

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 implication 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 implication 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

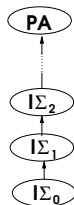
I believe:

- 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:

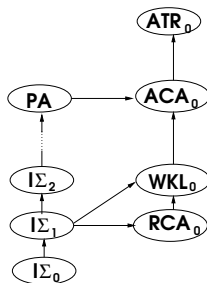
- systems of first-order arithmetic



Introduction

We have:

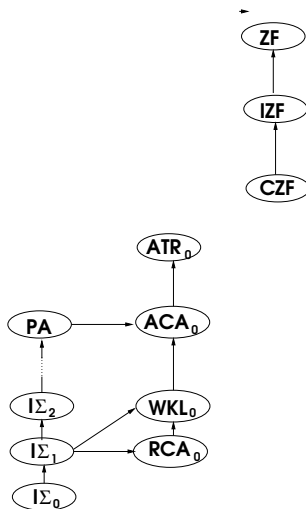
- systems of first-order arithmetic
- systems of second-order arithmetic



Introduction

We have:

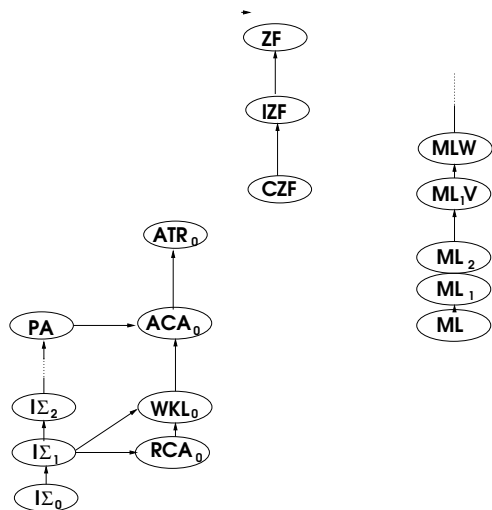
- systems of first-order arithmetic
- systems of second-order arithmetic
- set theories



Introduction

We have:

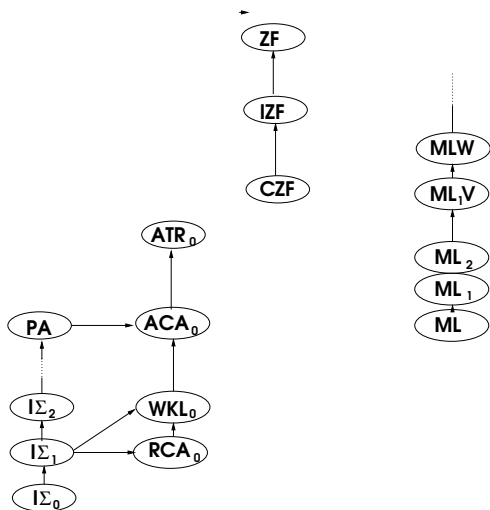
- systems of first-order arithmetic
- systems of second-order arithmetic
- set theories
- type theories



Introduction

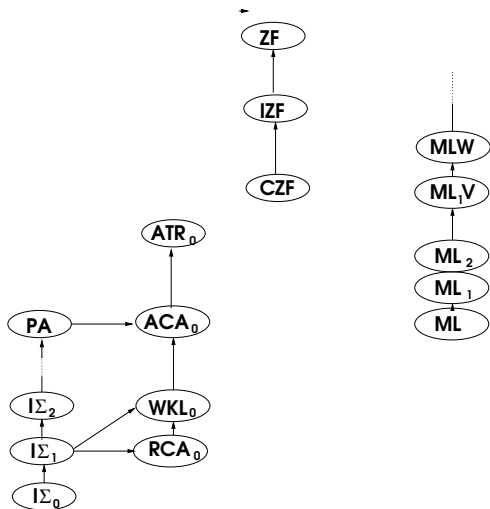
We have:

