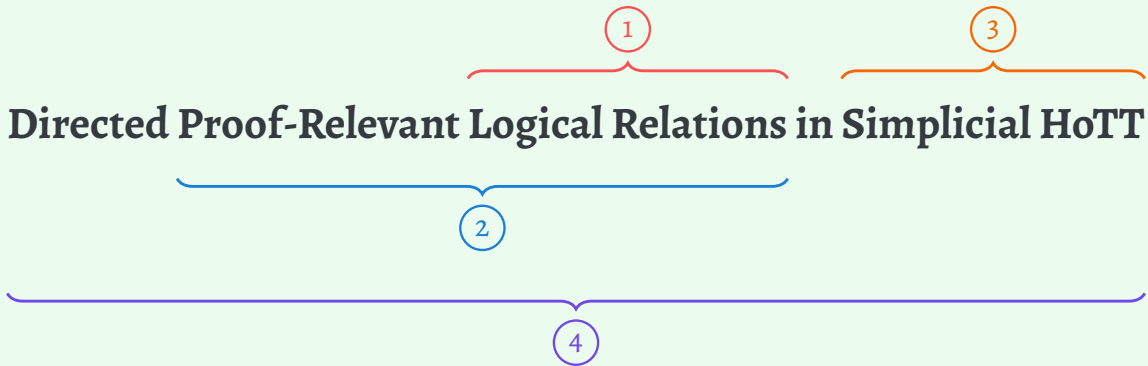


Directed Proof-Relevant Logical Relations in Simplicial HoTT

Speaking Skill — October 28, 2026

Runming Li

j.w.w. Harrison Grodin and Robert Harper



Type theory in type theory

representing /
proving properties of

^ Type theory in type theory

representing /
proving properties of

^ Type theory in type theory

a type theory Based
proof assistant

representing /
proving properties of

^ **Type theory** in **type theory**

How?

a type theory Based
proof assistant

representing /
proving properties of

^ **Type theory** in **type theory**

How?

a type theory Based
proof assistant

Which?

Write $\text{List}(n)$ for lists of length n .

$$f : (n : \text{Nat}) \rightarrow \text{List}(n + 1) \quad f(1) : \text{List}(1 + 1)$$

What is the type of $f(1)$?

$$\underbrace{1 + 1 \equiv 2}_{\beta\text{-computation}} \implies \underbrace{\text{List}(1 + 1) \equiv \text{List}(2)}_{\text{congruence}}$$

Hence $f(1) : \text{List}(2)$.

Judgments of type theory

$\Gamma \text{ ctx}$

$\Gamma \vdash A \text{ type}$

$\Gamma \vdash A \equiv A' \text{ type}$

$\Gamma \vdash M : A$

$\Gamma \vdash M \equiv M' : A$

$\Gamma \vdash M \equiv M : A$

$\Gamma \vdash M \equiv N : A \implies \Gamma \vdash N \equiv M : A$

$\Gamma \vdash M \equiv N : A, \Gamma \vdash N \equiv P : A \implies \Gamma \vdash M \equiv P : A$

$$\frac{\Gamma \vdash M \equiv M' : A \quad \Gamma \vdash N \equiv N' : B}{\Gamma \vdash \text{pair } M N \equiv \text{pair } M' N' : A \times B}$$

$\Gamma \vdash \text{fst} (\text{pair } M N) \equiv M : A$

$\Gamma \vdash \text{snd} (\text{pair } M N) \equiv N : B$

How do we represent those judgments and
judgmental equalities in a proof assistant?

Propositional equality

Proof assistants provide *propositional equality*: $x = y$ is a type of equality proofs.

Agda

```
trans : {x y z : A} →  
        x = y → y = z → x = z  
trans {x = x} p q = J (λ z _ → x = z) p q
```

Agda

```
cong : (f : A → B) {x y : A} →  
        x = y → f x = f y  
cong f {x = x} p = J (λ y _ → f x = f y) refl p
```

How about we use meta-language **propositional equality**
to represent object-language **judgmental equality**?

(Quotient) inductive syntax

Cubical Agda

Inductive **Ty** : **Type** where

...

