From Manifold to Machines: Formalizing Differential Geometry in Lean

Can machines give us trustworthy mathematical proofs?

Miraj Samarakkody

Texas Tech University

April 24, 2026

The Problem: Can we trust AI math?

AI is everywhere

From kindergarten homework to PhD research, students & researchers routinely ask AI math questions

The Problem: Can we trust AI math?

AI is everywhere

From kindergarten homework to PhD research, students & researchers routinely ask AI math questions

Hallucinated proofs

AI-generated proofs are often hallucinated, lack rigor, or contain subtle incorrect steps [1, 8].

The Problem: Can we trust AI math?

AI is everywhere

From kindergarten homework to PhD research, students & researchers routinely ask AI math questions

Hallucinated proofs

AI-generated proofs are often hallucinated, lack rigor, or contain subtle incorrect steps [1, 8].

Black-Box Reasoning

We have no insight (at least most of the time) into how the machine understands or arrives at a mathematical result.

The Problem: Can we trust AI math?

AI is everywhere

From kindergarten homework to PhD research, students & researchers routinely ask AI math questions

Hallucinated proofs

AI-generated proofs are often hallucinated, lack rigor, or contain subtle incorrect steps [1, 8].

Black-Box Reasoning

We have no insight (at least most of the time) into how the machine understands or arrives at a mathematical result.

What is the problem here?

How the generative AI works?

- Generative AI doesn't "understand" the proofs the way a mathematician does, it operates through pattern recognition over a vast amount of mathematical text [9].

How the generative AI works?

- Generative AI doesn't "understand" the proofs the way a mathematician does, it operates through pattern recognition over a vast amount of mathematical text [9].
- AlphaProof (DeepMind) takes a different approach: it couples a language model with a formal proof assistant, earning a silver-medal score at IMO 2024 [3].

What is Lean?

Lean is a proof assistant and functional programming language designed for writing formally verified mathematical proofs and verified software [2].

What is Lean?

Lean is a proof assistant and functional programming language designed for writing formally verified mathematical proofs and verified software [2].

- Python - General Code

What is Lean?

Lean is a proof assistant and functional programming language designed for writing formally verified mathematical proofs and verified software [2].

- Python - General Code
- Mathematica - Symbolic CAS

What is Lean?

Lean is a proof assistant and functional programming language designed for writing formally verified mathematical proofs and verified software [2].

- Python - General Code
- Mathematica - Symbolic CAS
- MATLAB - Numerical CAS

What is Lean?

Lean is a proof assistant and functional programming language designed for writing formally verified mathematical proofs and verified software [2].

- Python - General Code
- Mathematica - Symbolic CAS
- MATLAB - Numerical CAS
- Lean - Formal Proofs

Why formal proof verifications?

Advantages

- Absolute rigor

Why formal proof verifications?

Advantages

- Absolute rigor
- Machine-checkable

Why formal proof verifications?

Advantages

- Absolute rigor
- Machine-checkable
- Minimizes the human error

Why formal proof verifications?

Advantages

- Absolute rigor
- Machine-checkable
- Minimizes the human error
- Reveals AI reasoning

Why formal proof verifications?

Advantages

- Absolute rigor
- Machine-checkable
- Minimizes the human error
- Reveals AI reasoning

Trade-offs

- Verbosity

Why formal proof verifications?

Advantages

- Absolute rigor
- Machine-checkable
- Minimizes the human error
- Reveals AI reasoning

Trade-offs

- Verbosity
- Steep learning curve

Why formal proof verifications?

Advantages

- Absolute rigor
- Machine-checkable
- Minimizes the human error
- Reveals AI reasoning

Trade-offs

- Verbosity
- Steep learning curve
- Limited automation

Let's get to real examples

Differential Geometry in Lean 4/Mathlib4

Formalized in Mathlib4 [4, 6] (`Mathlib/Geometry/Manifold/`)

- Smooth manifolds, charted spaces, and smooth maps

Differential Geometry in Lean 4/Mathlib4

Formalized in Mathlib4 [4, 6] (`Mathlib/Geometry/Manifold/`)

- Smooth manifolds, charted spaces, and smooth maps
- Manifold derivatives (`mfderiv`), immersions, embeddings

Differential Geometry in Lean 4/Mathlib4

Formalized in Mathlib4 [4, 6] (`Mathlib/Geometry/Manifold/`)

- Smooth manifolds, charted spaces, and smooth maps
- Manifold derivatives (`mfderiv`), immersions, embeddings
- Tangent bundles, vector bundles, pullback bundles

Differential Geometry in Lean 4/Mathlib4

Formalized in Mathlib4 [4, 6] (`Mathlib/Geometry/Manifold/`)

- Smooth manifolds, charted spaces, and smooth maps
- Manifold derivatives (`mfderiv`), immersions, embeddings
- Tangent bundles, vector bundles, pullback bundles
- Vector fields and Lie bracket

Differential Geometry in Lean 4/Mathlib4

Formalized in Mathlib4 [4, 6] (`Mathlib/Geometry/Manifold/`)

- Smooth manifolds, charted spaces, and smooth maps
- Manifold derivatives (`mfderiv`), immersions, embeddings
- Tangent bundles, vector bundles, pullback bundles
- Vector fields and Lie bracket
- Riemannian metrics and **Levi-Civita connection** (2024–2025)

Differential Geometry in Lean 4/Mathlib4

Formalized in Mathlib4 [4, 6] (`Mathlib/Geometry/Manifold/`)

- Smooth manifolds, charted spaces, and smooth maps
- Manifold derivatives (`mfderiv`), immersions, embeddings
- Tangent bundles, vector bundles, pullback bundles
- Vector fields and Lie bracket
- Riemannian metrics and **Levi-Civita connection** (2024–2025)
- Differential forms and exterior derivative ($d^2 = 0$)

What is NOT yet formalized?

