# Compiling HoTT with Lean: Syntax and Interpretation

**Wojciech Nawrocki**[1] with Steve Awodey[1], Mario Carneiro[2], Sina Hazratpour[3], Joseph Hua[1], Spencer Woolfson[1], and Yiming Xu[4]

[1] Carnegie Mellon University [2] Chalmers University of Technology [3] Harmonic AI
[4] LMU Munich

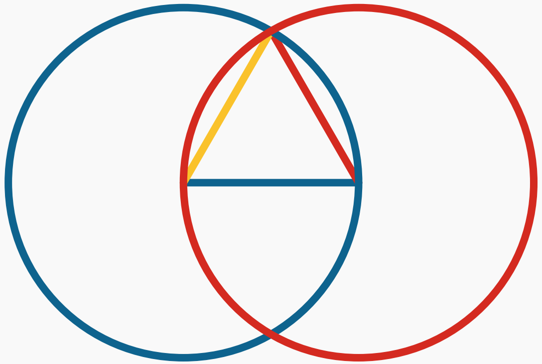
ASL North American Meeting | Special Session on Proof Assistants

May 15th, 2025

Slides at voidma.in/asl.pdf

# Synthetic (axiomatic) mathematics

Axiomatize an area of inquiry by viewing its basic objects as irreducible primitives.





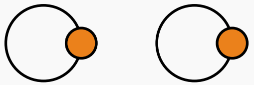
$$P \neq Q : \text{Point}$$
$$\exists (C_1 : \text{Circle}).\ \text{Center}(P, C_1) \wedge Q \in C_1$$
$$\exists (C_2 : \text{Circle}).\ \text{Center}(Q, C_2) \wedge P \in C_2$$
$$\text{Intersect}(C_1, C_2)$$

$$P \neq Q \in \mathbb{R}^2$$
$$C_1 = \left\{(x, y) \mid (x + \dots)^2 + (y + \dots)^2 = \dots\right\}$$
$$C_2 = \left\{(x, y) \mid (x + \dots)^2 + (y + \dots)^2 = \dots\right\}$$
$$\exists R \in C_1 \cap C_2$$

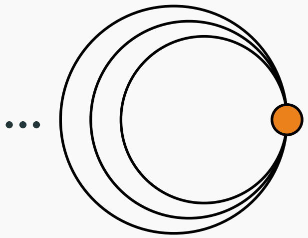
# Synthetic abstract homotopy theory

Homotopy type theory (HoTT) is an axiomatization of abstract homotopy theory. In many interesting models, types are ∞-groupoids.

```
data Bool : Type where
  false true : bool
```