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



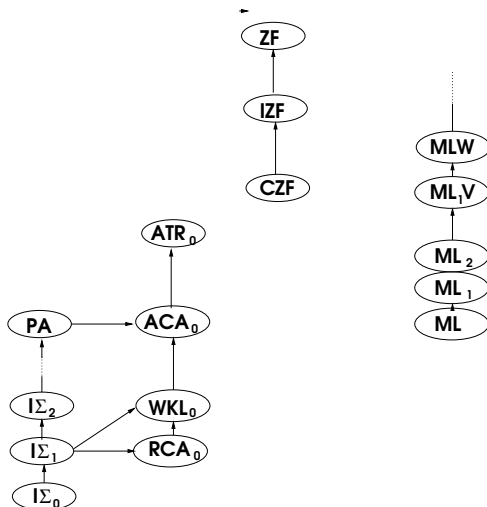
Introduction

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



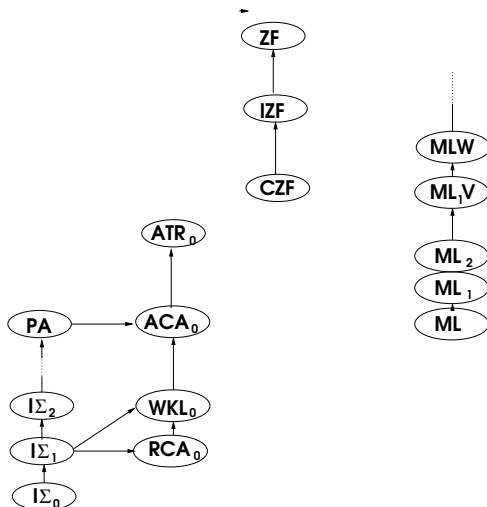
Introduction

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



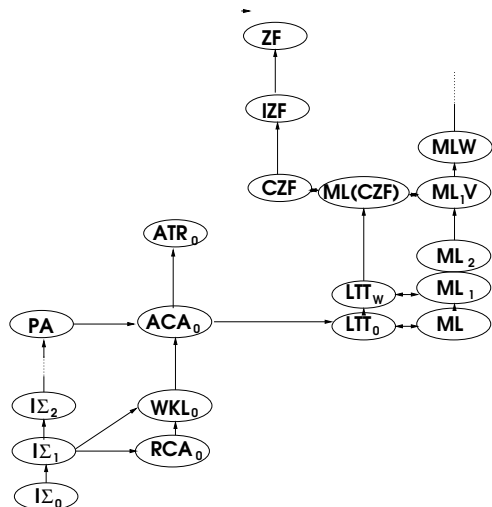
Introduction

- 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.



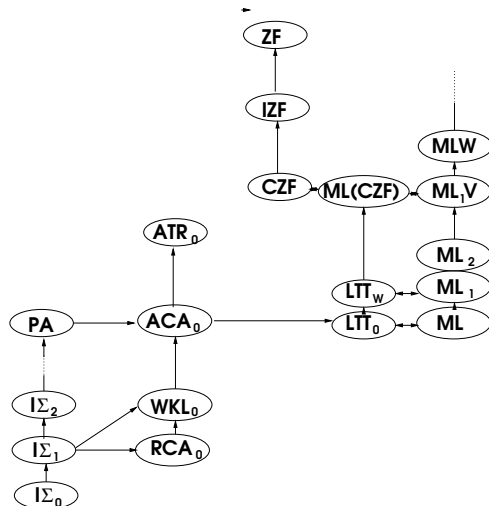
Introduction

- 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.



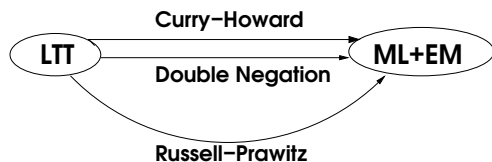
Introduction

- 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.



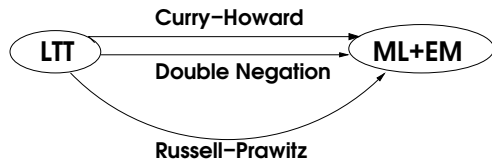
Introduction

- 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.



