

Logical Relations for a Logical Framework

presenting [RS:logrels:12] by Florian Rabe and Kristina Sojakova

Navid Roux¹

KWARC Seminar

Computer Science, FAU Erlangen-Nürnberg

TODO: no date yet



This work is licensed under a “CC BY-SA 4.0” license.

¹<https://orcid.org/0000-0002-8348-2441>

What are Logical Relations?

An (informal) class of **proof methods** used to prove

- strong normalization

no infinite evaluation path $t_1 \rightarrow t_2 \rightarrow \dots$

- type safety

$$t : T \text{ and } t \rightarrow^* t' \Rightarrow t' : T$$

- program equivalence

- correctness
- theorems-for-free
- security-typed languages

e.g. of optimizations

terms of $\forall \alpha. \alpha \rightarrow \alpha$ are the identity
 $\text{Prg}(s) \approx \text{Prg}(s')$ for “sensitive” s, s'

see appendix for references

Why embed into a Logical Framework?

Theories of logical relations

- have been stated for many formal systems
- have different flavors
- usually given in meta languages

System F, F_ω , CIC, ...

syntactic, semantic, reflective

General Desire: formalize formal systems and their properties

in proof assistants or logical frameworks

Why embed into a Logical Framework?

Theories of logical relations

- have been stated for many formal systems
- have different flavors
- usually given in meta languages

System F, F_ω , CIC, ...

syntactic, semantic, reflective

General Desire: formalize formal systems and their properties

in proof assistants or logical frameworks

Problem: requires formalization of meta languages, too

even more difficult

Why embed into a Logical Framework?

Theories of logical relations

- have been stated for many formal systems
- have **different flavors**
- usually **given in meta languages**

System F, F_ω , CIC, ...

syntactic, semantic, reflective

General Desire: formalize formal systems and their properties

in proof assistants or logical frameworks

Problem: requires formalization of meta languages, too

even more difficult

Idea: Is there a general notion of logical relations *usable for formalization* and that *unifies* many other notions?

yes, and this talk gives one

Goals of This Talk

- Building intuition for logical relations and how they can be captured in MMT/LF
- h
- formalization of *special cases* of logical relations in MMT/LF

[**RS:logrels:12**] gives much more general definitions and theorems than we do.

Prelims: MMT/LF as a Logical Framework

Definition (MMT/LF Grammar)

MMT/LF combines MMT's module system and the dependent type theory LF:

Thy	$::= T = \{Decl^*\}$	theory definition
$Decl$	$::= c : A [= A] \mid \mathbf{include} \ T$	declarations in a theory
$Morph$	$::= v : S \rightarrow T = \{Ass^*\}$	morphism definition
Ass	$::= c := A \mid \mathbf{include} \ v$	assignments in a morphism
A	$::= \mathbf{type} \mid c \mid x \mid A \ A \mid$ $\lambda x:A. A \mid \Pi x:A. A \mid A \rightarrow A$	terms

E.g. basic propositional logic (PL):

$$\mathbf{theory} \text{ PL} = \left\{ \begin{array}{l} \mathbf{prop} : \mathbf{type} \\ \neg : \mathbf{prop} \rightarrow \mathbf{prop} \\ \wedge, \vee : \mathbf{prop} \rightarrow \mathbf{prop} \rightarrow \mathbf{prop} \end{array} \right\}$$

Prop. Logic in MMT/LF

Example (PL on the meta-level)

P	$::=$	$A \mid \neg P \mid P \wedge P \mid P \vee P$	propositions
A	$::=$	$\langle \text{unspecified} \rangle$	atoms

Use variables p , a for derived terms, and \vdash for the usual intuitionistic proof calculus.

Prop. Logic in MMT/LF

Example (PL on the meta-level)

P	$::=$	$A \mid \neg P \mid P \wedge P \mid P \vee P$	propositions
A	$::=$	$\langle \text{unspecified} \rangle$	atoms

Use variables p , a for derived terms, and \vdash for the usual intuitionistic proof calculus.

theory PL = {
 prop: type
 atom: type
}

Prop. Logic in MMT/LF

Example (PL on the meta-level)

P	$::=$	$A \mid \neg P \mid P \wedge P \mid P \vee P$	propositions
A	$::=$	$\langle \text{unspecified} \rangle$	atoms

Use variables p, a for derived terms, and \vdash for the usual intuitionistic proof calculus.

$$\text{theory PL} = \left\{ \begin{array}{l} \text{prop: type} \\ \text{atom: type} \\ \text{inj: atom} \rightarrow \text{prop} \\ \neg: \text{prop} \rightarrow \text{prop} \\ \wedge, \vee: \text{prop} \rightarrow \text{prop} \rightarrow \text{prop} \end{array} \right\}$$

Prop. Logic in MMT/LF

Example (PL on the meta-level)

P	$::=$	$A \mid \neg P \mid P \wedge P \mid P \vee P$	propositions
A	$::=$	$\langle \text{unspecified} \rangle$	atoms

Use variables p, a for derived terms, and \vdash for the usual intuitionistic proof calculus.

$$\text{theory PL} = \left\{ \begin{array}{l} \text{prop: type} \\ \text{atom: type} \\ \\ \text{inj: atom} \rightarrow \text{prop} \\ \neg: \text{prop} \rightarrow \text{prop} \\ \wedge, \vee: \text{prop} \rightarrow \text{prop} \rightarrow \text{prop} \\ \\ \vdash: \text{prop} \rightarrow \text{type} \\ \wedge_{\text{I}}: \prod p q: \text{prop}. \vdash p \rightarrow \vdash q \rightarrow \vdash p \wedge q \\ \wedge_{\text{EL}}: \prod p q: \text{prop}. \vdash p \wedge q \rightarrow \vdash p \end{array} \right\}$$

Roadmap

- ① \Rightarrow Example 1: unary logical relations along identity
- ② Example 2: binary logical relations along identity
- ③ Example 3: unary logical relation along non-trivial morphism \approx Church
To Curry operator

Example 1: Tertium Non Datur in Prop. Logic

Example (PL on the meta-level)

P	$::=$	$A \mid \neg P \mid P \wedge P \mid P \vee P$	propositions
A	$::=$	$\langle \text{unspecified} \rangle$	atoms

has become an MMT theory

Example 1: Tertium Non Datur in Prop. Logic

Example (PL on the meta-level)

P	$::=$	$A \mid \neg P \mid P \wedge P \mid P \vee P$	propositions
A	$::=$	$\langle \text{unspecified} \rangle$	atoms

has become an MMT theory

Theorem (Tertium Non Datur in PL)

If $\vdash a \vee \neg a$ for all atoms, then $\vdash p \vee \neg p$ for all propositions.

will become a logical relation

TND as a Meta Proof

Proof. ($\vdash p \vee \neg p$ for all prop. p)

Apply **structural induction** for the stronger claim $\text{TND}_-(\text{---})$ where

- $\text{TND}_A(a): \vdash \text{inj}(a) \vee \neg \text{inj}(a)$ for atoms a
- $\text{TND}_P(p): \vdash p \vee \neg p$ for propositions p

For every non-terminal τ , $\text{TND}_\tau(\text{---})$ is a unary predicate on terms of τ !

\leadsto a type-indexed family of relations

TND as a Meta Proof

Proof. ($\vdash p \vee \neg p$ for all prop. p)

Apply **structural induction** for the stronger claim $\text{TND}_-(-)$ where

- $\text{TND}_A(a)$: $\vdash \text{inj}(a) \vee \neg \text{inj}(a)$ for atoms a
- $\text{TND}_P(p)$: $\vdash p \vee \neg p$ for propositions p

For every non-terminal τ , $\text{TND}_\tau(-)$ is a unary predicate on terms of τ !

\leadsto a type-indexed family of relations

Cases

- $\text{TND}_A(a)$: by assumption

TND as a Meta Proof

Proof. ($\vdash p \vee \neg p$ for all prop. p)

Apply **structural induction** for the stronger claim $\text{TND}_-(-)$ where

- $\text{TND}_A(a)$: $\vdash \text{inj}(a) \vee \neg \text{inj}(a)$ for atoms a
- $\text{TND}_P(p)$: $\vdash p \vee \neg p$ for propositions p

For every non-terminal τ , $\text{TND}_\tau(-)$ is a unary predicate on terms of τ !

