Certifying Synthetic Mathematics in Lean

Wojciech Nawrocki[1] with Joseph Hua[1], Mario Carneiro[2], Yiming Xu[3], Spencer Woolfson[4], Shuge Rong[1], Sina Hazratpour[5], and Steve Awodey[1]

[1] Carnegie Mellon University [2] Chalmers University of Technology
[3] LMU Munich [4] Chapman University [5] Stockholm University

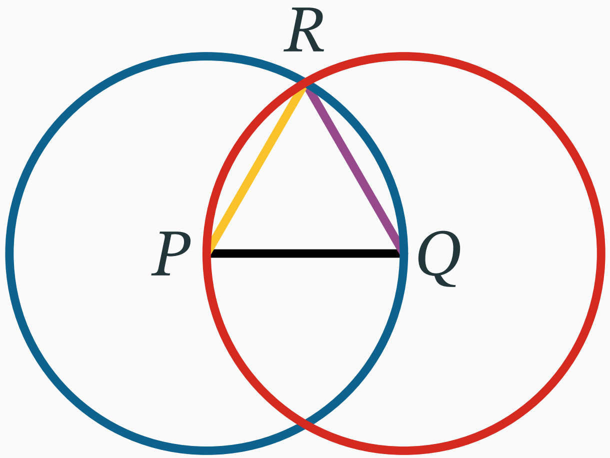
February 19th, 2026 | Formalisation of maths with ITPs (seminar series) | Cambridge, UK

This material is based upon work supported by the Air Force Office of Scientific Research under MURI award FA9550-21-1-0009, the National Science Foundation under grant number 2434614, and the European Union (ERC, Nekoka, 101083038).

Synthetic vs. analytic geometry

Proposition (Euclid, ~300BC). Given a finite line segment PQ, there exists an equilateral triangle with PQ as one side.

Euclid's synthetic proof	Descartes' analytic proof



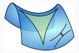
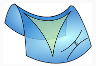
$$P = \begin{pmatrix} x_P \\ y_P \end{pmatrix} \qquad Q = \begin{pmatrix} x_Q \\ y_Q \end{pmatrix}$$

$$R = P + \begin{pmatrix} \cos 60° & -\sin 60° \\ \sin 60° & \cos 60° \end{pmatrix} (Q - P)$$

$$\|R - P\| = \|Q - P\| = \|R - Q\|$$

Theory-based vs. model-based reasoning

Let φ be Euclid's proposition as the first-order sentence "$\forall(P \neq Q : \mathsf{Point}).\ \exists(R : \mathsf{Point}).\ (\ldots)$" in the language of *geometry*.

Provability in the theory		Truth in a model
$\mathbb{T}_{\text{geom.}} \vdash \varphi$	soundness and $\mathbb{R}^2 \vDash \mathbb{T}_{\text{geom.}}$ $\Longrightarrow$	$\mathbb{R}^2 \vDash \varphi$
	soundness and  $\vDash \mathbb{T}_{\text{geom.}}$ $\Longrightarrow$	 $\vDash \varphi$

What are we *really* proving?

```
theorem one : 2 + 2 = 4 :=
  rfl

theorem two (n : ℕ) : n + 0 = n :=
  rfl

theorem three : FermatLastTheorem :=
  big_proof

theorem four (Γ₁) (Γ₂) … (Γₙ) : A :=
  t
#print axioms four
-- 'four' depends on axioms 𝕋
```

$\cdot \vdash_{\text{L∃∀N}-} \text{rfl} : 2 + 2 = 4$

$n : \mathbb{N} \vdash_{\text{L∃∀N}-} \text{rfl} : n + 0 = n$

$\cdot \vdash_{\text{L∃∀N}} \text{big\_proof} : \text{FLT}$

$\Gamma \vdash_{\mathbb{T}} t : A$

Types as groupoids

The set-based model of type theory is not the only one!

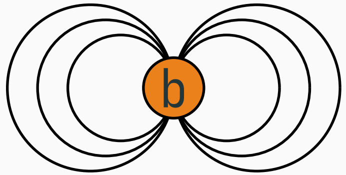
```
inductive Bool : Type
  | true | false
```

$(\cdot \vdash_{\text{L∃∀N}^-} \text{Bool} : \text{Type})$

$\xRightarrow{\textbf{Set} \vDash \text{L∃∀N}}$ $[\![\text{Bool}]\!] = \{\text{true}, \text{false}\}$

$\xRightarrow{\textbf{Gpd} \vDash \text{L∃AN}^-}$ $[\![\text{Bool}]\!] =$ 


```
inductive S¹ : Type
  | base : S¹
  | loop : Path base base
```

$\xRightarrow{\textbf{Gpd} \vDash \text{HoTT}_0}$ $[\![S^1]\!] = \cdots$  $\cdots$

Our approach

1. Formalize syntax and provability ($\Gamma \vdash_{\mathbb{T}} t : A$).
2. *Reflect* definitions `def foo : A := t` as Lean proofs of $\cdot \vdash_{\mathbb{T}} t : A$.
3. Formalize semantics ($\mathcal{M} \vDash \mathbb{T}$).
4. Use (2.) to reason about $\mathcal{M}$ using $\mathbb{T}$.