× : **Ty** → **Ty** → **Ty**

Cubical Agda

Inductive **Tm** : **Ctx** → **Ty** → **Type** where

...

pair : **Tm** Γ **A** → **Tm** Γ **B** → **Tm** Γ (**A** × **B**)

fst : **Tm** Γ (**A** × **B**) → **Tm** Γ **A**

snd : **Tm** Γ (**A** × **B**) → **Tm** Γ **B**

$\times_{\beta_1}$: **fst** (**pair** a b) = a

$\times_{\beta_2}$: **snd** (**pair** a b) = b

$\times_{\eta}$: **pair** (**fst** p) (**snd** p) = p

...

} **Higher inductive types** let us add **equations** to ordinary inductive constructors.

We get the following definitional equalities for free:

Agda

```
refl-Tm :  
  {M : Tm Γ A} →  
    M = M  
refl-Tm = refl
```

Agda

```
sym-Tm :  
  {M N : Tm Γ A} →  
    M = N → N = M  
sym-Tm = sym
```

Agda

```
trans-Tm :  
  {M N P : Tm Γ A} →  
    M = N → N = P → M = P  
trans-Tm = trans
```

Agda

```
fst-cong :  
  {p q : Tm Γ (A × B)} →  
    p = q → fst p = fst q  
fst-cong = cong fst
```

Properties of definitional equality

Canonicity. If $\vdash M : \text{Bool}$, then

$\vdash M \equiv \text{true} : \text{Bool}$ or $\vdash M \equiv \text{false} : \text{Bool}$.

Agda

```
canonicity : (M : Tm ε Bool) →  
             (M = true) + (M = false)
```

Logical relations

A logical relation is a **family of predicates indexed by types**.

For closed terms, $A^\bullet(M)$ is a type of evidence about M .

$$A^\bullet : \text{Tm } \varepsilon A \rightarrow \text{Type}$$

$$\text{Bool}^\bullet(M) := (M = \text{true}) + (M = \text{false})$$

$$(A \times B)^\bullet(P) := A^\bullet(\text{fst } P) \times B^\bullet(\text{snd } P)$$

Logical relations

A logical relation is a **family of predicates indexed by types**.

For closed terms, $A^\bullet(M)$ is a type of evidence about M .

$$A^\bullet : \text{Tm } \varepsilon A \rightarrow \text{Type}$$

$$\text{Bool}^\bullet(M) := (M = \text{true}) + (M = \text{false})$$

$$(A \times B)^\bullet(P) := A^\bullet(\text{fst } P) \times B^\bullet(\text{snd } P)$$



..

The logical relation for the Boolean type looks like a Boolean.
The logical relation for a product type looks like a product.



Types carry predicates.

Agda

```
record TY : Type1 where
  field A○ : Ty
        A• : Tm ⇒ A○ → Type
```

Agda

```
BOOL : TY
BOOL○ = Bool
BOOL• M = (M = true) + (M = false)
```

Agda

```
PROD : TY → TY → TY
(PROD A B)○ = A○ × B○
(PROD A B)• P = A•(fst P) × B•(snd P)
```



Types carry predicates.
Terms carry evidence.

Agda

```
record TY : Type1 where
  field A○ : Ty
        A• : Tm ε A○ → Type
```

Agda

```
record TM (A : TY) : Type where
  field M○ : Tm ε A○
        M• : A•(M○)
```

Agda

```
BOOL : TY
BOOL○ = Bool
BOOL• M = (M = true) + (M = false)
```

Agda

```
PROD : TY → TY → TY
(PROD A B)○ = A○ × B○
(PROD A B)• P = A•(fst P) × B•(snd P)
```

Pairs and projections

Each operation acts on both the term and its evidence.

Agda

```
PAIR : TM A → TM B → TM (PROD A B)
```

```
(PAIR M N)○ = pair M○ N○
```

```
(PAIR M N)• = (M•, N•)
```

Pairs and projections

Each operation acts on both the term and its evidence.

Agda

```
PAIR : TM A → TM B → TM (PROD A B)
(PAIR M N)◦ = pair M◦ N◦
(PAIR M N)• = (M•, N•)
```

Agda

```
FST : TM (PROD A B) → TM A
(FST P)◦ = fst P◦
(FST P)• = π1(P•)
```

Agda

```
SND : TM (PROD A B) → TM B
(SND P)◦ = snd P◦
(SND P)• = π2(P•)
```



Terms carry evidence.
Does the evidence satisfy the equations too?

$$\text{FST}(\text{PAIR } M \ N) = M$$



.○

Terms carry evidence.
Does the evidence satisfy the equations too?