\leadsto a type-indexed family of relations

Cases

- $\text{TND}_A(a)$: by assumption
- $\text{TND}_P(\text{inj}(a))$: immediately by IH $\text{TND}_A(a)$

TND as a Meta Proof

Proof. ($\vdash p \vee \neg p$ for all prop. p)

Apply **structural induction** for the stronger claim $\text{TND}_-(-)$ where

- $\text{TND}_A(a)$: $\vdash \text{inj}(a) \vee \neg \text{inj}(a)$ for atoms a
- $\text{TND}_P(p)$: $\vdash p \vee \neg p$ for propositions p

For every non-terminal τ , $\text{TND}_\tau(-)$ is a unary predicate on terms of τ !

\leadsto a type-indexed family of relations

Cases

- $\text{TND}_A(a)$: by assumption
- $\text{TND}_P(\text{inj}(a))$: immediately by IH $\text{TND}_A(a)$
- $\text{TND}_P(\neg p)$: $\vdash \neg p \vee \neg \neg p$
 - 1 by IH we have $\text{TND}_P(p)$: $\vdash p \vee \neg p$
 - 2 perform \vee_E on IH
 - if $\vdash p$: then $\vdash \neg \neg p$; done by \vee_{IR} .
 - if $\vdash \neg p$: by \vee_{IL} .

TND as a Meta Proof (cont.)

Proof ($\vdash p \vee \neg p$ for all prop. p ; cont.).

Cases

- $\text{TND}_P(p_1 \wedge p_2): \vdash (p_1 \wedge p_2) \vee \neg(p_1 \wedge p_2)$
 - ① by IH we have $\text{TND}_P(p_1): \vdash p_1 \vee \neg p_1$ and $\text{TND}_P(p_2): \vdash p_2 \vee \neg p_2$
 - ② perform \vee_E on both
 - if $\vdash p_1$ and $\vdash p_2$: apply \vee_{IL}, \wedge_I , done.
 - if at least one $\vdash \neg p_i$: apply $\vee_{IR}, \neg_I, \wedge_{E1}$, contradiction.
- $\text{TND}_P(p_1 \vee p_2): \vdash p_1 \vee p_2 \vee \neg p_1 \vee p_2$: similar.



TND as a Logical Relation

Key Ideas of previous proof:

- 1 state **type-indexed family of predicates** $\text{TND}_-(\text{--})$
- 2 for every type τ , **prove predicate is preserved** under constructors

$c: \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \tau$:

if $\text{TND}_{\tau_1}(t_1) \wedge \dots \wedge \text{TND}_{\tau_n}(t_n)$, then $\text{TND}_{\tau}(c\ t_1 \dots t_n)$

TND as a Logical Relation

Key Ideas of previous proof:

- 1 state **type-indexed family of predicates** $\text{TND}_-(\text{--})$
- 2 for every type τ , **prove predicate is preserved** under constructors

$c: \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \tau$:

if $\text{TND}_{\tau_1}(t_1) \wedge \dots \wedge \text{TND}_{\tau_n}(t_n)$, then $\text{TND}_{\tau}(c\ t_1 \dots t_n)$

In MMT/LF: mimic that with new syntax!

relation $\text{TND}: id_{\text{PL}}: \text{PL} \rightarrow \text{PL} = \{$

$\}$

TND as a Logical Relation

Key Ideas of previous proof:

- 1 state **type-indexed family of predicates** $\text{TND}_-(\text{--})$
- 2 for every type τ , **prove predicate is preserved** under constructors

$c: \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \tau$:

if $\text{TND}_{\tau_1}(t_1) \wedge \dots \wedge \text{TND}_{\tau_n}(t_n)$, then $\text{TND}_{\tau}(c\ t_1 \dots t_n)$

In MMT/LF: mimic that with new syntax!

Recall: we had `atom: type`, `prop: type`

$$\begin{aligned} \text{relation TND: } id_{\text{PL}}: \text{PL} \rightarrow \text{PL} = \{ \\ & \text{atom} := \lambda a: \text{atom}. \vdash \text{inj}(a) \vee \neg \text{inj}(a) \\ & \text{prop} := \underbrace{\lambda p: \text{prop}. \vdash p \vee \neg p}_{\text{encoding of } \text{TND}_{\text{prop}}(-)} \\ & \vdots \\ & \} \end{aligned}$$

TND as a Logical Relation (cont.)

relation TND: $id_{PL}: PL \rightarrow PL = \{$
 $atom := \lambda a: atom. \vdash inj(a) \vee \neg inj(a)$
 $prop := \lambda p: prop. \vdash p \vee \neg p$

TND as a Logical Relation (cont.)

Recall: $\text{inj} : \text{atom} \rightarrow \text{prop}$; encode a proof of $\text{TND}_P(a)$

relation TND: $\text{id}_{\text{PL}} : \text{PL} \rightarrow \text{PL} = \{$
 $\text{atom} := \lambda a : \text{atom}. \vdash \text{inj}(a) \vee \neg \text{inj}(a)$
 $\text{prop} := \lambda p : \text{prop}. \vdash p \vee \neg p$
 $\text{inj} := \lambda a : \text{atom}. \lambda a^* : \vdash \text{inj}(a) \vee \neg \text{inj}(a). a^*$

TND as a Logical Relation (cont.)

Recall: $\neg: \text{prop} \rightarrow \text{prop}$; encode a proof of $\text{TND}_P(p)$ implying $\text{TND}_P(\neg p)$

relation TND: $id_{\text{PL}}: \text{PL} \rightarrow \text{PL} = \{$
 $\text{atom} := \lambda a: \text{atom}. \vdash \text{inj}(a) \vee \neg \text{inj}(a)$
 $\text{prop} := \lambda p: \text{prop}. \vdash p \vee \neg p$

 $\text{inj} := \lambda a: \text{atom}. \lambda a^*: \vdash \text{inj}(a) \vee \neg \text{inj}(a). a^*$
 $\neg := \lambda p: \text{prop}. \lambda p^*: \vdash p \vee \neg p.$
 ... $\langle \text{of type } \vdash \neg p \vee \neg \neg p \rangle$

TND as a Logical Relation (cont.)

Recall: $\neg: \text{prop} \rightarrow \text{prop}$; encode a proof of $\text{TND}_P(p)$ implying $\text{TND}_P(\neg p)$

relation TND: $id_{\text{PL}}: \text{PL} \rightarrow \text{PL} = \{$
 $\text{atom} := \lambda a: \text{atom}. \vdash \text{inj}(a) \vee \neg \text{inj}(a)$
 $\text{prop} := \lambda p: \text{prop}. \vdash p \vee \neg p$

 $\text{inj} := \lambda a: \text{atom}. \lambda a^*: \vdash \text{inj}(a) \vee \neg \text{inj}(a). a^*$
 $\neg := \lambda p: \text{prop}. \lambda p^*: \vdash p \vee \neg p.$
 $\vee_E p^* (\lambda p_{\top}. \vee_{\text{IR}} (\neg_I \lambda p_{\perp}. \neg_E p_{\top} p_{\perp}))$
 $(\lambda p_{\perp}: \vdash \neg p. \vee_{\text{IL}} p_{\perp}) \quad \langle \text{of type } \vdash \neg p \vee \neg \neg p \rangle$

TND as a Logical Relation (cont.)

Recall: $\wedge, \vee: \text{prop} \rightarrow \text{prop} \rightarrow \text{prop}$; encode proofs of $\text{TND}_P(p), \text{TND}_P(q)$ implying $\text{TND}_P(p \wedge q)$ and $\text{TND}_P(p \vee q)$

relation TND: $\text{id}_{\text{PL}}: \text{PL} \rightarrow \text{PL} = \{$

$\text{atom} := \lambda a: \text{atom}. \vdash \text{inj}(a) \vee \neg \text{inj}(a)$

$\text{prop} := \lambda p: \text{prop}. \vdash p \vee \neg p$

$\text{inj} := \lambda a: \text{atom}. \lambda a^*: \vdash \text{inj}(a) \vee \neg \text{inj}(a). a^*$

$\neg := \lambda p: \text{prop}. \lambda p^*: \vdash p \vee \neg p.$

$\vee_E p^* (\lambda p_{\top}. \vee_{\text{IR}} (\neg_I \lambda p_{\perp}. \neg_E p_{\top} p_{\perp}))$

$(\lambda p_{\perp}: \vdash \neg p. \vee_{\text{IL}} p_{\perp}) \quad \langle \text{of type } \vdash \neg p \vee \neg \neg p \rangle$

$\wedge := \lambda p: \text{prop}. \lambda p^*: \vdash p \vee \neg p.$

$\lambda q: \text{prop}. \lambda q^*: \vdash q \vee \neg q. \dots \langle \text{of type } \vdash p \wedge q \vee \neg p \wedge q \rangle$

$\vee := \lambda p: \text{prop}. \lambda p^*: \vdash p \vee \neg p.$

$\lambda q: \text{prop}. \lambda q^*: \vdash q \vee \neg q. \dots \langle \text{of type } \vdash p \vee q \vee \neg p \vee q \rangle$

TND as a Logical Relation (cont.)