Usage of SynthLean

```
@[reflect] axiom X : Type
@[reflect] axiom p : X

#print p.reflection
-- def p.reflection : ReflectedAx [X] :=
--   { tp := .el X.reflection.val,
--     wf_tp := (⋯ : [] ⊢[X] tp),
--     … }

@[reflect] def q : X := p
```

```
def M : Model := ⋯

def M_X : 1_ _ → M.Ty := ⋯
def M_p : 1_ _ → M.Tm := ⋯

def I : Interpretation [X,p] M :=
  ax := fun
    | ``X => M_X
    | ``p => M_p

example : I.intp q.reflection = M_p
```

Disclaimer: the API is oversimplified here.

Technical details

Components

1. Deep embedding of Martin-Löf type theories with U, Π, Σ, Id types, and base constants, e.g. HoTT (without inductive types).

2. SynthLean, an embedded proof assistant for (1.).

3. Natural model semantics (cf. CwFs) and their soundness for the syntax.

MLTTs stratified by universe level

$\vdash_{\mathbb{T}} \Gamma$ $\quad$ $\Gamma \vdash_{\mathbb{T}}^{\ell} A$ $\quad$ $\Gamma \vdash_{\mathbb{T}}^{\ell} t : A$

$\Gamma \vdash_{\mathbb{T}}^{\ell} A \equiv B$ $\quad$ $\Gamma \vdash_{\mathbb{T}}^{\ell} t \equiv u : A$

$$\frac{\ell < \ell_{\text{max}}}{\Gamma \vdash_{\mathbb{T}}^{\ell+1} U_{\ell}}$$

$$\frac{\Gamma \vdash_{\mathbb{T}}^{\ell} A \qquad \Gamma.A \vdash_{\mathbb{T}}^{\ell'} B}{\Gamma \vdash_{\mathbb{T}}^{\max(\ell,\ell')} \Pi A.\, B} \qquad \frac{\cdots \qquad \mathbb{T}(c) = (A, \ell)}{\Gamma \vdash_{\mathbb{T}}^{\ell} c : A}$$

MLTTs stratified by universe level

$\vdash_{\mathbb{T}} \Gamma$ $\quad$ $\Gamma \vdash^{\ell}_{\mathbb{T}} A$ $\quad$ $\Gamma \vdash^{\ell}_{\mathbb{T}} t : A$

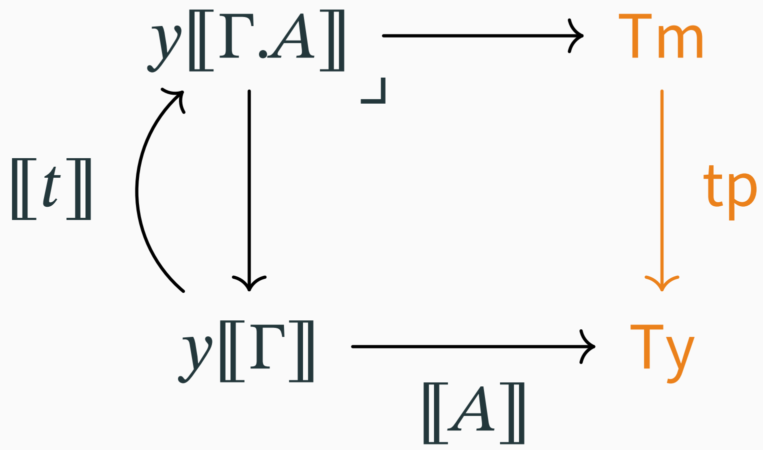
$\Gamma \vdash^{\ell}_{\mathbb{T}} A \equiv B$ $\quad$ $\Gamma \vdash^{\ell}_{\mathbb{T}} t \equiv u : A$

```
inductive Expr (χ : Type u) where
  | univ l | pi A B | ...

mutual
inductive WfCtx : Theory χ → List (Expr χ) → Prop
inductive WfTp : Theory χ → List (Expr χ) → Nat → Expr χ → Prop
...
```