$$\text{FST } (\text{PAIR } M \ N) = M$$

On terms:

$$\begin{aligned} (\text{FST } (\text{PAIR } M \ N))^{\circ} &= \text{fst } ((\text{PAIR } M \ N)^{\circ}) && \text{by FST} \\ &= \text{fst } (\text{pair } M^{\circ} \ N^{\circ}) && \text{by PAIR} \\ &= M^{\circ} && \text{by } \beta_1 \text{ equation.} \end{aligned}$$



◦◦

Terms carry evidence.
Does the evidence satisfy the equations too?

$$\text{FST} (\text{PAIR } M \ N) = M$$

On terms:

$$\begin{aligned} (\text{FST} (\text{PAIR } M \ N))^{\circ} &= \text{fst} ((\text{PAIR } M \ N)^{\circ}) && \text{by FST} \\ &= \text{fst} (\text{pair } M^{\circ} \ N^{\circ}) && \text{by PAIR} \\ &= M^{\circ} && \text{by } \beta_1 \text{ equation.} \end{aligned}$$

On evidence:

$$\begin{aligned} (\text{FST} (\text{PAIR } M \ N))^{\bullet} &= \pi_1 ((\text{PAIR } M \ N)^{\bullet}) && \text{by FST} \\ &= \pi_1 (M^{\bullet}, N^{\bullet}) && \text{by PAIR} \\ &= M^{\bullet} && \text{by } \beta \text{ law.} \end{aligned}$$

Proof-relevance:

definitionally equal terms induce equal evidence.

Proof-relevance:

definitionally equal terms induce equal evidence.



The logical relation for the Boolean type
looks like a Boolean.

The logical relation for a product type
looks like a product.

Proof-relevance:

definitionally equal terms induce equal evidence.



The logical relation for the Boolean type
looks like a Boolean.

The logical relation for a product type
looks like a product.

For pairs, the evidence is a pair;
for `fst`, the evidence is first projection.
This evidence satisfies the β -equation too.

FTLR and canonicity

Fundamental theorem of logical relations (FTLR).

Every closed term lifts to a term with evidence.

$$M^\circ : \textcolor{red}{Tm} \, \varepsilon \, A^\circ \implies (M^\circ, M^\bullet) : \textcolor{blue}{TM} \, A \implies M^\bullet : A^\bullet(M^\circ)$$

FTLR and canonicity

Fundamental theorem of logical relations (FTLR).

Every closed term lifts to a term with evidence.

$$M^\circ : \textcolor{red}{Tm} \varepsilon A^\circ \implies (M^\circ, M^\bullet) : \textcolor{blue}{TM} A \implies M^\bullet : A^\bullet(M^\circ)$$

Specialize to $A = \textcolor{blue}{BOOL}$:

$$\begin{aligned} M^\bullet &: \textcolor{blue}{BOOL}^\bullet(M^\circ) \\ &= (M^\circ = \textcolor{red}{true}) + (M^\circ = \textcolor{red}{false}) \end{aligned}$$

Every closed Boolean term equals true or false: **canonicity**.

We represented **definitional equality** using **equality type** in the proof assistant.

$$\underbrace{M \equiv N}_{\text{the theory we study}} \rightsquigarrow \underbrace{M = N}_{\text{the proof assistant}}$$

We represented **definitional equality** using **equality type** in the proof assistant.

$$\underbrace{M \equiv N}_{\text{the theory we study}} \rightsquigarrow \underbrace{M = N}_{\text{the proof assistant}}$$

What can represents **reduction** in the proof assistant?

$$\underbrace{M \rightarrow_{\beta} N}_{\text{the theory we study}} \rightsquigarrow \underbrace{M ??? N}_{\text{the proof assistant}}$$

Reduction is **asymmetric**,
so we want an asymmetric “equality” in type theory.

Directed type theories

Equality

Agda

```
sort :  
  (xs : List Nat) →  
  mergeSort xs = insertSort xs
```

Directed type theories

Equality

Agda

```
sort :  
  (xs : List Nat) →  
  mergeSort xs = insertSort xs
```

Inequality

Directed type theory

```
monoid-hom :  
  (M N : Monoid) →  
  (M ≤ N) ≃ MonoidHom M N
```

Gratzer–Weinberger–Buchholtz, Lemma 7.11

Simplicial homotopy type theory

A TYPE THEORY FOR SYNTHETIC ∞ -CATEGORIES

EMILY RIEHL AND MICHAEL SHULMAN

ABSTRACT. We propose foundations for a synthetic theory of $(\infty, 1)$ -categories within homotopy type theory. First we develop a simplicial type theory within a new three-layered type theory with shapes, whose contexts are extended by polytopes within directed cubes, which can be abstracted over using “extension types” that generalize the path-types of cubical type theory. The simplices are then used to probe the internal categorical structures of types, allowing us to define *Segal types*, in which binary composites exist uniquely up to homotopy, and *Rezk types*, in which the categorical isomorphisms are additionally equivalent to the type-theoretic identities — a “local univalence” condition.