Final Result:

relation TND: $id_{PL}: PL \rightarrow PL = \{$

$atom := \lambda a: atom. \vdash inj(a) \vee \neg inj(a)$

$prop := \lambda p: prop. \vdash p \vee \neg p$

$inj := \lambda a: atom. \lambda a^*: \vdash inj(a) \vee \neg inj(a). a^*$

$\neg := \lambda p: prop. \lambda p^*: \vdash p \vee \neg p.$

$\vee_E p^* (\lambda p_{\top}. \vee_{IR} (\neg_I \lambda p_{\perp}. \neg_E p_{\top} p_{\perp}))$

$(\lambda p_{\perp}: \vdash \neg p. \vee_{IL} p_{\perp}) \quad \langle \text{of type } \vdash \neg p \vee \neg \neg p \rangle$

$\wedge := \lambda p: prop. \lambda p^*: \vdash p \vee \neg p.$

$\lambda q: prop. \lambda q^*: \vdash q \vee \neg q. \dots \langle \text{of type } \vdash p \wedge q \vee \neg p \wedge q \rangle$

$\vee := \lambda p: prop. \lambda p^*: \vdash p \vee \neg p.$

$\lambda q: prop. \lambda q^*: \vdash q \vee \neg q. \dots \langle \text{of type } \vdash p \vee q \vee \neg p \vee q \rangle$

$\}$

Consequences of TND as a Log. Rel.

So far only a piece of syntax:

$$\begin{aligned} \text{relation TND: } id_{\text{PL}}: \text{PL} \rightarrow \text{PL} = \{ \\ \quad \text{atom} := \lambda a: \text{atom}. \vdash \text{inj}(a) \vee \neg \text{inj}(a) \\ \quad \text{prop} := \lambda p: \text{prop}. \vdash p \vee \neg p \\ \quad \vdots \\ \} \end{aligned}$$

Consequences of TND as a Log. Rel.

So far only a piece of syntax:

$$\begin{array}{l} \text{relation TND: } id_{\text{PL}}: \text{PL} \rightarrow \text{PL} = \{ \\ \quad \text{atom} := \lambda a: \text{atom. } \vdash \text{inj}(a) \vee \neg \text{inj}(a) \\ \quad \text{prop} := \underbrace{\lambda p: \text{prop. } \vdash p \vee \neg p}_{=:\text{TND}_{\text{prop}}} \\ \quad \vdots \\ \} \end{array}$$

Consequences of TND as a Log. Rel.

So far only a piece of syntax:

$$\begin{aligned} \text{relation TND: } id_{PL}: PL \rightarrow PL = \{ \\ & \text{atom} := \lambda a: \text{atom. } \vdash \text{inj}(a) \vee \neg \text{inj}(a) \\ & \text{prop} := \underbrace{\lambda p: \text{prop. } \vdash p \vee \neg p}_{=:\text{TND}_{\text{prop}}} \\ & \vdots \\ & \} \end{aligned}$$

But meta theory gives: if $r: id_T: T \rightarrow T$ is a logical relation (i.e. well-typed), then

Theorem (Basic Lemma)

If $\vdash_T t: \tau$, then $\vdash_T r(t): r(\tau)$.

Consequences of TND as a Log. Rel.

So far only a piece of syntax:

$$\begin{aligned} \text{relation TND: } id_{\text{PL}}: \text{PL} \rightarrow \text{PL} = \{ \\ \text{atom} := \lambda a: \text{atom. } \vdash \text{inj}(a) \vee \neg \text{inj}(a) \\ \text{prop} := \underbrace{\lambda p: \text{prop. } \vdash p \vee \neg p}_{=:\text{TND}_{\text{prop}}} \\ \vdots \\ \} \end{aligned}$$

But meta theory gives: if $r: id_T: T \rightarrow T$ is a logical relation (i.e. well-typed), then

Theorem (Basic Lemma)

If $\vdash_T t: \tau$, then $\vdash_T r(t): r(\tau)$.

Corollary

If $\vdash_{\text{PL}} p: \text{prop}$ is a proposition, then $\vdash_{\text{PL}} r(p): \text{TND}_{\text{prop}} p$, i.e. $\vdash_{\text{PL}} p \vee \neg p$ is provable.

Logical Relations: A Definition

Definition

Let T be a theory and $(r_c)_{c \in T}$ a family of T -expressions. Define

$$\begin{aligned}r(\mathbf{type}) &= \lambda A: \mathbf{type}. A \rightarrow \mathbf{type} \\r(\Pi x: A. B) &= \lambda f: (\Pi x: A. B). \Pi x: A. \Pi x^*: r(A) \ x. r(B) \ (f \ x) \\r(A \ B) &= r(A) \ B \ r(B) \\r(\lambda x: A. B) &= \lambda x: A. \lambda x^*: r(A) \ x. r(B) \\r(c) &= r_c \\r(x) &= x^*\end{aligned}$$

We call r a **unary logical relation** on id_T if $\vdash_T r_c: r(\tau) \ c$ for all constants $c: \tau$ in T .

- e.g. $\mathbf{prop}: \mathbf{type}$, hence $r_{\mathbf{prop}}: \mathbf{prop} \rightarrow \mathbf{type}$

$$r_{\mathbf{prop}} := \lambda p. \vdash p \vee \neg p$$

- e.g. $\neg: \mathbf{prop} \rightarrow \mathbf{prop}$, hence

$$r_{\neg}: \Pi p: \mathbf{prop}. \Pi p^*: r(\mathbf{prop}) \ p. r(\mathbf{prop}) \ (\neg p)$$

$$r_{\neg} := \langle \text{our proof} \rangle$$

Logical Relations: A Definition

Definition

Let T be a theory and $(r_c)_{c \in T}$ a family of T -expressions. Define

$$\begin{aligned}r(\mathbf{type}) &= \lambda A: \mathbf{type}. A \rightarrow \mathbf{type} \\r(\Pi x: A. B) &= \lambda f: (\Pi x: A. B). \Pi x: A. \Pi x^*: r(A) \ x. r(B) \ (f \ x) \\r(A \ B) &= r(A) \ B \ r(B) \\r(\lambda x: A. B) &= \lambda x: A. \lambda x^*: r(A) \ x. r(B) \\r(c) &= r_c \\r(x) &= x^*\end{aligned}$$

We call r a **unary logical relation** on id_T if $\vdash_T r_c: r(\tau) \ c$ for all constants $c: \tau$ in T .

- e.g. $\mathbf{prop}: \mathbf{type}$, hence $r_{\mathbf{prop}}: \mathbf{prop} \rightarrow \mathbf{type}$

$$r_{\mathbf{prop}} := \lambda p. \vdash p \vee \neg p$$

- e.g. $\neg: \mathbf{prop} \rightarrow \mathbf{prop}$, hence

$$r_{\neg}: \Pi p: \mathbf{prop}. \Pi p^*: \vdash p \vee \neg p. \vdash \neg p \vee \neg \neg p$$

$$r_{\neg} := \langle \text{our proof} \rangle$$

Logical Relations: A Definition

Definition

Let T be a theory and $(r_c)_{c \in T}$ a family of T -expressions. Define

$$\begin{aligned}r(\mathbf{type}) &= \lambda A: \mathbf{type}. A \rightarrow \mathbf{type} \\r(\Pi x: A. B) &= \lambda f: (\Pi x: A. B). \Pi x: A. \Pi x^*: r(A) \ x. r(B) \ (f \ x) \\r(A \ B) &= r(A) \ B \ r(B) \\r(\lambda x: A. B) &= \lambda x: A. \lambda x^*: r(A) \ x. r(B) \\r(c) &= r_c \\r(x) &= x^*\end{aligned}$$

We call r a **unary logical relation** on id_T if $\vdash_T r_c: r(\tau) \ c$ for all constants $c: \tau$ in T .