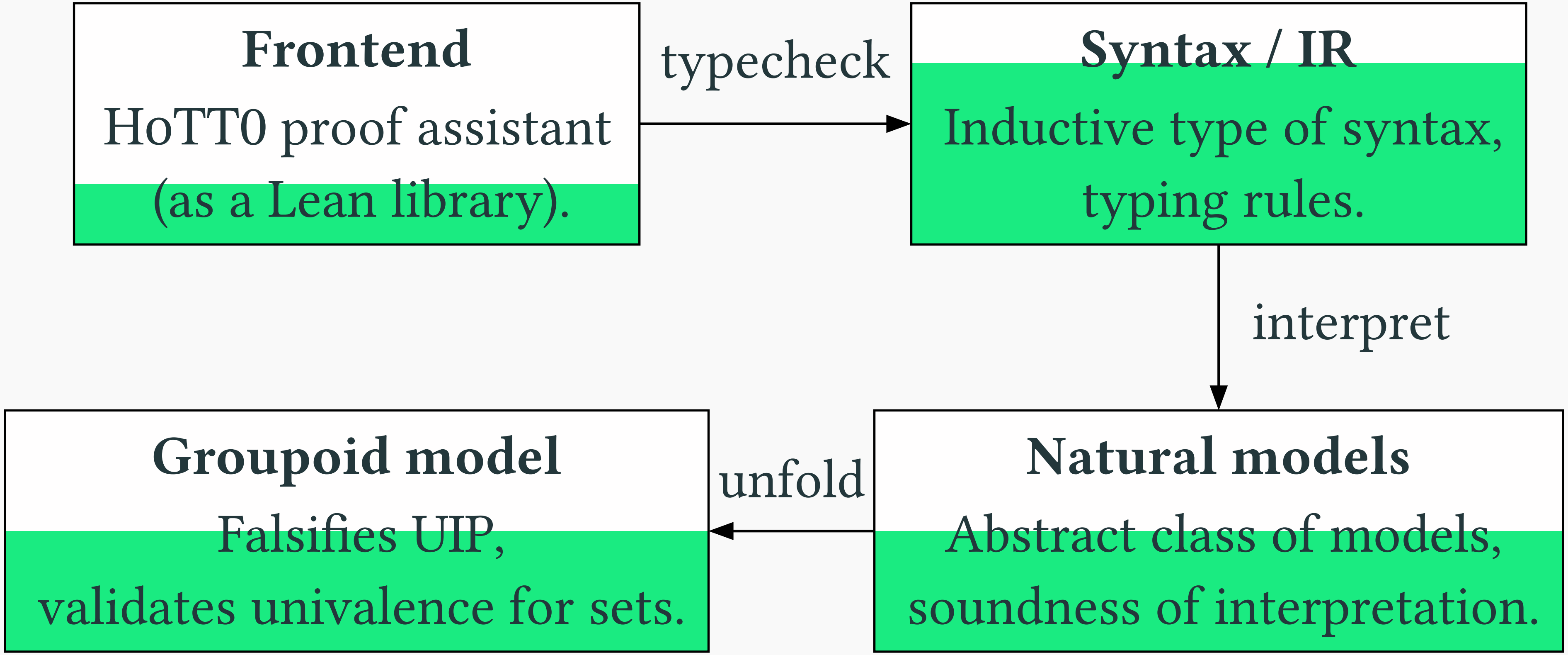


```
data S¹ : Type where
  base : S¹
  loop : base ≡ base
```




Our goal: formally verified theory-model connection.

# Project pipeline



# Syntax: PER-style MLTT with finite universe hierarchy

$\boxed{\Gamma \vdash_\ell A \equiv B}$

$$\frac{\ell < \ell_{\max}}{\Gamma \vdash_{\ell+1} U_\ell \equiv U_\ell} \qquad \frac{\Gamma \vdash_{\ell+1} \mathrm{a} \equiv \mathrm{b} : U_\ell}{\Gamma \vdash_\ell \mathrm{el}\ \mathrm{a} \equiv \mathrm{el}\ \mathrm{b}}$$

$\boxed{\Gamma \vdash_\ell t \equiv u : A}$

$$\frac{\ell < \ell_{\max} \qquad \Gamma \vdash_\ell \mathrm{A} \equiv \mathrm{B}}{\Gamma \vdash_{\ell+1} \mathrm{code}\ A \equiv \mathrm{code}\ \mathrm{B} : U_\ell}$$

$\boxed{\Gamma \vdash_\ell A} \triangleq \Gamma \vdash_\ell A \equiv A$ $\qquad$ $\boxed{\Gamma \vdash_\ell t : A} \triangleq \Gamma \vdash_\ell t \equiv t : A$

# Semantics: natural models in $[\mathcal{C}^{\mathrm{op}}, \mathrm{Set}]$

Suppose $\Gamma \vdash_0 A$. Then the extended context $\vdash \Gamma.A$ is well-formed.

Separately, $\cdot \vdash_1 U_0$.

# Semantics: natural models in $[\mathcal{C}^{op}, \mathrm{Set}]$

Suppose $\Gamma \vdash_0 A$. Then the extended context $\vdash \Gamma.A$ is well-formed.