The single contractibility condition characterizing Segal types in fact ensures coherent categorical structure at all dimensions. In particular, from it we can prove that composition is associative and unital and develop a theory of functors and natural transformations. We then define *covariant fibrations*, which are type families varying functorially over a Segal type, and prove a “dependent Yoneda lemma” that can be viewed as a directed form of the usual elimination rule for identity types. We conclude by studying homotopically

Emily Riehl and Michael Shulman (2017)

How about we use meta-language **inequality types**
to represent object-language **reduction**?

Directed quotient inductive syntax

Simplicial type theory

Inductive $\mathsf{Tm} : \mathsf{Ctx} \rightarrow \mathsf{T}y \rightarrow \mathsf{Type}$ where

...

$\mathsf{pair} : \mathsf{Tm} \ \Gamma \ A \rightarrow \mathsf{Tm} \ \Gamma \ B \rightarrow \mathsf{Tm} \ \Gamma \ (A \times B)$

$\mathsf{fst} : \mathsf{Tm} \ \Gamma \ (A \times B) \rightarrow \mathsf{Tm} \ \Gamma \ A$

$\mathsf{snd} : \mathsf{Tm} \ \Gamma \ (A \times B) \rightarrow \mathsf{Tm} \ \Gamma \ B$

$\times_{\beta_1} : \mathsf{fst} \ (\mathsf{pair} \ a \ b) \leq a$

$\times_{\beta_2} : \mathsf{snd} \ (\mathsf{pair} \ a \ b) \leq b$

...

} **Directed constructors**
specify **reductions** using
inequality types.

We get reflexivity, transitivity, and monotonicity for free:

Simplicial type theory

```
refl-Tm :  
  {M : Tm  $\Gamma$  A}  $\rightarrow$   
  M  $\leq$  M  
refl-Tm = refl
```

Simplicial type theory

```
trans-Tm :  
  {M N P : Tm  $\Gamma$  A}  $\rightarrow$   
  M  $\leq$  N  $\rightarrow$  N  $\leq$  P  $\rightarrow$  M  $\leq$  P  
trans-Tm = trans
```

Simplicial type theory

```
fst-mono : {p q : Tm  $\Gamma$  (A  $\times$  B)}  $\rightarrow$  p  $\leq$  q  $\rightarrow$  fst p  $\leq$  fst q  
fst-mono = mono fst
```

Canonicity. If $\vdash M : \text{Bool}$, then

$M \rightarrow_{\beta} \text{true}$ or $M \rightarrow_{\beta} \text{false}$.

Simplicial type theory

```
canonicity : (M : Tm ε Bool) →  
             (M ≤ true) + (M ≤ false)
```

Simplicial type theory

 $\text{PROD} : \text{TY} \rightarrow \text{TY} \rightarrow \text{TY}$ $(\text{PROD } A \ B)^{\circ} = A^{\circ} \times B^{\circ}$ $(\text{PROD } A \ B)^{\bullet} P = A^{\bullet}(\text{fst } P) \times B^{\bullet}(\text{snd } P)$

Simplicial type theory

 $\text{PROD} : \text{TY} \rightarrow \text{TY} \rightarrow \text{TY}$ $(\text{PROD } A \ B)^\circ = A^\circ \times B^\circ$ $(\text{PROD } A \ B)^\bullet P = A^\bullet(\text{fst } P) \times B^\bullet(\text{snd } P)$

Simplicial type theory

 $\text{PAIR} : \text{TM } A \rightarrow \text{TM } B \rightarrow \text{TM } (\text{PROD } A \ B)$ $(\text{PAIR } M \ N)^\circ = \text{pair } M^\circ \ N^\circ$

$$(\text{PAIR } M \ N)^\bullet = \left(\underbrace{\quad ? \quad}, \underbrace{\quad ? \quad} \right)$$

$$A^\bullet(\text{fst } (\text{pair } M^\circ \ N^\circ)) \ B^\bullet(\text{snd } (\text{pair } M^\circ \ N^\circ))$$