Theorem (Basic Lemma)

If $\Gamma \vdash_T t: \tau$, then $r(\Gamma) \vdash_T r(t): r(\tau) \ t$.

Example 1: Summary

Summary:

- Logical relations often occur in a type theory, and there
 - associate to every type τ a meta proposition $C_\tau(-)$
 - “the basic lemma” then states: if $t : \tau$, then $C_\tau(t)$
- In LF, we can
 - map every type to a function giving us the meta proposition
e.g. $\text{prop} := \lambda p. \vdash p \vee \neg p$
 - prove for every term constructor $f : \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \tau$ the preservation of $C_\tau(-)$ given $C_{\tau_1}(-), \dots, C_{\tau_n}(-)$
e.g. $\text{TND}_P(p)$ must imply $\text{TND}_P(\neg p)$
“unary congruences”
- As such, they subsume (some) inductive arguments

Next: Binary Logical Relations

Towards Binary Logical Relations

Key Ideas of Unary Relations:

already seen

- 1 state **type-indexed family of predicates** $\text{TND}_-(\text{--})$
- 2 for every type τ , **prove predicate is preserved** under constructors

$C : \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \tau$:

if $\text{TND}_{\tau_1}(t_1) \wedge \dots \wedge \text{TND}_{\tau_n}(t_n)$, then $\text{TND}_{\tau}(C\ t_1 \dots t_n)$

How about $C_P(p, p') := \vdash p \Leftrightarrow p'$?

Towards Binary Logical Relations

Key Ideas of Unary Relations:

already seen

- 1 state **type-indexed family of predicates** $\text{TND}_-(\text{--})$
- 2 for every type τ , **prove predicate is preserved** under constructors
 $c : \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \tau$:

if $\text{TND}_{\tau_1}(t_1) \wedge \dots \wedge \text{TND}_{\tau_n}(t_n)$, then $\text{TND}_{\tau}(c\ t_1 \dots t_n)$

How about $C_P(p, p') := \vdash p \Leftrightarrow p'$?

- 1 forms a type-indexed family of **binary** predicates
together with $C_A(a, b) := \vdash \text{inj}(a) \Leftrightarrow \text{inj}(b)$
- 2 e.g. is preserved under \wedge :
 - given $\vdash p \Leftrightarrow p'$ and $\vdash q \Leftrightarrow q'$
 - we can derive $\vdash (p \wedge q) \Leftrightarrow (p' \wedge q')$

Example 2: \Leftrightarrow congruence in Prop. Logic

Example (PL Extended)

We extend PL's grammar by

$$\begin{array}{ll} P & ::= A \mid \neg P \mid P \wedge P \mid P \vee P \mid \text{propositions} \\ & \quad \quad \quad \textcolor{red}{P \Leftrightarrow P} \\ A & ::= \langle \text{unspecified} \rangle \quad \text{atoms} \end{array}$$

and its proof calculus accordingly.

Theorem (\Leftrightarrow (LRA) Congruence)

\Leftrightarrow is a congruence on P . That is, if $\vdash p \Leftrightarrow p'$ and $\vdash q \Leftrightarrow q'$, then

- $\vdash \neg p \Leftrightarrow \neg p'$
- $\vdash p \vee q \Leftrightarrow p' \vee q'$
- $\vdash p \wedge q \Leftrightarrow p' \wedge q'$
- $\vdash (p \Leftrightarrow q) \Leftrightarrow (p' \Leftrightarrow q')$,

and, trivially, if $\vdash \text{inj}(a) \Leftrightarrow \text{inj}(a')$, then $\vdash \text{inj}(a) \Leftrightarrow \text{inj}(a')$.

LRA as a Logical Relation

relation LRA: $id_{\text{PL}} \times id_{\text{PL}}: \text{PL} \rightarrow \text{PL} = \{$

LRA as a Logical Relation

Recall: `atom`: type, map to the desired binary relation on atoms

relation LRA: $id_{PL} \times id_{PL}: PL \rightarrow PL = \{$
 $atom := \lambda a: atom. \lambda a': atom. \vdash inj(a) \Leftrightarrow inj(a')$

LRA as a Logical Relation

Recall: `prop`: type, map to the desired binary relation on propositions

```
relation LRA:  $id_{PL} \times id_{PL}$ : PL  $\rightarrow$  PL = {  
  atom :=  $\lambda a: atom. \lambda a': atom. \vdash inj(a) \Leftrightarrow inj(a')$   
  prop :=  $\lambda p: prop. \lambda p': prop. \vdash p \Leftrightarrow p'$ 
```

LRA as a Logical Relation

Recall: $\text{inj} : \text{atom} \rightarrow \text{prop}$, map to appropriate proof

relation LRA: $id_{\text{PL}} \times id_{\text{PL}} : \text{PL} \rightarrow \text{PL} = \{$
 $\text{atom} := \lambda a : \text{atom}. \lambda a' : \text{atom}. \vdash \text{inj}(a) \Leftrightarrow \text{inj}(a')$
 $\text{prop} := \lambda p : \text{prop}. \lambda p' : \text{prop}. \vdash p \Leftrightarrow p'$
 $\text{inj} := \lambda a : \text{atom}. \lambda a' : \text{atom}. \lambda a^* : \vdash \text{inj}(a) \Leftrightarrow \text{inj}(a'). \ a^*$

LRA as a Logical Relation

Recall: $\neg: \text{prop} \rightarrow \text{prop}$, map to appropriate proof

relation LRA: $id_{PL} \times id_{PL}: PL \rightarrow PL = \{$
 $\text{atom} := \lambda a: \text{atom}. \lambda a': \text{atom}. \vdash \text{inj}(a) \Leftrightarrow \text{inj}(a')$
 $\text{prop} := \lambda p: \text{prop}. \lambda p': \text{prop}. \vdash p \Leftrightarrow p'$

 $\text{inj} := \lambda a: \text{atom}. \lambda a': \text{atom}. \lambda a^*: \vdash \text{inj}(a) \Leftrightarrow \text{inj}(a'). \quad a^*$
 $\neg := \lambda p: \text{prop}. \lambda p': \text{prop}. \lambda p^*: \vdash p \Leftrightarrow p'.$
 $\dots \langle \text{of type } \vdash \neg p \Leftrightarrow \neg p' \rangle$

LRA as a Logical Relation

Recall: $\wedge, \vee: \text{prop} \rightarrow \text{prop} \rightarrow \text{prop}$, map to appropriate proofs

```
relation LRA:  $id_{PL} \times id_{PL}: PL \rightarrow PL = \{$   
   $\text{atom} := \lambda a: \text{atom}. \lambda a': \text{atom}. \vdash \text{inj}(a) \Leftrightarrow \text{inj}(a')$   
   $\text{prop} := \lambda p: \text{prop}. \lambda p': \text{prop}. \vdash p \Leftrightarrow p'$   
  
   $\text{inj} := \lambda a: \text{atom}. \lambda a': \text{atom}. \lambda a^*: \vdash \text{inj}(a) \Leftrightarrow \text{inj}(a'). \quad a^*$   
   $\neg := \lambda p: \text{prop}. \lambda p': \text{prop}. \lambda p^*: \vdash p \Leftrightarrow p'.$   
    ...  $\langle \text{of type } \vdash \neg p \Leftrightarrow \neg p' \rangle$   
   $\wedge := \lambda p: \text{prop}. \lambda p': \text{prop}. \lambda p^*: \vdash p \Leftrightarrow p'. \quad // \text{ first arg of } \wedge$   
     $\lambda q: \text{prop}. \lambda q': \text{prop}. \lambda q^*: \vdash q \Leftrightarrow q'. \quad // \text{ second arg of } \wedge$   
    ...  $\langle \text{of type } \vdash p \wedge q \Leftrightarrow p' \wedge q' \rangle$ 
```

LRA as a Logical Relation

Recall: $\wedge, \vee: \text{prop} \rightarrow \text{prop} \rightarrow \text{prop}$, map to

relation LRA: $id_{PL} \times id_{PL}: PL \rightarrow PL = \{$

$\text{atom} := \lambda a: \text{atom}. \lambda a': \text{atom}. \vdash \text{inj}(a) \Leftrightarrow \text{inj}(a')$

$\text{prop} := \lambda p: \text{prop}. \lambda p': \text{prop}. \vdash p \Leftrightarrow p'$

