Formalizing Hyperbolic Geometry

ai4math Workshop 2026

Théo Tyburn

August 1, 2026

AI is now good at autoformalization

- 2016: Maryna Viazovska proves Sphere Packing theorem in dimension 8
- 2024: Start of the formalization project.
- Nov 2025: Math Inc. starts contributing.
- Feb 2026: Gauss AI agent (Math Inc.) formalizes the proof autonomously.

Questions

Is formalization an interesting topic for a Master Thesis in mathematics?

- is it mathematical enough?
- is it not just computer science?
- can it not just be done with AI?

Can a math Master student do formalization successfully?

Goal of this talk

- Report on the process of formalizing standard mathematics from a student's point of view (process, difficulties, tools)
- Motivate students and advisors.

The Thesis

Project Goals:

- Formalize aspects of hyperbolic geometry in Lean
- Integration into Mathlib

Motivation:

- Learn Lean and Mathlib
- Solidify my mathematical understanding

Advisors:

- Michael Rothgang (Uni Bonn): Lean, Differential Geometry
- Jan Techter (TU Berlin): Projective Geometry

My Background

Geometry:

- Projective Geometry / Cayley-Klein Geometries / Erlangen Program
- Differential Geometry

Formalization:

- Lean FU Block-Seminar (2 weeks)

Programming:

- Familiarity with functional programming languages (Haskell, Lisp, ML)

Which definition of hyperbolic geometry ?

There are multiple equivalent approaches to defining hyperbolic geometry:

Synthetic (axiomatic)

- Euclid's axioms without the parallel postulate

Projective (Cayley–Klein)

- $\{[x] \in \mathbb{R}P^n \mid q(x,x) < 0\}$, with q a symmetric bilinear form of signature $(n,1)$

Riemannian (differential-geometric)

- simply connected Riemannian manifold with constant sectional curvature of -1.

Building on top of Mathlib

Previous work:

- Upper Half-Space model (as a metric space)

Available building blocks:

- Some Differential Geometry
  - Manifolds
  - Riemannian metric
  - Geodesics (soon)
- Less Projective Geometry (missing projections, quadrics).

$\rightarrow$ We chose the differential geometric approach.

Scope of the formalization

How much do we want to formalize ?

- Which model(s) of hyperbolic geometry ?
- Which dimension(s) ?
- What level of generality best serves future users?

$\rightarrow$ We decided to start with the Poincaré Disk model

Design choices

Which types, structures, or typeclasses are most appropriate?

Naive:

```
def PoincareDisk : Set ℂ := Metric.ball (0 : ℂ) 1
```

Better:

```
structure PoincareDisk where
  protected coe : ℂ
  coe_norm_lt_one : Complex.normSq coe < 1
```

→ Needs to align with Mathlib's existing design patterns.

Let's formalize !

Formalization challenges

Navigating Mathlib:

- where is this theorem?
- how is it called?
- in which level of abstraction is it formulated?
- does it even exist?

Type theory as a language to express mathematics

- function of multiple arguments need to be curried
- some conventions: forall, conjunctions
- coercion: "we can see this object as an element of . . . "

Type theory conventions

Lean makes an effort to be accessible to mathematicians.
But for usability it has to stick to some type theory conventions

```
example (n : ℝ) (h : 1 < n) : 0 < n := by linarith
```

is prefered over

```
example (n : ℝ) : 1 < n → 0 < n := fun h ↦ by linarith
```

and

```
example (n : ℕ) : 0 ≤ n := Nat.zero_le n
```

is prefered over

```
example : ∀ (n : ℕ), 0 ≤ n := Nat.zero_le
```

Tools to the rescue

Tools to navigate Mathlib:

- Mathlib documentation
- Lean Search
- Loogle
- Mathlib Phrasebook
- Zulip
- AI
- Your advisor

Tools to navigate Type Theory:

- Editor
- Error messages
- AI
- Your advisor

Tools to collaborate:

- git / codeberg
- `https://live.lean-lang.org/`

AI Tools

Explored AI Tools:

- Aristotle
- Gemini

Used for:

- navigating Mathlib: find the right theorem
- get used to type theory: "how to do this in Lean ?"
- scaffolding proofs
- correcting syntax

Can AI finish my thesis?

$\rightarrow$ Fable seems promissing.

Should not go against main goals of the project:

- learn how to formalize,
- meet Mathlib quality standards.

Community-Driven Formalization

Mathlib is:

- a currated library of mathematics,
- an open source software project,
- a mathematical community.

→ Mathlib's structure reflects the community's mental model of mathematics.

Progress and Next Steps

Current Status:

Initial formalization of the Poincaré Disk model

- Smooth Manifold
- Riemannian metric

Ongoing:

- Isometries
- Levi-Civita, Curvature
- Geodesics, Triangles
- Gauss-Bonnet ?

Conclusion

Takeaways:

- Formalizing mathematics in Lean is both challenging and rewarding
- Community-driven libraries like Mathlib make formalization a proper mathematical activity.
- Type Theory is difficult but quite intersting in its own right.

Invitation:

- Try to formalize things you know.
- Find people in the community to onboard you.

Thank you!

Links:

- Lean Theorem Prover: `https://lean-lang.org/`
- Mathlib: `https://leanprover-community.github.io/`
- Mathlib Phrasebook: `https://leanprover-community.github.io/mathlib-phrasebook/`