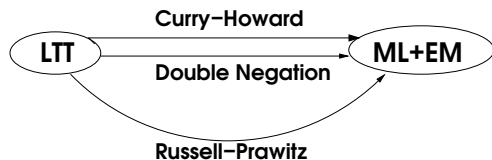
Introduction

- 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

LTT₀ is a system with:

Judgement forms:

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

and associated equality judgements.

Syntax of an LTT

LTT₀ is a system with:

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

Syntax of an LTT

LTT_0 is a system with:

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

Syntax of an LTT

LTT_0 is a system with:

- arrow types $A \rightarrow B$
- 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 $A \rightarrow B$
- 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

LTT₀ is a system with:

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

Syntax of an LTT

LTT_0 is a system with:

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

We write $\text{Set}(A)$ for $A \rightarrow \text{prop}$.

The Propositional Universe

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

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

The Propositional Universe

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

- The universe U contains \hat{N} , $\hat{N} \hat{\times} \hat{N}$, \dots
- 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{N} , $\hat{N} \hat{\times} \hat{N}$, \dots
- 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{N} , $\hat{N} \hat{\times} \hat{N}$, \dots
- 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 $\hat{0} \hat{=}_{\hat{N}} S(0)$, $\hat{\forall} x : \hat{N}. x \hat{=}_{\hat{N}} x$, etc.

The Propositional Universe

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

- The universe U contains \hat{N} , $\hat{N} \hat{\times} \hat{N}$, \dots
- 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 $\hat{0} \hat{=} \hat{N} S(0)$, $\hat{\forall} x : \hat{N}. x \hat{=} \hat{N} x$, 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 $\hat{N}, \hat{N} \hat{\times} \hat{N}, \dots$
- 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 $\hat{0} \hat{=} \hat{N} S(0), \hat{\forall} x : \hat{N}. x \hat{=} \hat{N} x$, 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 $\hat{N}, \hat{N} \hat{\times} \hat{N}, \dots$
- 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 $\hat{0} \hat{=}_{\hat{N}} S(0), \hat{\forall} x : \hat{N}. x \hat{=}_{\hat{N}} x$, 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 $\hat{\mathbb{N}}, \hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}, \dots$
- 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 $\hat{0} \hat{=}_{\hat{\mathbb{N}}} S(0), \hat{\forall} x : \hat{\mathbb{N}}. x \hat{=}_{\hat{\mathbb{N}}} x$, 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 $\hat{\mathbb{N}}, \hat{\mathbb{N}} \hat{\times} \hat{\mathbb{N}}, \dots$
- 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 $\hat{0} \hat{=}_{\hat{\mathbb{N}}} S(0), \hat{\forall} x : \hat{\mathbb{N}}. x \hat{=}_{\hat{\mathbb{N}}} x$, 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 $\hat{\mathbb{N}}, \hat{\mathbb{N}} \times \hat{\mathbb{N}}, \dots$
- 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 $\hat{0} \hat{=}_{\hat{\mathbb{N}}} S(0), \hat{\forall} x : \hat{\mathbb{N}}. x \hat{=}_{\hat{\mathbb{N}}} x$, 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* $\Gamma \vdash A$ Type);

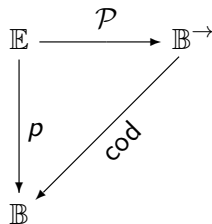
\mathbb{E}

\mathbb{B}

Categorical Semantics for a Dependent Type Theory

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

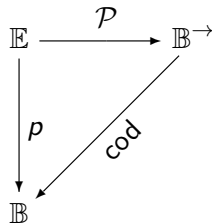
- a category \mathbb{B} (whose objects interpret *contexts* Γ);
- a category \mathbb{E} (whose objects interpret *types-in-context* $\Gamma \vdash A \text{ Type}$);
- a functor $\mathcal{P} : \mathbb{E} \rightarrow \mathbb{B}^{\rightarrow}$ (mapping $\Gamma \vdash A$ to $(\Gamma, x : A) \rightarrow \Gamma$).