$\text{inj} := \lambda a: \text{atom}. \lambda a': \text{atom}. \lambda a^*: \vdash \text{inj}(a) \Leftrightarrow \text{inj}(a'). \quad a^*$

$\neg := \lambda p: \text{prop}. \lambda p': \text{prop}. \lambda p^*: \vdash p \Leftrightarrow p'.$

$\dots \langle \text{of type } \vdash \neg p \Leftrightarrow \neg p' \rangle$

$\wedge := \lambda p: \text{prop}. \lambda p': \text{prop}. \lambda p^*: \vdash p \Leftrightarrow p'. \quad // \text{ first arg of } \wedge$

$\lambda q: \text{prop}. \lambda q': \text{prop}. \lambda q^*: \vdash q \Leftrightarrow q'. \quad // \text{ second arg of } \wedge$

$\dots \langle \text{of type } \vdash p \wedge q \Leftrightarrow p' \wedge q' \rangle$

$\vee := \lambda p: \text{prop}. \lambda p': \text{prop}. \lambda p^*: \vdash p \Leftrightarrow p'. \quad // \text{ first arg of } \vee$

$\lambda q: \text{prop}. \lambda q': \text{prop}. \lambda q^*: \vdash q \Leftrightarrow q'. \quad // \text{ second arg of } \vee$

$\dots \langle \text{of type } \vdash p \vee q \Leftrightarrow p' \vee q' \rangle$

LRA as a Logical Relation

relation LRA: $id_{PL} \times id_{PL}: PL \rightarrow PL = \{$
 $atom := \lambda a: atom. \lambda a': atom. \vdash inj(a) \Leftrightarrow inj(a')$
 $prop := \lambda p: prop. \lambda p': prop. \vdash p \Leftrightarrow p'$

 $inj := \lambda a: atom. \lambda a': atom. \lambda a^*: \vdash inj(a) \Leftrightarrow inj(a'). \quad a^*$
 $\neg := \lambda p: prop. \lambda p': prop. \lambda p^*: \vdash p \Leftrightarrow p'.$
 $\dots \langle \text{of type } \vdash \neg p \Leftrightarrow \neg p' \rangle$
 $\wedge := \lambda p: prop. \lambda p': prop. \lambda p^*: \vdash p \Leftrightarrow p'. \quad // \text{ first arg of } \wedge$
 $\lambda q: prop. \lambda q': prop. \lambda q^*: \vdash q \Leftrightarrow q'. \quad // \text{ second arg of } \wedge$
 $\dots \langle \text{of type } \vdash p \wedge q \Leftrightarrow p' \wedge q' \rangle$
 $\vee := \lambda p: prop. \lambda p': prop. \lambda p^*: \vdash p \Leftrightarrow p'. \quad // \text{ first arg of } \vee$
 $\lambda q: prop. \lambda q': prop. \lambda q^*: \vdash q \Leftrightarrow q'. \quad // \text{ second arg of } \vee$
 $\dots \langle \text{of type } \vdash p \vee q \Leftrightarrow p' \vee q' \rangle$

Logical Relations: A Second Definition

Definition

Let T be a theory and $(r_c)_{c \in T}$ a family of T -expressions. Define

$$r(\text{type}) = \lambda A : \text{type}. \lambda A' : \text{type}. A \rightarrow A' \rightarrow \text{type}$$

$$r(\Pi a : A. B) = \lambda f : (\Pi a : A. B). \lambda f' : (\Pi a' : A'. B'). \\ \Pi x : A. \Pi x' : A'. \Pi x^* : r(A) \ x \ x'. \\ r(B) \ (f \ x) \ (f' \ x')$$

$$r(A \ B) = r(A) \ B \ r(B)$$

$$r(\lambda x : A. B) = \lambda x : A. \lambda x' : A'. \lambda x^* : r(A) \ x \ x'. r(B)$$

$$r(c) = r_c$$

$$r(x) = x^*$$

where A' denotes systematic priming. We call r a **binary logical relation on** $id_T \times id_T$ if $\vdash_T r_c : r(\tau) \ c \ c$ for all constants $c : \tau$ in T .

- $\text{prop} : \text{type}$, hence $r_{\text{prop}} : \text{prop} \rightarrow \text{prop} \rightarrow \text{type}$

$$r_{\text{prop}} := \lambda p. \lambda p'. \vdash p \Leftrightarrow p'$$

Logical Relations: A Second Definition

Definition

Let T be a theory and $(r_c)_{c \in T}$ a family of T -expressions. Define

$$r(\text{type}) = \lambda A : \text{type}. \lambda A' : \text{type}. A \rightarrow A' \rightarrow \text{type}$$

$$r(\Pi a : A. B) = \lambda f : (\Pi a : A. B). \lambda f' : (\Pi a' : A'. B').$$

$$\Pi x : A. \Pi x' : A'. \Pi x^* : r(A) \ x \ x'.$$

$$r(B) \ (f \ x) \ (f' \ x')$$

$$r(A \ B) = r(A) \ B \ r(B)$$

$$r(\lambda x : A. B) = \lambda x : A. \lambda x' : A'. \lambda x^* : r(A) \ x \ x'. r(B)$$

$$r(c) = r_c$$

$$r(x) = x^*$$

where A' denotes systematic priming. We call r a **binary logical relation** on $id_T \times id_T$ if $\vdash_T r_c : r(\tau) \ c \ c$ for all constants $c : \tau$ in T .

- $\neg : \text{prop} \rightarrow \text{prop}$, hence

$$r_{\neg} : \Pi p' : \text{prop}. \Pi p^* : \vdash p \Leftrightarrow p'. \vdash \neg p \Leftrightarrow \neg p'$$

Logical Relations: A Second Definition

Definition

Let T be a theory and $(r_c)_{c \in T}$ a family of T -expressions. Define

$$r(\text{type}) = \lambda A : \text{type}. \lambda A' : \text{type}. A \rightarrow A' \rightarrow \text{type}$$

$$r(\Pi a : A. B) = \lambda f : (\Pi a : A. B). \lambda f' : (\Pi a' : A'. B').$$

$$\Pi x : A. \Pi x' : A'. \Pi x^* : r(A) \ x \ x'.$$

$$r(B) \ (f \ x) \ (f' \ x')$$

$$r(A \ B) = r(A) \ B \ r(B)$$

$$r(\lambda x : A. B) = \lambda x : A. \lambda x' : A'. \lambda x^* : r(A) \ x \ x'. r(B)$$

$$r(c) = r_c$$

$$r(x) = x^*$$

where A' denotes systematic priming. We call r a **binary logical relation on** $id_T \times id_T$ if $\vdash_T r_c : r(\tau) \ c \ c$ for all constants $c : \tau$ in T .

Theorem (Basic Lemma). If $\vdash_T t : \tau$, then $\vdash_T r(t) : r(\tau) \ t \ t'$.

Example 2: Summary

Summary: we have *n*-ary logical relations of the form

$$\text{relation } r: id_T \times \cdots \times id_T: T \rightarrow T = \{...\}$$

- they associate to every type τ an *n*-ary relation $C_\tau(-, \dots, -)$
- their well-typedness expresses that these relations are congruences
- “the basic lemma” states: if $t: \tau$, then $C_\tau(t, \dots, t)$

reflexivity? TODO

Next: Logical Relations along a *non-trivial* morphism

Logical Relations of a Single Morphism

theory PL

prop: type	$\lambda p: \text{prop}. \vdash p \vee \neg p$
atom: type	$\lambda a: \text{atom}. \vdash \text{inj}(a) \vee \neg \text{inj}(a)$
inj: atom \rightarrow prop	for $a: \text{atom}$: $\text{inj}(a)$
\wedge : prop \rightarrow prop \rightarrow prop	for $p q: \text{prop}$: $p \wedge q$

theory PL $\xrightarrow{id_{\text{PL}}}$ theory PL

prop: type
atom: type
inj: atom \rightarrow prop
 \wedge : prop \rightarrow prop \rightarrow prop

“structure space”

$\lambda p: id_{\text{PL}}(\text{prop}). \vdash p \vee \neg p$
 $\lambda a: id_{\text{PL}}(\text{atom}). \vdash \text{inj}(a) \vee \neg \text{inj}(a)$
for $a: id_{\text{PL}}(\text{atom})$: $id_{\text{PL}}(\text{inj})(a)$
for $p q: id_{\text{PL}}(\text{prop})$: $p id_{\text{PL}}(\wedge) q$