Separately, $\cdot \vdash_1 U_0$.

$$\begin{array}{c} \mathrm{Tm}_0 \\ \downarrow \\ \mathrm{Ty}_0 \end{array}$$

# Semantics: natural models in $[\mathcal{C}^{\mathrm{op}}, \mathrm{Set}]$

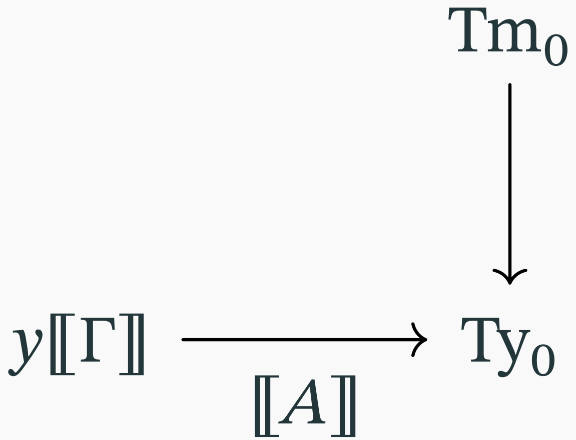
Suppose $\Gamma \vdash_0 A$. Then the extended context $\vdash \Gamma.A$ is well-formed.

Separately, $\cdot \vdash_1 U_0$.

$$\begin{array}{ccc} & & \mathrm{Tm}_0 \\ & & \downarrow \\ y[\![\Gamma]\!] & & \mathrm{Ty}_0 \end{array}$$

Suppose $\Gamma \vdash_0 A$. Then the extended context $\vdash \Gamma.A$ is well-formed.

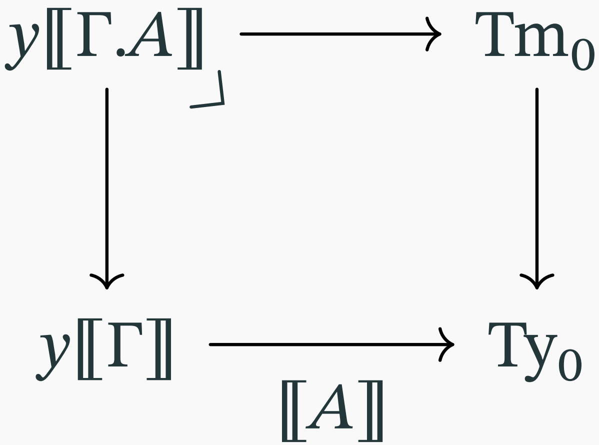
Separately, $\cdot \vdash_1 U_0$.



# Semantics: natural models in $[\mathcal{C}^{\mathrm{op}}, \mathrm{Set}]$

Suppose $\Gamma \vdash_0 A$. Then the extended context $\vdash \Gamma.A$ is well-formed.

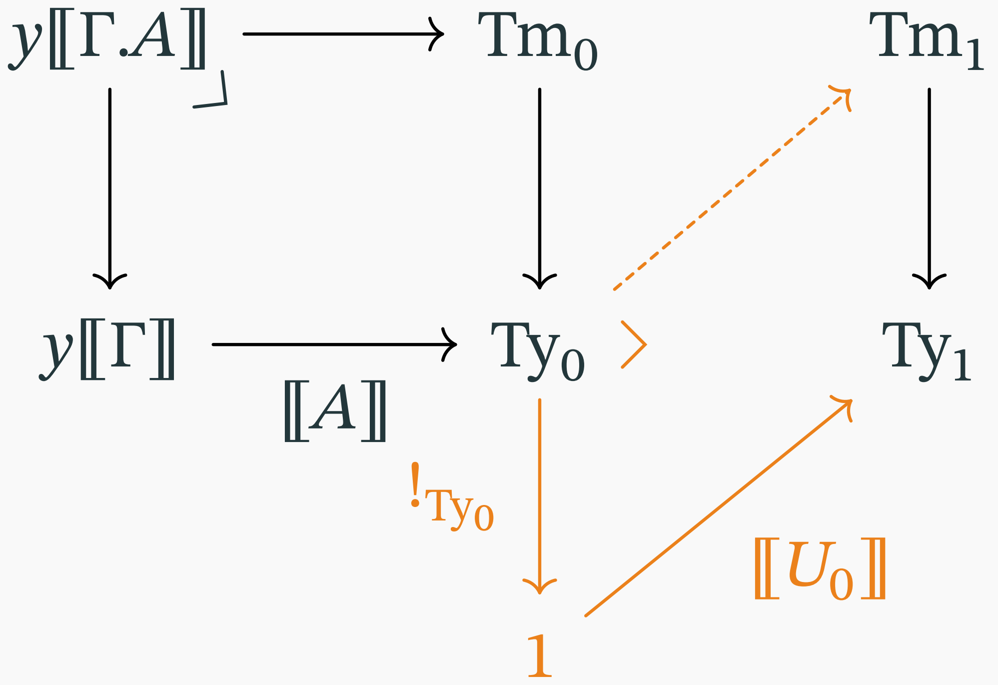
Separately, $\cdot \vdash_1 U_0$.



