mid \mathrm{Set}\,(A) \\
\text{Term} & M & ::= & x \mid \lambda x : A.M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid \\
 & & & 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \mid \hat{\mathbb{N}} \mid M \hat{\times} M \mid \\
 & & & \{x : A \mid P\} \\
\text{Proposition} & \phi & ::= & M =_A M \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \to \phi \mid \\
 & & & \forall x : A.\phi \mid \exists x : A.\phi \mid \mathrm{prop} \mid V(P) \\
\text{Proof} & P & ::= & \cdots \mid M \hat{=}_M M \mid \hat{\neg}\phi \mid \phi \hat{\wedge} \phi \mid \cdots \mid M \in M
\end{array}$$

We can give a semantic proof of this result:

A function $f : \mathbb{N}^n \to \mathbb{N}$ is *definable* in *PA* iff there is a formula $\phi[x_1, \ldots, x_n, y]$ such that:

1. for all a_1, …, a_n, $\mathrm{PA} \vdash \phi[\overline{a_1}, \ldots, \overline{a_n}, \overline{f(a_1, \ldots, a_n)}]$;
2. $\mathrm{PA} \vdash \forall x_1 \cdots \forall x_n \exists! y \phi[x_1, \ldots, x_n, y]$

Theorem

The functions definable in PA *are exactly the* ϵ_0*-recursive functions.*

Proof.

Construct a model of LTT_0 in which the arrows are the ϵ_0-recursive functions. Then apply conservativity. □

History of LTTs

2002 Aczel and Gambino [?] define translations between Constructive ZF (CZF) and the type theory $\mathrm{ML}_1\mathrm{V}$.

History of LTTs

2002 Aczel and Gambino [?] define translations between Constructive ZF (CZF) and the type theory $\mathrm{ML_1V}$.

2006 Gambino and Aczel [?] introduce the logic-enriched type theory $\mathrm{ML(CZF)}$ as a half-way stage.

History of LTTs

2002 Aczel and Gambino [?] define translations between Constructive ZF (CZF) and the type theory $\mathrm{ML_1V}$.

2006 Gambino and Aczel [?] introduce the logic-enriched type theory $\mathrm{ML(CZF)}$ as a half-way stage.

2007 Adams and Luo [?] show how an LTT can represent Weyl's school of predicativism.

History of LTTs

2002 Aczel and Gambino [?] define translations between Constructive ZF (CZF) and the type theory $\mathrm{ML_1V}$.

2006 Gambino and Aczel [?] introduce the logic-enriched type theory $\mathrm{ML(CZF)}$ as a half-way stage.

2007 Adams and Luo [?] show how an LTT can represent Weyl's school of predicativism.

2010 Adams and Luo [?] show their system is *not* conservative over PA.

The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.

The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.

The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;

The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- . . . which becomes just one of a family of translations.

The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- . . . which becomes just one of a family of translations.
- But I need semantics to guide future research.

The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- . . . which becomes just one of a family of translations.
- But I need semantics to guide future research.
- I also need to think of better names.

The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- . . . which becomes just one of a family of translations.
- But I need semantics to guide future research.
- I also need to think of better names.

The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- . . . which becomes just one of a family of translations.
- But I need semantics to guide future research.
- I also need to think of better names.

I do *not* claim:

- LTTs are 'better' than predicate logics or type theories.

The Moral of the Story

From this work, I take the message:

- LTTs can do *some* things better than either orthodox logics or type theories.
- LTTs are very useful as an intermediary between orthodox logics and type theories.
- LTTs turn Curry-Howard into a mathematical *object* — a translation from an LTT to a type theory;
- . . . which becomes just one of a family of translations.
- But I need semantics to guide future research.
- I also need to think of better names.

I do *not* claim:

- LTTs are 'better' than predicate logics or type theories.
- Curry-Howard is 'bad'