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* $\Gamma \vdash A \text{ Type}$);
- a functor $\mathcal{P} : \mathbb{E} \rightarrow \mathbb{B}^{\rightarrow}$ (mapping $\Gamma \vdash A$ to $(\Gamma, x : A) \rightarrow \Gamma$).



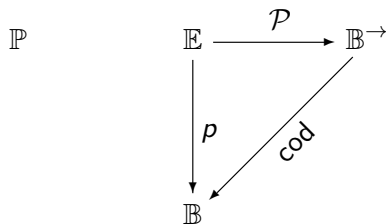
such that

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

Categorical Semantics for a LTT

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

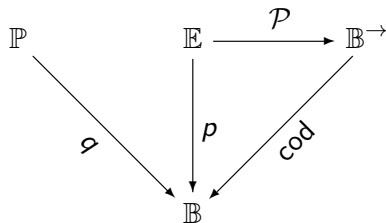
- a category \mathbb{P} (whose objects interpret *propositions-in-context* $\Gamma \vdash \phi \text{ Prop}$);



Categorical Semantics for a LTT

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

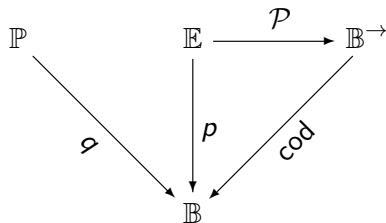
- a category \mathbb{P} (whose objects interpret *propositions-in-context* $\Gamma \vdash \phi \text{ Prop}$);
- a fibration $q : \mathbb{P} \rightarrow \mathbb{B}$ (mapping $\Gamma \vdash \phi \text{ Prop}$ to Γ)



Categorical Semantics for a LTT

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

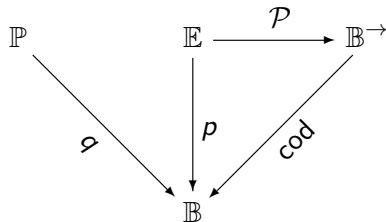
- a category \mathbb{P} (whose objects interpret *propositions-in-context* $\Gamma \vdash \phi \text{ Prop}$);
- a fibration $q : \mathbb{P} \rightarrow \mathbb{B}$ (mapping $\Gamma \vdash \phi \text{ Prop}$ to Γ)
- for every object $\Gamma \vdash A \text{ Type}$ in \mathbb{E} , a right adjoint $\pi^*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



Categorical Semantics for a LTT

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

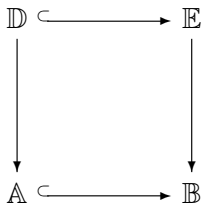
- a category \mathbb{P} (whose objects interpret *propositions-in-context* $\Gamma \vdash \phi \text{ Prop}$);
- a fibration $q : \mathbb{P} \rightarrow \mathbb{B}$ (mapping $\Gamma \vdash \phi \text{ Prop}$ to Γ)
- for every object $\Gamma \vdash A \text{ Type}$ in \mathbb{E} , a right adjoint $\pi^*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

$$\begin{array}{ccc} \mathbb{D} & \hookrightarrow & \mathbb{E} \\ \downarrow & & \downarrow \\ \mathbb{A} & \hookrightarrow & \mathbb{B} \end{array}$$

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 $x : U \vdash T(x)$ Type in \mathbb{E} over $\text{dom } \mathcal{P}(\vdash U \text{ Type})$;

$$\begin{array}{ccc} \mathbb{D} & \longleftarrow & \mathbb{D} \times_{\mathbb{A}} \mathbb{B} \\ & \lrcorner & \downarrow \\ & & \mathbb{B} \\ \downarrow & & \longleftarrow \\ \mathbb{A} & & \end{array}$$

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 $x : U \vdash T(x)$ Type in \mathbb{E} over $\text{dom } \mathcal{P}(\vdash U \text{ Type})$;

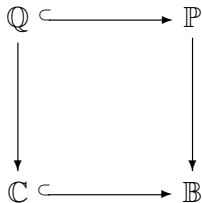
Categorical Semantics for Universes

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 $x : U \vdash T(x)$ Type in \mathbb{E} over $\text{dom } \mathcal{P}(\vdash U \text{ Type})$;