# Semantics: natural models in $[\mathcal{C}^{op}, \mathrm{Set}]$

Suppose $\Gamma \vdash_0 A$. Then the extended context $\vdash \Gamma.A$ is well-formed.

Separately, $\cdot \vdash_1 U_0$.



# Partial interpretation

$[\![\Gamma \vdash_\ell A]\!] : (\Gamma : \text{List Expr})\,(\ell : \mathbb{N})\,(A : \text{Expr}) \to (\Gamma \vdash_\ell A) \to \text{Hom}(y[\![\Gamma]\!], \text{Ty}_\ell)$

❌ Recursive-recursive! Not definable in Lean.

Instead, define $[\![A]\!]_{X,\ell} : (A : \text{Expr})\,(X : \mathcal{C})\,(\ell : \mathbb{N}) \rightharpoonup \text{Hom}(yX, \text{Ty}_\ell)$

✅ Soundness: if $\Gamma \vdash_\ell A \equiv B$, then $[\![A]\!]_{[\![\Gamma]\!],\ell}\downarrow$ and $[\![A]\!]_{[\![\Gamma]\!],\ell} = [\![B]\!]_{[\![\Gamma]\!],\ell}$

# Frontend: domain-specific language

```
hott def idfun : Π {A : Type}, A → A := fun a ⇒ a
```

# Frontend: domain-specific language

```
hott def idfun : Π {A : Type}, A → A := fun a ⇒ a

-- { l := 1,
--   val := lam 1 0 (univ 0) (lam 0 0 (el (bvar 0)) (bvar 0)),
--   tp := pi 1 0 (univ 0) (pi 0 0 (el (bvar 0)) (el (bvar 1))),
--   wf := (⋯ : [] ⊢[l] val ≡ val : tp) }
#eval! idfun.checked
```

# Frontend: domain-specific language

```
hott def idfun : Π {A : Type}, A → A := fun a ⇒ a

-- { l := 1,
--   val := lam 1 0 (univ 0) (lam 0 0 (el (bvar 0)) (bvar 0)),
--   tp := pi 1 0 (univ 0) (pi 0 0 (el (bvar 0)) (el (bvar 1))),
--   wf := (⋯ : [] ⊢[l] val ≡ val : tp) }
#eval! idfun.checked

noncomputable def idfun.grpd :
    ⊤_ _ ⟶ GroupoidModel.Ctx.ofCategory.{1,4} Grpd.{1,1} :=
  (uHomSeqPis.interpType ⋯
    idfun.checked.wf.wf_tp ⋯ uHomSeqPis.nilCObj ⋯).app (.op ⊲ ⊤_ _) (𝟙 _)
```

# Translation & typechecking

```
partial def translate : Lean.Expr → TranslateM Q(HoTT.Expr)

partial def checkEqTm (Γ : HoTT.Ctx) (l : ℕ) (t u A : HoTT.Expr) :
  Except String Q($Γ ⊢[$l] $t ≡ $u : $A)

elab "hott def" name ":" tp ":=" val : command => do
  let tpE ← Lean.Elab.Term.elabTerm tp
  let valE ← Lean.Elab.Term.elabTerm tpE val
  let tpHott ← translate tpE
  let valHott ← translate valE
  let pf ← checkEqTm [] 0 valHott valHott tpHott
```

# Challenges

- Proof checking performance
- Formalizing "obvious" naturality laws (see Zulip discussion)
- Non-cumulativity of Lean universes (`A : Type 0` $\not\Rightarrow$ `A : Type 1`)
- Sparsity of prior work

  Sozeau/Tabareau. *Towards an Internalization of the Groupoid Interpretation of Type Theory*. TYPES 2014.

# Thank you

github.com/sinhp/groupoid_model_in_lean4