Simplicial type theory

 $\text{PROD} : \text{TY} \rightarrow \text{TY} \rightarrow \text{TY}$ $(\text{PROD } A \ B)^\circ = A^\circ \times B^\circ$ $(\text{PROD } A \ B)^\bullet P = A^\bullet(\text{fst } P) \times B^\bullet(\text{snd } P)$

Simplicial type theory

 $\text{PAIR} : \text{TM } A \rightarrow \text{TM } B \rightarrow \text{TM } (\text{PROD } A \ B)$ $(\text{PAIR } M \ N)^\circ = \text{pair } M^\circ \ N^\circ$

$$(\text{PAIR } M \ N)^\bullet = \left(\underbrace{\quad\quad\quad}_?, \underbrace{\quad\quad\quad}_? \right)$$

$$A^\bullet(\text{fst } (\text{pair } M^\circ \ N^\circ)) \ B^\bullet(\text{snd } (\text{pair } M^\circ \ N^\circ))$$

Induction hypotheses give evidence

 $M^\bullet : A^\bullet(M^\circ)$ $N^\bullet : B^\bullet(N^\circ)$ β -reductions are $\text{fst } (\text{pair } M^\circ \ N^\circ) \leq M^\circ$ $\text{snd } (\text{pair } M^\circ \ N^\circ) \leq N^\circ$



Hmm, I've seen this before...



Hmm, I've seen this before...

Lemma (Closure under reverse reduction)

expansion : $(M \leq N) \rightarrow A^\bullet(N) \rightarrow A^\bullet(M)$



Hmm, I've seen this before...

Lemma (Closure under reverse reduction)

$\text{expansion} : (M \leqslant N) \rightarrow A^\bullet(N) \rightarrow A^\bullet(M)$

In a proof-relevant setting, we need to ask more:

$\text{expansion refl } e = e,$

$\text{expansion (trans } p \ q) \ e = \text{expansion } p \ (\text{expansion } q \ e).$

For $p : M \leqslant N$ and $q : N \leqslant P$.

Simplicial type theory already provides such a notion: **contravariance**.

Contravariant transport

Let $C : X \rightarrow \text{Type}$ be a contravariant family.

For $f : x \leq_x y$, there is a transport map

$$f^* : C(y) \rightarrow C(x),$$

such that

$$(\text{refl})^* v = v,$$

$$(\text{trans } f \ g)^* w = f^*(g^* w).$$

Here $g : y \leq_A z$, $v : C(x)$, and $w : C(z)$.

Simplicial type theory already provides such a notion: **contravariance**.

Contravariant transport

Let $C : X \rightarrow \text{Type}$ be a contravariant family.

For $f : x \leq_x y$, there is a transport map

$$f^* : C(y) \rightarrow C(x),$$

such that

$$(\text{refl})^* v = v,$$

$$(\text{trans } f \ g)^* w = f^*(g^* w).$$



Contravariant families are synthetic right fibrations.

Here $g : y \leq_A z$, $v : C(x)$, and $w : C(z)$.

Types carry contravariant predicates.

Simplicial type theory

record $\mathbf{TY} : \mathbf{Type}_1$ where

field $A^\circ : \mathbf{Ty}$

$A^\bullet : \mathbf{Tm} \ \varepsilon \ A^\circ \rightarrow \mathbf{Type}$

$c_A : \mathbf{isContrav} \ A^\bullet$

Simplicial type theory

 $\text{PROD} : \text{TY} \rightarrow \text{TY} \rightarrow \text{TY}$ $(\text{PROD } A \ B)^\circ = A^\circ \times B^\circ$ $(\text{PROD } A \ B)^\bullet P = A^\bullet(\text{fst } P) \times B^\bullet(\text{snd } P)$

Simplicial type theory

 $\text{PAIR} : \text{TM } A \rightarrow \text{TM } B \rightarrow \text{TM } (\text{PROD } A \ B)$ $(\text{PAIR } M \ N)^\circ = \text{pair } M^\circ \ N^\circ$

$$(\text{PAIR } M \ N)^\bullet = \left(\underbrace{(\times_{\beta_1})^* M^\bullet}_{A^\bullet(\text{fst } (\text{pair } M^\circ \ N^\circ))}, \underbrace{(\times_{\beta_2})^* N^\bullet}_{B^\bullet(\text{snd } (\text{pair } M^\circ \ N^\circ))} \right)$$