“assertion/proof space”

Logical Relations of a Single Morphism

theory PL

prop: type

atom: type

inj: atom \rightarrow prop

\wedge : prop \rightarrow prop \rightarrow prop

λp : prop. $\vdash p \vee \neg p$

λa : atom. $\vdash \text{inj}(a) \vee \neg \text{inj}(a)$

for a : atom: $\text{inj}(a)$

for $p q$: prop: $p \wedge q$

theory PL \xrightarrow{v} theory T

prop: type

atom: type

inj: atom \rightarrow prop

\wedge : prop \rightarrow prop \rightarrow prop

“structure space”

λp : $v(\text{prop})$. $\vdash p \vee \neg p$

λa : $v(\text{atom})$. $\vdash \text{inj}(a) \vee \neg \text{inj}(a)$

for a : $v(\text{atom})$: $v(\text{inj})(a)$

for $p q$: $v(\text{prop})$: $p v(\wedge) q$

“assertion/proof space”

Logical Relations of a Single Morphism

theory PL

prop: type

atom: type

inj: atom \rightarrow prop

\wedge : prop \rightarrow prop \rightarrow prop

λp : prop. $\vdash p \vee \neg p$

λa : atom. $\vdash \text{inj}(a) \vee \neg \text{inj}(a)$

for a : atom: $\text{inj}(a)$

for $p q$: prop: $p \wedge q$

theory PL $\xRightarrow[\textcolor{red}{w}]{\textcolor{red}{v}}$ theory T

prop: type

atom: type

inj: atom \rightarrow prop

\wedge : prop \rightarrow prop \rightarrow prop

λp : $\textcolor{red}{v}(\text{prop})$. $\lambda p'$: $\textcolor{red}{w}(\text{prop})$

λa : $\textcolor{red}{v}(\text{atom})$. $\lambda a'$: $\textcolor{red}{w}(\text{atom})$

for a : $\textcolor{red}{v}(\text{atom})$, a' : $\textcolor{red}{w}(\text{atom})$: $\textcolor{red}{v}(\text{inj})(a)$

for $p q$: $\textcolor{red}{v}(\text{prop})$: $\textcolor{red}{p} \textcolor{red}{v}(\wedge) q$

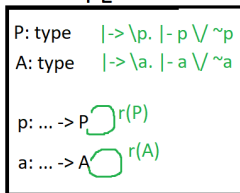
“structure space”

“assertion/proof space”

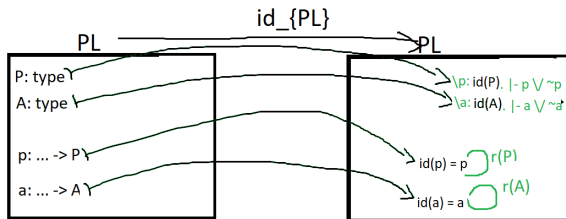
Logical Relations of a Single Morphism

So far: **relation** TND: $id_{PL}: PL \rightarrow PL = \{...\}$:

PL



What does id_{PL} stand for?



Towards n -ary Logical Relations

Intuition:

- have a **single identity morphism** id_T

idea: generalize to $\mu : S \rightarrow T$

- relation asserts a **unary predicate** $C_\tau(t) = C_\tau(id_T(t))$ for every $t : \tau$ of the domain

idea: generalize to n -ary predicates with n morphisms $S \rightarrow T$

- domain is “structure” space
- codomain is “assertion and proof” space

idea: need not be the same

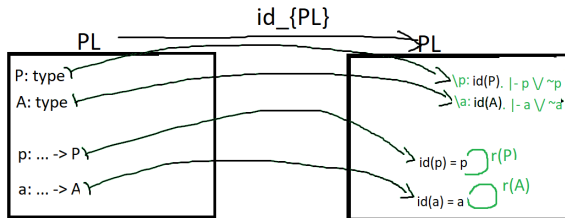
n -ary Logical Relations

- let $\mu_1, \dots, \mu_n: S \rightarrow T$ be morphisms

on common (co)domain theories

- introduce construct

$$\text{relation } r: \mu_1 \times \dots \times \mu_n: S \rightarrow T = \{c_1 := e_1, \dots\}$$



Example 3: Church To Curry

Example (Intrinsic vs. Extrinsic Typing)

A type theory is

- **intrinsically typed** if its terms carry their types.

pair constructor takes type arguments *and* typed terms

$(a, b)^{A \times B}$ is a term

- **extrinsically typed** if its terms are untyped, but “external” typing judgements exist.

pair constructor takes untyped terms

(a, b) is a term

$(a, b) :: A \times B$ a typing judgement

also “Church vs. Curry”

Church and Curry Products

$$\begin{aligned}
 \text{theory ChurchProd} &= \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\} \\
 \\
 \text{theory CurryProd} &= \left\{ \begin{array}{l} \text{tp, tm} : \text{type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A
 \end{array} \right.
 \end{aligned}$$

$$\begin{array}{llll}
 \text{view} & \text{TypeEras:} & \text{ChurchProd} & \rightarrow \text{CurryProd} = \\
 \text{tp} & := \text{tp} & & \\
 \text{tm} & := \lambda _. \text{tm} & & \\
 _ \times _ & := _ \times _ & & \\
 (_, _)^{-\times} & := \lambda _. \lambda _. (_, _) & & \\
 \end{array}$$

e.g. $\text{TypeEras}((a, b)^{A \times B}) = (a, b)$
type information lost!

Capturing Type Preservation

Desired property:

to be captured *within* the formalization

Theorem (Type Preservation)

If $\vdash_{\text{ChurchProd}} t : \text{tm } A$, then

$$\vdash_{\text{CurryProd}} \text{TypeEras}(t) :: A.$$

Capturing Type Preservation

Desired property:

to be captured *within* the formalization

Theorem (Type Preservation)

If $\vdash_{\text{ChurchProd}} t : \text{tm } A$, then

$$\vdash_{\text{CurryProd}} \text{TypeEras}(t) :: \text{TypeEras}(A).$$

TypeEras is identity on types $A : \text{tp}$

Capturing Type Preservation

Desired property:

to be captured *within* the formalization

Theorem (Type Preservation)

If $\vdash_{\text{ChurchProd}} t : \text{tm } A$, *there is wit st.*

$$\vdash_{\text{CurryProd}} \text{wit} : \text{TypeEras}(t) :: \text{TypeEras}(A).$$

Capturing Type Preservation

Desired property:

to be captured *within* the formalization

Theorem (Type Preservation)

If $\vdash_{\text{ChurchProd}} t : \text{tm } A$, *there is* $\text{wit } st$.

$$\vdash_{\text{CurryProd}} \text{wit} : \text{TypeEras}(t) :: \text{TypeEras}(A).$$

Capturing Type Preservation

Desired property:

to be captured *within* the formalization

Theorem (Type Preservation)

If $\vdash_{\text{ChurchProd}} t : \text{tm } A$, *there is wit st.*

$$\vdash_{\text{CurryProd}} \text{wit} : \text{TypeEras}(t) :: \text{TypeEras}(A).$$

Resembles Basic Lemma: let $r : \mu : S \rightarrow T$ be a logical relation over $\mu : S \rightarrow T$, then

Theorem (Basic Lemma)

If $\vdash_S t : \tau$, then $\vdash_T r(t) : r(\tau) \ \mu(t)$.