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 $x : U \vdash T(x)$ Type in \mathbb{E} over $\text{dom } \mathcal{P}(\vdash U \text{ Type})$;

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 $x : U \vdash T(x)$ Type in \mathbb{E} over $\text{dom } \mathcal{P}(\vdash U \text{ Type})$;

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 $\text{dom } \mathcal{P}_{\text{prop}}$.

$$\begin{array}{ccc} \mathbb{Q} & \longleftarrow & \mathbb{Q} \times_{\mathbb{C}} \mathbb{B} \\ & \lrcorner & \downarrow \\ & & \mathbb{B} \\ \downarrow & & \downarrow \\ \mathbb{C} & \longleftarrow & \mathbb{B} \end{array}$$

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 $x : U \vdash T(x) \text{ Type}$ in \mathbb{E} over $\text{dom } \mathcal{P}(\vdash U \text{ Type})$;

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 $\text{dom } \mathcal{P}_{\text{prop}}$.

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 $x : U \vdash T(x)$ Type in \mathbb{E} over $\text{dom } \mathcal{P}(\vdash U \text{ Type})$;

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 $\text{dom } \mathcal{P}_{\text{prop}}$.

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

Interpretation

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

- for every valid context Γ , an object $\llbracket \Gamma \rrbracket$ of \mathbb{B} ;
- for every type A such that $\Gamma \vdash A \text{ Type}$, an object $\llbracket \Gamma \vdash A \rrbracket$ of \mathbb{E} such that $\rho \llbracket \Gamma \vdash A \rrbracket = \llbracket \Gamma \rrbracket$
- for every term M such that $\Gamma \vdash M : A$, an arrow $\llbracket \Gamma \vdash M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \text{dom } \mathcal{P} \llbracket \Gamma \vdash A \rrbracket$
- for every proposition ϕ such that $\Gamma \vdash \phi \text{ prop}$, an object $\llbracket \Gamma \vdash \phi \rrbracket$ of \mathbb{P} over $\llbracket \Gamma \rrbracket$

Soundness Theorem

Theorem

Every judgement is true in any LTT_W -category. That is:

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

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 $\text{Cl}(T)$, 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 $\Gamma \vdash A \text{ Type}$, quotiented by equality;
- ...

If a judgement is true in $\text{Cl}(T)$, 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 $\text{Cl}(T)$, 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 $\Gamma \vdash A \text{ Type}$, quotiented by equality;
- ...

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

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

Completeness Theorem

Theorem

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

Proof.

Define the category $\text{Cl}(T)$, 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 $\Gamma \vdash A \text{ Type}$, quotiented by equality;
- ...

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

In fact, $\text{Cl}(T)$ is an initial object in the metacategory of $\text{LTT}_{\mathbb{W}}$ -categories. The interpretation given earlier is the unique functor $\text{Cl}(T) \rightarrow \mathbb{C}$. 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 hom 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 , \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 . □

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 $I\Sigma_0(\text{exp})$.

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 $I\Sigma_0(\text{exp})$.

(Show that the functions in $\mathbb{N} \rightarrow \mathbb{N}$ are all defined by a Σ_0 -formula in $I\Sigma_0(\text{exp})$. 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

LTT₀ is a system with:

Judgement forms:

$$\Gamma \vdash A \text{ type} \quad \Gamma \vdash M : A$$
$$\Gamma \vdash \phi \text{ Prop} \quad \Gamma \vdash P : \phi$$

and associated equality judgements.