The transports have exactly the required types:

$$(\times_{\beta_1})^* : A^\bullet(M^\circ) \rightarrow A^\bullet(\text{fst } (\text{pair } M^\circ \ N^\circ))$$

$$(\times_{\beta_2})^* : B^\bullet(N^\circ) \rightarrow B^\bullet(\text{snd } (\text{pair } M^\circ \ N^\circ))$$

As before, we want the evidence to satisfy the β -inequality as well:

$$\text{FST}(\text{PAIR } M \ N) \leq M$$

As before, we want the evidence to satisfy the β -inequality as well:

$$\text{FST } (\text{PAIR } M \ N) \leqslant M$$

Want to show:

$$\begin{aligned} (\text{FST } (\text{PAIR } M \ N))^{\bullet} &= \pi_1((\text{PAIR } M \ N)^{\bullet}) && \text{by FST} \\ &= \pi_1((\times_{\beta_1})^* M^{\bullet}, (\times_{\beta_2})^* N^{\bullet}) && \text{by PAIR} \\ &= (\times_{\beta_1})^* M^{\bullet} && \text{by } \beta \text{ law} \\ &\leqslant_{A^{\bullet}(\times_{\beta_1})} M^{\bullet} && \text{want to show} \end{aligned}$$

As before, we want the evidence to satisfy the β -inequality as well:

$$\text{FST } (\text{PAIR } M \ N) \leq M$$

Want to show:

$$\begin{aligned} (\text{FST } (\text{PAIR } M \ N))^{\bullet} &= \pi_1((\text{PAIR } M \ N)^{\bullet}) && \text{by FST} \\ &= \pi_1((\times_{\beta_1})^* M^{\bullet}, (\times_{\beta_2})^* N^{\bullet}) && \text{by PAIR} \\ &= (\times_{\beta_1})^* M^{\bullet} && \text{by } \beta \text{ law} \\ &\leq_{A^{\bullet}(\times_{\beta_1})} M^{\bullet} && \text{want to show} \end{aligned}$$

Lemma (Universal property of contravariance)

For a contravariant family $C : X \rightarrow \text{Type}$, $f : x \leq y$, $u : C(x)$, and $v : C(y)$,

$$(u \leq_{C(f)} v) \simeq (u = f^* v).$$

As before, we want the evidence to satisfy the β -inequality as well:

$$\text{FST } (\text{PAIR } M \ N) \leq M$$

Want to show:

$$\begin{aligned} (\text{FST } (\text{PAIR } M \ N))^{\bullet} &= \pi_1((\text{PAIR } M \ N)^{\bullet}) && \text{by FST} \\ &= \pi_1((\times_{\beta_1})^* M^{\bullet}, (\times_{\beta_2})^* N^{\bullet}) && \text{by PAIR} \\ &= (\times_{\beta_1})^* M^{\bullet} && \text{by } \beta \text{ law} \\ &\leq_{A^{\bullet}(\times_{\beta_1})} M^{\bullet} && \text{want to show} \end{aligned}$$

Lemma (Universal property of contravariance)

For a contravariant family $C : X \rightarrow \text{Type}$, $f : x \leq y$, $u : C(x)$, and $v : C(y)$,

$$(u \leq_{C(f)} v) \simeq (u = f^* v).$$

$$((\times_{\beta_1})^* M^{\bullet} \leq_{A^{\bullet}(\times_{\beta_1})} M^{\bullet}) \simeq ((\times_{\beta_1})^* M^{\bullet} = (\times_{\beta_1})^* M^{\bullet}).$$

Contravariance under reindexing

If $C : Y \rightarrow \text{Type}$ is contravariant and $g : X \rightarrow Y$, then $C \circ g$ is contravariant.

$$x \mapsto C(g(x))$$

Contravariance under reindexing**Contravariance under products**

If

co

 x

If $C, D : X \rightarrow \text{Type}$ are contravariant, their pointwise product is contravariant.

$$x \mapsto C(x) \times D(x)$$

Contravariance under reindexing**Contravariance under products****Contravariance under dependent products**

If $x \mapsto C(x, y)$ is contravariant for every $y : Y$, then the family of dependent functions is contravariant.

$$x \mapsto ((y : Y) \rightarrow C(x, y))$$

Contravariance under reindexing**Contravariance under products****Contravariance under dependent products****Contravariance under dependent sums**

If $C : X \rightarrow \text{Type}$ and $D : (\sum_{x:X} C(x)) \rightarrow \text{Type}$ are contravariant, so is their dependent sum.

$$x \mapsto \sum_{c:C(x)} D(x, c)$$

Contravariance under reindexing**Contravariance under products****Contravariance under dependent products****Contravariance under dependent sums****Contravariance over a discrete base**

If X is discrete and $C : X \rightarrow \text{Type}$ has discrete fibers, then C is contravariant.