\Rightarrow create relation $\text{TypePres} : \text{TypeEras} : \text{ChurchProd} \rightarrow \text{CurryProd}$
mapping $\text{tm} - \text{to} _ :: _$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{ll} \text{tp} & : \text{type} \\ \text{tm} & : \text{tp} \rightarrow \text{type} \\ _ \times _ & : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} & : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{ll} \text{tp, tm} : \text{type} & \\ _ :: _ & : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ & : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) & : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -^{\times}} & : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ & a :: A \rightarrow b :: B \rightarrow (a, b) :: (A, B) \end{array} \right.$$

$$\begin{array}{llll} \text{relation} & \text{TypePres: TypeEras: ChurchProd} & \rightarrow & \text{CurryProd} \\ \text{tp} & := \lambda A : \text{TypeEras}(\text{tp}). \text{Unit} & & \end{array}$$

Type Preservation as a Logical Relation

$$\begin{aligned}
 \text{theory ChurchProd} &= \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\} \\
 \\
 \text{theory CurryProd} &= \left\{ \begin{array}{l} \text{tp, tm} : \text{type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -^{\times}} \quad : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A \times B) \end{array} \right\} \\
 \\
 \text{relation} \quad \text{TypePres} : \text{TypeEras} : \text{ChurchProd} \quad \rightarrow \quad \text{CurryProd} &= \\
 \text{tp} \quad \quad \quad := \lambda A : \text{tp}. \text{Unit} &
 \end{aligned}$$

Type Preservation as a Logical Relation

$$\begin{aligned}
 \text{theory ChurchProd} &= \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B: \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\} \\
 \\
 \text{theory CurryProd} &= \left\{ \begin{array}{l} \text{tp, tm: type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} \quad : \Pi A B: \text{tp}. \Pi a b: \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A \times B) \end{array} \right\} \\
 \\
 \text{relation} \quad \text{TypePres: TypeEras: ChurchProd} &\rightarrow \text{CurryProd} \\
 \text{tp} &:= \lambda A: \text{tp}. \text{Unit} \\
 \text{tm} &:= \lambda A: \text{TypeEras}(tp).
 \end{aligned}$$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B: \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{l} \text{tp, tm: type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} \quad : \Pi A B: \text{tp}. \Pi a b: \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A$$

$$\begin{array}{llll} \text{relation} & \text{TypePres:} & \text{TypeEras:} & \text{ChurchProd} \quad \rightarrow \quad \text{CurryProd} \\ \text{tp} & := \lambda A: \text{tp}. \text{Unit} & & \\ \text{tm} & := \lambda A: \text{tp}. & & \end{array}$$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{l} \text{tp, tm} : \text{type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A$$

$$\begin{array}{llll} \text{relation} & \text{TypePres:} & \text{TypeEras:} & \text{ChurchProd} \quad \rightarrow \quad \text{CurryProd} \\ \text{tp} & := \lambda A : \text{tp}. \text{Unit} & & \\ \text{tm} & := \lambda A : \text{tp}. \lambda t : \text{TypeEras}(\text{tm } A). & & \end{array}$$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B: \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{l} \text{tp, tm: type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} \quad : \Pi A B: \text{tp}. \Pi a b: \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A$$

$$\begin{array}{llll} \text{relation} & \text{TypePres:} & \text{TypeEras:} & \text{ChurchProd} \quad \rightarrow \quad \text{CurryProd} \\ \text{tp} & := \lambda A: \text{tp}. \text{Unit} & & \\ \text{tm} & := \lambda A: \text{tp}. \lambda t: \text{tm}. & & \end{array}$$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B: \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{l} \text{tp, tm: type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} \quad : \Pi A B: \text{tp}. \Pi a b: \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A$$

$$\begin{array}{llll} \text{relation} & \text{TypePres:} & \text{TypeEras:} & \text{ChurchProd} \quad \rightarrow \quad \text{CurryProd} \\ \text{tp} & := \lambda A: \text{tp}. \text{Unit} & & \\ \text{tm} & := \lambda A: \text{tp}. \lambda t: \text{tm}. t :: A & & \end{array}$$

Type Preservation as a Logical Relation

$$\begin{aligned}
 \text{theory ChurchProd} &= \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\} \\
 \\
 \text{theory CurryProd} &= \left\{ \begin{array}{l} \text{tp, tm} : \text{type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A, B) \end{array} \right\} \\
 \\
 \text{relation} \quad \text{TypePres} : \text{TypeEras} : \text{ChurchProd} \quad \rightarrow \quad \text{CurryProd} \quad = \\
 \text{tp} \quad \quad \quad &:= \lambda A : \text{tp}. \text{Unit} \\
 \text{tm} \quad \quad \quad &:= \lambda A : \text{tp}. \lambda t : \text{tm}. t :: A \\
 _ \times _ \quad \quad &:= \dots \text{unit}
 \end{aligned}$$

Type Preservation as a Logical Relation

$$\begin{aligned}
 \text{theory ChurchProd} &= \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\} \\
 \\
 \text{theory CurryProd} &= \left\{ \begin{array}{l} \text{tp, tm} : \text{type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A, B) \end{array} \right\} \\
 \\
 \text{relation} \quad \text{TypePres} : \text{TypeEras} : \text{ChurchProd} \quad \rightarrow \quad \text{CurryProd} \quad = \\
 \text{tp} \quad &:= \lambda A : \text{tp}. \text{Unit} \\
 \text{tm} \quad &:= \lambda A : \text{tp}. \lambda t : \text{tm}. t :: A \\
 _ \times _ &:= \dots \text{unit} \\
 (_, _)^{-\times} &:=
 \end{aligned}$$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{l} \text{tp, tm} : \text{type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A, B) \end{array} \right.$$

$$\begin{array}{llll} \text{relation} & \text{TypePres:} & \text{TypeEras:} & \text{ChurchProd} \quad \rightarrow \quad \text{CurryProd} \\ \text{tp} & := \lambda A : \text{tp}. \text{Unit} & & \\ \text{tm} & := \lambda A : \text{tp}. \lambda t : \text{tm}. t :: A & & \\ _ \times _ & := \dots \text{unit} & & \\ (_, _)^{-\times} & := \lambda A : \text{TypeEras}(\text{tp}). \lambda A^*. & & \end{array}$$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B: \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{l} \text{tp, tm: type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} \quad : \Pi A B: \text{tp}. \Pi a b: \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A \times B) \end{array} \right.$$

relation	TypePres:	TypeEras:	ChurchProd	\rightarrow	CurryProd	$=$
tp	$:= \lambda A: \text{tp}. \text{Unit}$					
tm	$:= \lambda A: \text{tp}. \lambda t: \text{tm}. t :: A$					
$_ \times _$	$:= \dots \text{unit}$					
$(_, _)^{-\times}$	$:= \lambda A: \text{tp}. \lambda A^*.$					

Type Preservation as a Logical Relation

$$\begin{aligned}
 \text{theory ChurchProd} &= \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B: \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\} \\
 \\
 \text{theory CurryProd} &= \left\{ \begin{array}{l} \text{tp, tm: type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} \quad : \Pi A B: \text{tp}. \Pi a b: \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A \times B) \end{array} \right\} \\
 \\
 \text{relation} \quad \text{TypePres: TypeEras: ChurchProd} &\rightarrow \text{CurryProd} \\
 \text{tp} &:= \lambda A: \text{tp}. \text{Unit} \\
 \text{tm} &:= \lambda A: \text{tp}. \lambda t: \text{tm}. t :: A \\
 _ \times _ &:= \dots \text{unit} \\
 (_, _)^{-\times} &:= \lambda A: \text{tp}. \lambda A^*. \lambda B: \text{TypeEras(tp)}. \lambda B^*.
 \end{aligned}$$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{l} \text{tp, tm} : \text{type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A, B) \end{array} \right.$$

relation	TypePres:	TypeEras:	ChurchProd	\rightarrow	CurryProd	$=$
tp	$:= \lambda A : \text{tp}. \text{Unit}$					
tm	$:= \lambda A : \text{tp}. \lambda t : \text{tm}. t :: A$					
$_ \times _$	$:= \dots \text{unit}$					
$(_, _)^{-\times}$	$:= \lambda A : \text{tp}. \lambda A^*. \lambda B : \text{tp}. \lambda B^*.$					

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{ll} \text{tp} & : \text{type} \\ \text{tm} & : \text{tp} \rightarrow \text{type} \\ _ \times _ & : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} & : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{ll} \text{tp, tm} : \text{type} & \\ _ :: _ & : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ & : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) & : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} & : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ & a :: A \rightarrow b :: B \rightarrow (a, b) :: (A, B) \end{array} \right.$$

$$\begin{array}{llll} \text{relation} & \text{TypePres:} & \text{TypeEras:} & \text{ChurchProd} \rightarrow \text{CurryProd} \\ \text{tp} & := \lambda A : \text{tp}. \text{Unit} & & \\ \text{tm} & := \lambda A : \text{tp}. \lambda t : \text{tm}. t :: A & & \\ _ \times _ & := \dots \text{unit} & & \\ (_, _)^{-\times} & := \lambda A : \text{tp}. \lambda A^*. \lambda B : \text{tp}. \lambda B^*. \\ & \lambda a : \text{TypeEras}(\text{tm } A). & & \end{array}$$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{l} \text{tp, tm} : \text{type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A, B) \end{array} \right.$$

relation	TypePres:	TypeEras:	ChurchProd	\rightarrow	CurryProd	$=$
tp	$:=$	$\lambda A : \text{tp}. \text{Unit}$				
tm	$:=$	$\lambda A : \text{tp}. \lambda t : \text{tm}. t :: A$				
$_ \times _$	$:=$	$\dots \text{unit}$				
$(_, _)^{-\times}$	$:=$	$\lambda A : \text{tp}. \lambda A^*.$	$\lambda B : \text{tp}. \lambda B^*.$			
		$\lambda a : \text{tm}.$				