A **natural model universe** in $[\mathbf{Ctx}^{\mathrm{op}}, \mathbf{Set}]$

Let $\Gamma \vdash A$ be a type, $\vdash \Gamma.A$ the extended context, and $\Gamma \vdash t : A$ a term.




```
structure Universe where
  Tm : Psh Ctx
  Ty : Psh Ctx
  tp : Tm ⟶ Ty
  ext {Γ : Ctx} (A : y(Γ) ⟶ Ty) : Ctx
  ...
```

Interpretation of syntax in semantics

Following Streicher [2:Ch. III], define

$$[\![\Gamma]\!] : (\Gamma : \mathsf{List\ Expr}) \rightharpoonup \mathbf{Ctx}$$
$$[\![A]\!]_X : (A : \mathsf{Expr})\,(X : \mathbf{Ctx}) \rightharpoonup [\mathbf{Ctx}^{\mathrm{op}}, \mathbf{Set}](yX, \mathsf{Ty})$$

by recursion on raw expressions. For typing contexts:

$$[\![\cdot]\!] = \mathbf{1} \qquad [\![\Gamma.A]\!] = \begin{array}{l} \mathsf{let}\ X \leftarrow [\![\Gamma]\!]\ \ \mathsf{in} \\ \mathsf{let}\ f \leftarrow [\![A]\!]_X\ \mathsf{in} \\ \mathsf{ext}(f) \end{array}$$

Soundness of interpretation

Theorem.

If $\Gamma \vdash A$, then $[\![A]\!]_{[\![\Gamma]\!]}\downarrow$.

If $\Gamma \vdash A \equiv B$, then $[\![A]\!]_{[\![\Gamma]\!]} = [\![B]\!]_{[\![\Gamma]\!]}$.

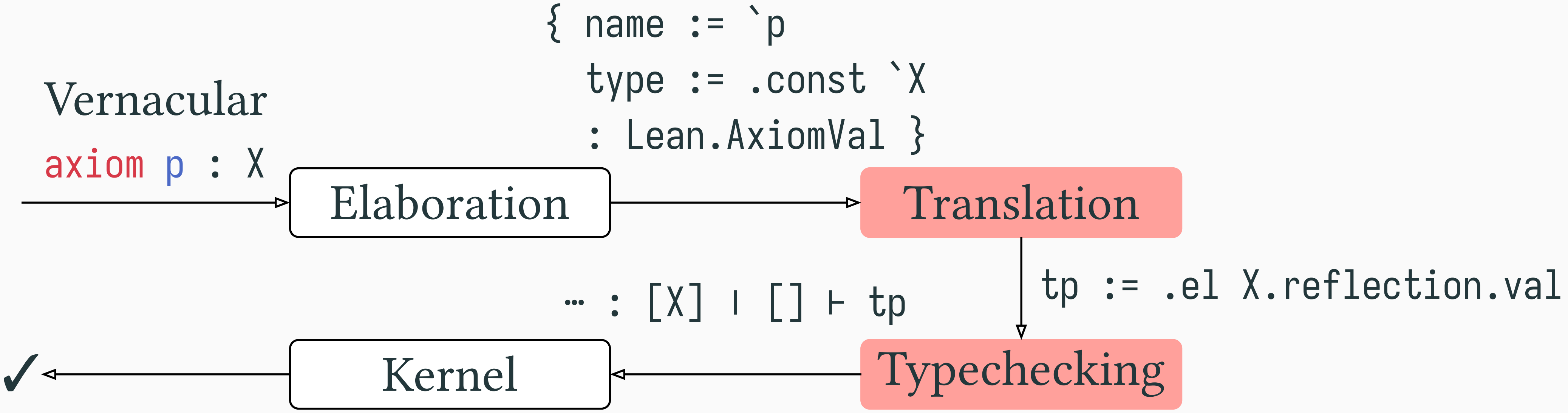
Previously De Boer/Brunerie/Lumsdaine/Mörtberg [1] in Agda.