`isContrav(C)`

Contravariance under reindexing

Contravariance under products

Contravariance under dependent products

Contravariance under dependent sums

Contravariance over a discrete base

Representable contravariant families

If X is Segal and $a : X$, the representable family on a is contravariant.

$$x \mapsto (x \leqslant_X a)$$

Contravariance under reindexing

Contravariance under products

Contravariance under dependent products

Contravariance under dependent sums

Contravariance over a discrete base

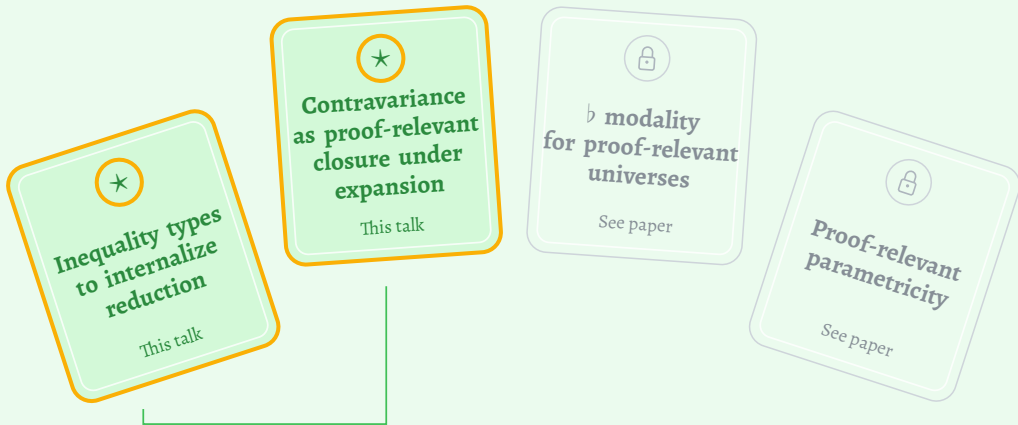
Representable contravariant families

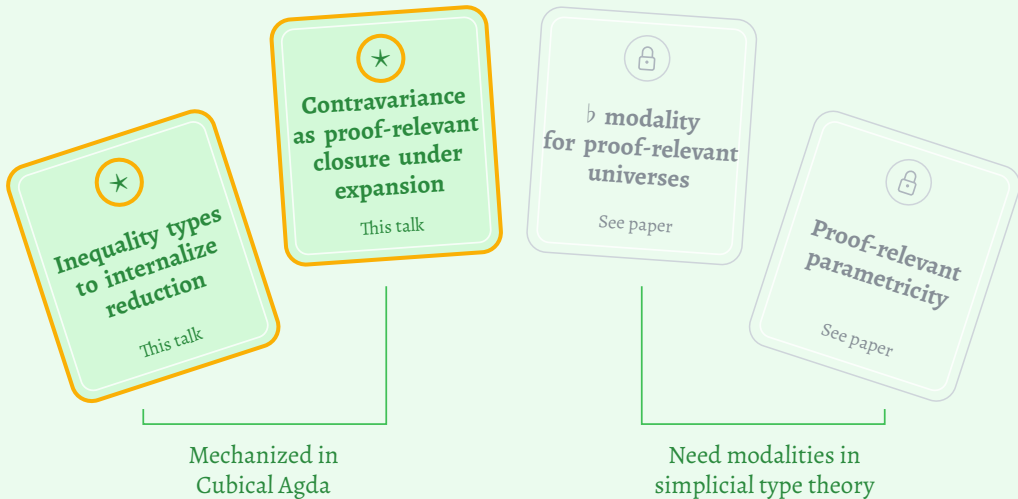
If X is Segal and $a : X$, the representable family on a is contravariant.