Type Preservation as a Logical Relation

$$\begin{aligned}
 \text{theory ChurchProd} &= \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\} \\
 \\
 \text{theory CurryProd} &= \left\{ \begin{array}{l} \text{tp, tm} : \text{type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A, B) \end{array} \right. \\
 \\
 \text{relation} \quad \text{TypePres} : \text{TypeEras} : \text{ChurchProd} &\rightarrow \text{CurryProd} = \\
 \text{tp} &:= \lambda A : \text{tp}. \text{Unit} \\
 \text{tm} &:= \lambda A : \text{tp}. \lambda t : \text{tm}. t :: A \\
 _ \times _ &:= \dots \text{unit} \\
 (_, _)^{-\times} &:= \lambda A : \text{tp}. \lambda A^* : \text{tp}. \lambda B : \text{tp}. \lambda B^* : \text{tp}. \\
 &\quad \lambda a : \text{tm}. \lambda a^* : \text{tm}. a :: A \rightarrow a^* :: A^*
 \end{aligned}$$

Type Preservation as a Logical Relation

$$\begin{aligned}
 \text{theory ChurchProd} &= \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B : \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\} \\
 \\
 \text{theory CurryProd} &= \left\{ \begin{array}{l} \text{tp, tm} : \text{type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} : \Pi A B : \text{tp}. \Pi a b : \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A, B) \end{array} \right. \\
 \\
 \text{relation} \quad \text{TypePres} : \text{TypeEras} : \text{ChurchProd} &\rightarrow \text{CurryProd} \\
 \text{tp} &:= \lambda A : \text{tp}. \text{Unit} \\
 \text{tm} &:= \lambda A : \text{tp}. \lambda t : \text{tm}. t :: A \\
 _ \times _ &:= \dots \text{unit} \\
 (_, _)^{-\times} &:= \lambda A : \text{tp}. \lambda A^* : \text{tp}. \lambda B : \text{tp}. \lambda B^* : \text{tp}. \\
 &\quad \lambda a : \text{tm}. \lambda a^* : \text{tm}. a :: A \rightarrow a^* :: A^* \\
 &\quad \lambda b : \text{tm}. \lambda b^* : \text{tm}. b :: B \rightarrow b^* :: B^*
 \end{aligned}$$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B: \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{l} \text{tp, tm: type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} \quad : \Pi A B: \text{tp}. \Pi a b: \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A, B) \end{array} \right.$$

$$\begin{array}{llll} \text{relation} & \text{TypePres:} & \text{TypeEras:} & \text{ChurchProd} \quad \rightarrow \quad \text{CurryProd} \\ \text{tp} & := \lambda A: \text{tp}. \text{Unit} & & \\ \text{tm} & := \lambda A: \text{tp}. \lambda t: \text{tm}. t :: A & & \\ _ \times _ & := \dots \text{unit} & & \\ (_, _)^{-\times} & := \lambda A: \text{tp}. \lambda A^*. \lambda B: \text{tp}. \lambda B^*. \\ & \quad \lambda a: \text{tm}. \lambda a^*: a :: A. \\ & \quad \lambda b: \text{tm}. \lambda b^*: b :: B. & & \end{array}$$

Type Preservation as a Logical Relation

$$\text{theory ChurchProd} = \left\{ \begin{array}{l} \text{tp} \quad : \text{type} \\ \text{tm} \quad : \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _)^{-\times} \quad : \Pi A B: \text{tp}. \text{tm } A \rightarrow \text{tm } B \rightarrow \text{tm } A \times B \end{array} \right\}$$

$$\text{theory CurryProd} = \left\{ \begin{array}{l} \text{tp, tm: type} \\ _ :: _ \quad : \text{tm} \rightarrow \text{tp} \rightarrow \text{type} \\ _ \times _ \quad : \text{tp} \rightarrow \text{tp} \rightarrow \text{tp} \\ (_, _) \quad : \text{tm} \rightarrow \text{tm} \rightarrow \text{tm} \\ (_, _)^{T, -\times} \quad : \Pi A B: \text{tp}. \Pi a b: \text{tm}. \\ \qquad \qquad \qquad a :: A \rightarrow b :: B \rightarrow (a, b) :: (A$$

$$\begin{array}{llll} \text{relation} & \text{TypePres:} & \text{TypeEras:} & \text{ChurchProd} \quad \rightarrow \quad \text{CurryProd} \\ \text{tp} & := \lambda A: \text{tp}. \text{Unit} & & \\ \text{tm} & := \lambda A: \text{tp}. \lambda t: \text{tm}. t :: A & & \\ _ \times _ & := \dots \text{unit} & & \\ (_, _)^{-\times} & := \lambda A: \text{tp}. \lambda A^*. \lambda B: \text{tp}. \lambda B^*. \\ & \quad \lambda a: \text{tm}. \lambda a^*: a :: A. \\ & \quad \lambda b: \text{tm}. \lambda b^*: b :: B. (a, b)^{T, A \times B} a^* b^* & & \end{array}$$

Example 3: Summary

- formalized two flavors of type theories: ChurchProd, CurryProd
extrinsic vs. intrinsic
 $tm: tp \rightarrow type$ vs. $tm: type$
- saw **type erasure as a “lossy” translation**

```
view TypeEras: ChurchProd → CurryProd = {  
  tm := λ_. tm  
  :  
}
```

- used a logical relation to complement the morphism** to recover a meta theorem (within the system!)

```
relation TypePres: TypeEras: ChurchProd → CurryProd = {  
  tm := λA: tp. λt: tm. t :: A  
  :  
}
```

Overall Summary

Logical Relations:

- many type theories admit this proof technique:
 - ① map every type τ to an n -ary relation $C_\tau(-)$
 - ② prove that every constructor $f: \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \tau$ preserves the relation $C_\tau(-)$
- “the basic lemma”: if $t: \tau$, then $C_\tau(t)$

Concretely in MMT/LF: for morphisms $\mu_1, \dots, \mu_n: S \rightarrow T$

relation $r: \mu_1 \times \dots \times \mu_n: S \rightarrow T$

- special case for morphisms being identities:
 - TND: $id_{PL}: PL \rightarrow PL$: unary congruence proving $p: \text{prop} \implies \vdash p \vee \neg p$
 - LRA: $id_{PL} \times id_{PL}: PL \rightarrow PL$: (binary) congruence whose well-typedness shows \Leftrightarrow being a congruence
- special case

relation $\text{TypePres}: \text{TypeEras}: \text{ChurchProd} \rightarrow \text{CurryProd}$

- complements lossy translation TypeEras
- $\vdash_{\text{ChurchProd}} t: \text{tm } A \implies \vdash_{\text{CurryProd}} \text{wit}: \text{TypeEras}(t) :: \text{TypeEras}(A)$

Future Work

...

TODO

log relations incomplete wrt mapping to natural deduction calculus things
why do we need systematic renaming by priming?

def and basic lemma for binary congruence correct? how do congruence
properties follow from basic lemma?

TODO: Choose other letter than r for relation variables on previous slide!

Warning: slightly misrepresented logical relations as nothing more than
inductions

Further Pointers I

- Logical Relations:
 - refer to `[ahmed_log_rel]`² for an intro showing strong normalization and type safety for STLC
 - some interesting thoughts on relations with automata simulations: `[stackexchange_log_rel_and_simulations]`
- OMDoc/MT Language and Representation: `[rabe:howto:14; RabMue:WADT18]`
- MMT System: `[MueRab:rpfs19; ShaRab:dcm19; Rabe:MAGMS13]`
- Old LF papers: `[HarperEtAl:affdl93]`
- Diagram Operators: TODO insert WADT paper `[ShaRab:dcm19]`

²available as recorded videos; notes can be found at `[skorstengaard_log_rel]`