Certificate-producing proof assistant

```
@[reflect] axiom X : Type
@[reflect] axiom p : X
```

SynthLean ⇝

```
def p.reflection : ReflectedAx [X] :=
  { tp := SynthLean.el X.reflection.val,
    wf_tp := (⋯ : [X] ı [] ⊢ tp),
    … }
```



Open problems

Internal reasoning in the groupoid model

```
def isProp (A : Type) := (a b : A) → Path a b
def isSet (A : Type) := (a b : A) → isProp (Path a b)
def isGrpd (A : Type) := (a b : A) → isSet (Path a b)
def is2Grpd (A : Type) := (a b : A) → isGrpd (Path a b)
```

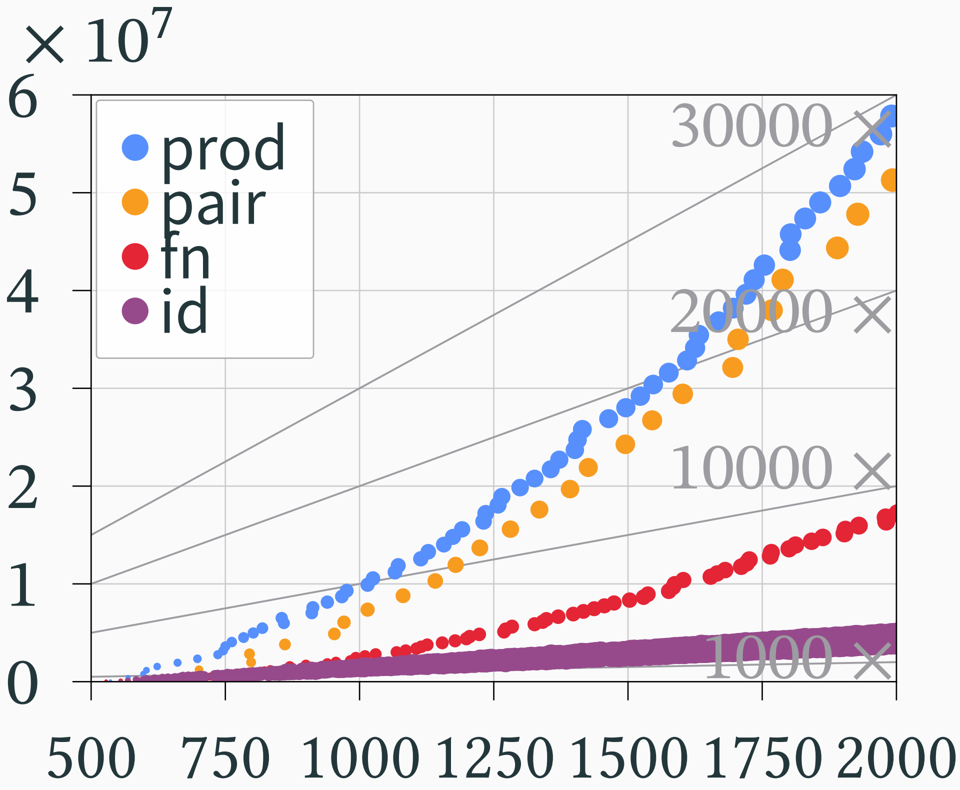
Prove that every type A in the groupoid model is internally a groupoid.

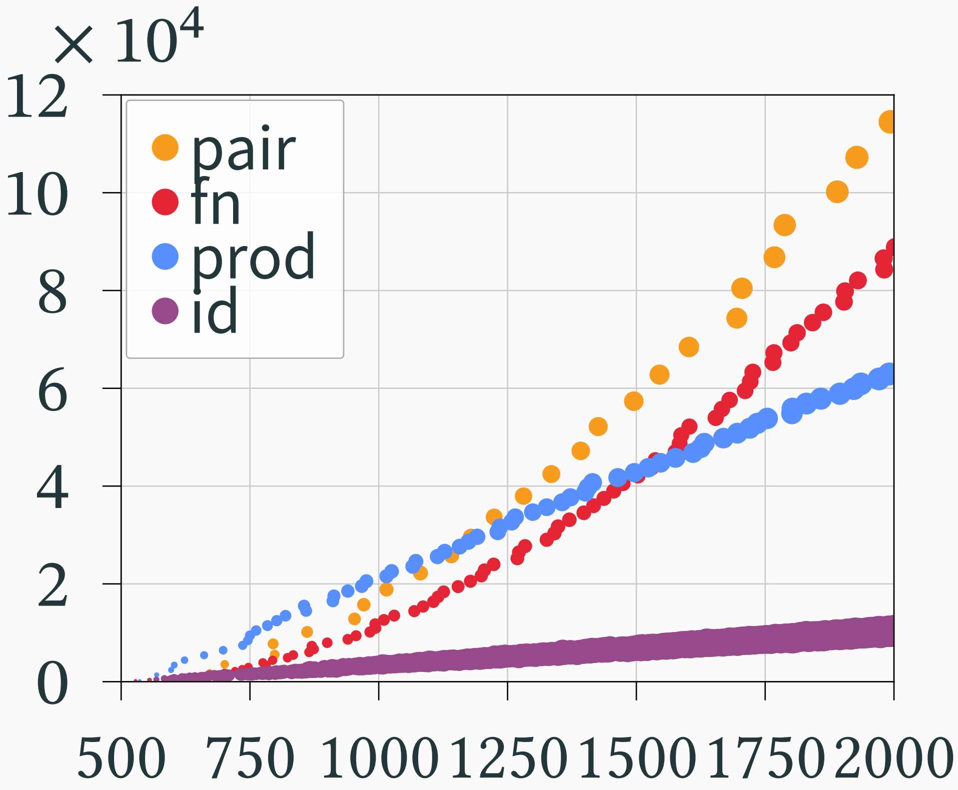
```
axiom isGrpd_all (A : Type) : isGrpd A
```