Type $A ::=$

Term $M ::= x$

Proposition $\phi ::=$

Proof $P ::=$

Syntax of an LTT

LTT₀ is a system with:

- arrow types

Type $A ::= A \rightarrow A$

Term $M ::= x \mid \lambda x : A. M \mid MM$

Proposition $\phi ::=$

Proof $P ::=$

Syntax of an LTT

LTT₀ is a system with:

- arrow types
- product types

Type $A ::= A \rightarrow A \mid A \times A$

Term $M ::= x \mid \lambda x : A. M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M)$

Proposition $\phi ::=$

Proof $P ::=$

Syntax of an LTT

LTT₀ is a system with:

- arrow types
- product types
- natural numbers

Type $A ::= A \rightarrow A \mid A \times A \mid \mathbb{N}$

Term $M ::= x \mid \lambda x : A. M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M)$

Proposition $\phi ::=$

Proof $P ::=$

Syntax of an LTT

LTT₀ is a system with:

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

Type $A ::= A \rightarrow A \mid A \times A \mid \mathbb{N} \mid U \mid T(M)$

Term $M ::= x \mid \lambda x : A. M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \mid \hat{\mathbb{N}} \mid M \hat{\times} M$

Proposition $\phi ::=$

Proof $P ::=$

Syntax of an LTT

LTT₀ is a system with:

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

| | |
|-------------|--|
| Type | $A ::= A \rightarrow A \mid A \times A \mid \mathbb{N} \mid U \mid T(M)$ |
| Term | $M ::= x \mid \lambda x : A. M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \mid \hat{\mathbb{N}} \mid M \hat{\times} M$ |
| Proposition | $\phi ::= M =_A M \mid \neg \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \forall x : A. \phi \mid \exists x : A. \phi$ |
| Proof | $P ::= \dots$ |

Syntax of an LTT

LTT₀ 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*

| | |
|-------------|---|
| Type | $A ::= A \rightarrow A \mid A \times A \mid \mathbb{N} \mid U \mid T(M)$ |
| Term | $M ::= x \mid \lambda x : A. M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \mid \hat{\mathbb{N}} \mid M \hat{\times} M$ |
| Proposition | $\phi ::= M =_A M \mid \neg \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \forall x : A. \phi \mid \exists x : A. \phi \mid \text{prop} \mid V(P)$ |
| Proof | $P ::= \dots \mid M \hat{=} M \mid \hat{\wedge} \phi \mid \phi \hat{\wedge} \phi \mid \dots$ |

Syntax of an LTT

LTT₀ 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*

| | |
|-------------|--|
| Type | $A ::= A \rightarrow A \mid A \times A \mid \mathbb{N} \mid U \mid T(M) \mid \text{Set}(A)$ |
| Term | $M ::= x \mid \lambda x : A. M \mid MM \mid (M, M) \mid \pi_1(M) \mid \pi_2(M) \mid 0 \mid S(M) \mid E_{\mathbb{N}}(M, M, M, M) \mid \hat{\mathbb{N}} \mid M \hat{\times} M \mid \{x : A \mid P\}$ |
| Proposition | $\phi ::= M =_A M \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \forall x : A. \phi \mid \exists x : A. \phi \mid \text{prop} \mid V(P)$ |
| Proof | $P ::= \dots \mid M \hat{=} M \mid \hat{\wedge}\phi \mid \phi \hat{\wedge}\phi \mid \dots \mid M \in M$ |

We can give a semantic proof of this result:

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

- 1 for all a_1, \dots, a_n , $PA \vdash \phi[\overline{a_1}, \dots, \overline{a_n}, \overline{f(a_1, \dots, a_n)}]$;
- 2 $PA \vdash \forall x_1 \dots \forall x_n \exists! y \phi[x_1, \dots, x_n, y]$

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 ML_1V .

History of LTTs

- 2002 Aczel and Gambino [?] define translations between Constructive ZF (CZF) and the type theory ML_1V .
- 2006 Gambino and Aczel [?] introduce the logic-enriched type theory $ML(CZF)$ as a half-way stage.

History of LTTs

- 2002 Aczel and Gambino [?] define translations between Constructive ZF (CZF) and the type theory ML_1V .
- 2006 Gambino and Aczel [?] introduce the logic-enriched type theory $ML(CZF)$ 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 ML_1V .
- 2006 Gambino and Aczel [?] introduce the logic-enriched type theory $ML(CZF)$ 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'