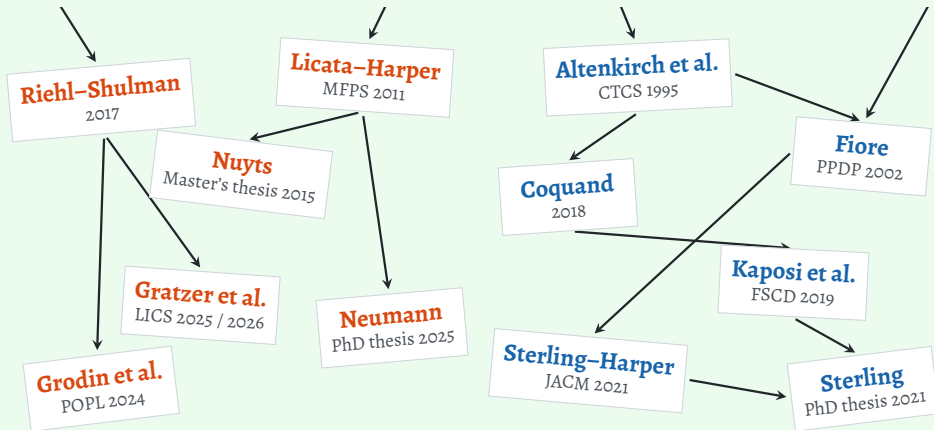
$$x \mapsto (x \leq_X a)$$

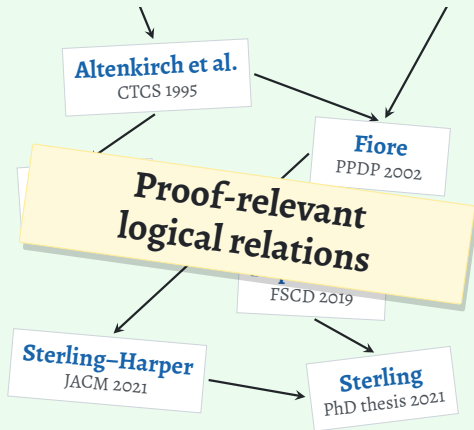
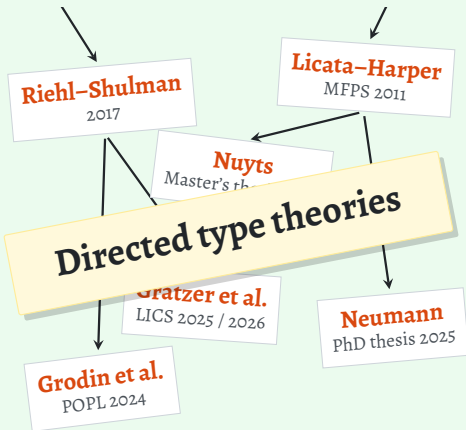


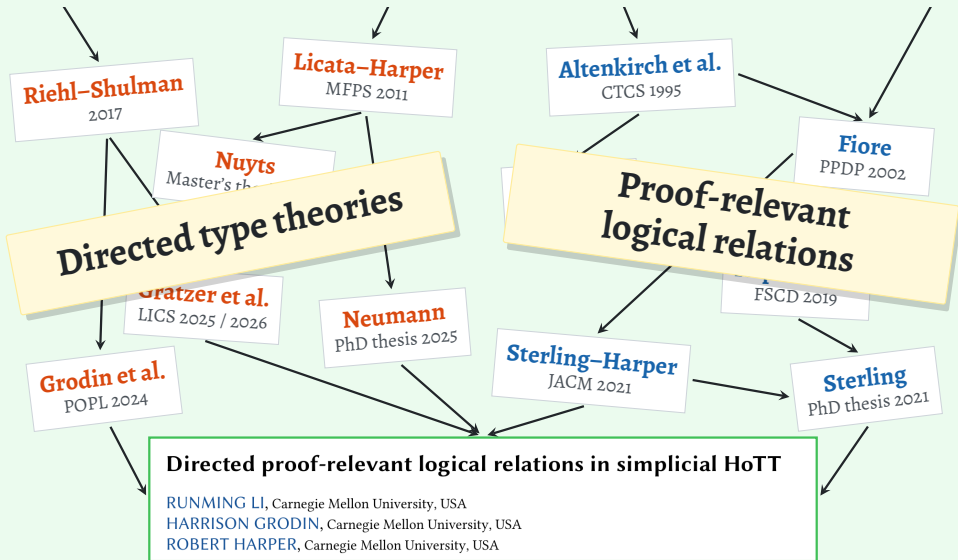
Simplicial type theory provides the
"standard library" for closure under
reverse reduction in logical relations.











Conditionally accepted at POPL 2027