Prove that set-level univalence holds in the groupoid model.

```
axiom setUv (A B : Type) (hA : isSet A) (hB : isSet B) : (A ≃ B) ≃ (Path A B)
```

Typechecking performance (HoTTLean#142)





Defining $[\![-]\!]$ by recursion on raw syntax needs typing annotations, e.g. $\mathsf{app}(B, f, a)$ instead of $f\ a$.

$$\frac{\Gamma \vdash_{\mathrm{L\exists\forall N}} f \Rightarrow \Pi A.\, B \qquad \Gamma \vdash_{\mathrm{L\exists\forall N}} a \Leftarrow A}{\Gamma \vdash_{\mathrm{L\exists\forall N}} f\ a \Rightarrow B[a]} \quad \overset{\text{SynthLean}}{\rightsquigarrow} \quad \frac{\overline{\Gamma} \vdash_{\mathrm{SL}} \overline{a} \Rightarrow A' \qquad \overline{\Gamma} \vdash_{\mathrm{SL}} \overline{B} \qquad \overline{\Gamma} \vdash_{\mathrm{SL}} \overline{f} \Leftarrow \Pi A'.\, \overline{B}}{\overline{\Gamma} \vdash_{\mathrm{SL}} \mathsf{app}(\overline{B}, \overline{f}, \overline{a}) \Rightarrow \overline{B}[\overline{a}]}$$

🐌 Thanks to Pesara Amarasekera for pushing the system.

“So I ran it overnight, I think 5hrs or so (I went from fifty-thousand heartbeats to five-hundred-million) and it type checked...”

Universe polymorphism (HoTTLean#143)

Our definition of type theory does not include universe polymorphism.

```
@[reflect] def rfl₀ {α : Type 0} {a : α} : Path a a
@[reflect] def rfl₁ {α : Type 1} {a : α} : Path a a
```

Universe quantification in prenex form (what Lean does) may not require changes to this definition.

```
@[reflect]                  SynthLean      def X.reflection :
axiom.{u} X : Type u          ⇝              (u : ℕ) → ReflectedAx [] := ..     ?
```

Mutual induction (HoTTLean#155)

Mutual induction is everywhere in PL theory, yet support in Lean is tragic.

```
theorem le_univMax_all :
  (∀ {Γ}, WfCtx E Γ → ∀ {A i l}, Lookup Γ i A l → l ≤ univMax) ∧
  (∀ {Γ l A}, E ∣ Γ ⊢[l] A → l ≤ univMax) ∧
  (∀ {Γ l A B}, E ∣ Γ ⊢[l] A ≡ B → l ≤ univMax) ∧
  (∀ {Γ l A t}, E ∣ Γ ⊢[l] t : A → l ≤ univMax) ∧
  (∀ {Γ l A t u}, E ∣ Γ ⊢[l] t ≡ u : A → l ≤ univMax)
```

Early work by Jonathan Chan: github.com/ionathanch/MutualInduction

Notions of model

Which approach to semantics enables efficient model constructions?

- Natural models in $[\mathbf{Ctx}^{\mathrm{op}}, \mathbf{Set}]$?
- *Elementary models* in **Ctx**?
- *Π-clans* as an abstract setting for polynomial functors?

Want efficient constructions of ➰ groupoid model and eventually 🔺 simplicial set model.

See

- Awodey & Hua. *Path types in algebraic type theory.*
- Hua & Xu. *Polynomial functors in π-clans for the semantics of type theory.*

github.com/sinhp/HoTTLean

Bibliography

[1] Menno de Boer. 2020. A Proof and Formalization of the Initiality Conjecture of Dependent Type Theory. Licentiate Thesis. Department of Mathematics, Stockholm University.

[2] Thomas Streicher. 1991. Semantics of type theory: correctness, completeness, and independence results. Birkhäuser Boston Inc., USA.