Open gaps in Mathlib4 (as of 2025)

- de Rham cohomology

What is NOT yet formalized?

Open gaps in Mathlib4 (as of 2025)

- de Rham cohomology
- Curvature tensors (Riemann, Ricci, scalar curvature)

What is NOT yet formalized?

Open gaps in Mathlib4 (as of 2025)

- de Rham cohomology
- Curvature tensors (Riemann, Ricci, scalar curvature)
- Geodesics and the exponential map

What is NOT yet formalized?

Open gaps in Mathlib4 (as of 2025)

- de Rham cohomology
- Curvature tensors (Riemann, Ricci, scalar curvature)
- Geodesics and the exponential map
- Characteristic classes

What is NOT yet formalized?

Open gaps in Mathlib4 (as of 2025)

- de Rham cohomology
- Curvature tensors (Riemann, Ricci, scalar curvature)
- Geodesics and the exponential map
- Characteristic classes
- General fiber/principal bundles

What is NOT yet formalized?

Open gaps in Mathlib4 (as of 2025)

- de Rham cohomology
- Curvature tensors (Riemann, Ricci, scalar curvature)
- Geodesics and the exponential map
- Characteristic classes
- General fiber/principal bundles

My Work: Isoperimetric Inequality in Lean 4

The Classical Statement [5]

Among all planar regions with a fixed perimeter L, the disk has the largest area A. Equivalently,

$$A \leq \frac{L^2}{4\pi},$$

with equality if and only if the region is a disk.

My Work: Isoperimetric Inequality in Lean 4

The Classical Statement [5]

Among all planar regions with a fixed perimeter L, the disk has the largest area A. Equivalently,

$$A \leq \frac{L^2}{4\pi},$$

with equality if and only if the region is a disk.

This work [7]

A fully machine-checked formalization in Lean 4 / Mathlib4 following Hurwitz's classical proof via Fourier series — **zero** `sorry` placeholders.

Proof Strategy: Hurwitz's Method

1. **Arc-length parameterization** — represent the boundary as a `SimpleClosedC1Curve`, reparameterized by arc length $\theta \in [-\pi, \pi]$.

Proof Strategy: Hurwitz's Method

1. **Arc-length parameterization** — represent the boundary as a `SimpleClosedC1Curve`, reparameterized by arc length $\theta \in [-\pi, \pi]$.
2. **Fourier expansion** — decompose coordinates into

$$x(\theta) = a_0 + \sum_{n=1}^{\infty} \big(a_n \cos n\theta + b_n \sin n\theta\big),$$

and similarly for $y(\theta)$.

Proof Strategy: Hurwitz's Method

1. **Arc-length parameterization** — represent the boundary as a `SimpleClosedC1Curve`, reparameterized by arc length $\theta \in [-\pi, \pi]$.
2. **Fourier expansion** — decompose coordinates into

$$x(\theta) = a_0 + \sum_{n=1}^{\infty} \big(a_n \cos n\theta + b_n \sin n\theta\big),$$

   and similarly for $y(\theta)$.
3. **Parseval's identity** — relate the arc-length constraint to the Fourier coefficients via Parseval.

Proof Strategy: Hurwitz's Method

1. **Arc-length parameterization** — represent the boundary as a `SimpleClosedC1Curve`, reparameterized by arc length $\theta \in [-\pi, \pi]$.
2. **Fourier expansion** — decompose coordinates into

$$x(\theta) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos n\theta + b_n \sin n\theta \right),$$

   and similarly for $y(\theta)$.
3. **Parseval's identity** — relate the arc-length constraint to the Fourier coefficients via Parseval.
4. **Wirtinger's inequality** — bound the area integral by the derivative norm, yielding $A \leq L^2/4\pi$.

Proof Strategy: Hurwitz's Method

1. **Arc-length parameterization** — represent the boundary as a `SimpleClosedC1Curve`, reparameterized by arc length $\theta \in [-\pi, \pi]$.
2. **Fourier expansion** — decompose coordinates into

$$x(\theta) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos n\theta + b_n \sin n\theta \right),$$

   and similarly for $y(\theta)$.
3. **Parseval's identity** — relate the arc-length constraint to the Fourier coefficients via Parseval.
4. **Wirtinger's inequality** — bound the area integral by the derivative norm, yielding $A \leq L^2/4\pi$.
5. **Equality** — the bound is tight iff all higher modes vanish, i.e. the curve is a circle.

Repository Structure

`IsoperimetricInequality/Basic.lean`

- Fourier series orthogonality lemmas

Repository Structure

`IsoperimetricInequality/Basic.lean`

- Fourier series orthogonality lemmas
- Uniform convergence via Weierstrass M-test

Repository Structure

`IsoperimetricInequality/Basic.lean`

- Fourier series orthogonality lemmas
- Uniform convergence via Weierstrass M-test
- **Parseval's identity**

Repository Structure

`IsoperimetricInequality/Basic.lean`

- Fourier series orthogonality lemmas
- Uniform convergence via Weierstrass M-test
- **Parseval's identity**
- **Wirtinger's inequality**

Repository Structure

`IsoperimetricInequality/Basic.lean`

- Fourier series orthogonality lemmas
- Uniform convergence via Weierstrass M-test
- **Parseval's identity**
- **Wirtinger's inequality**

`IsoperimetricInequality/Adolf_Hurwitz_proof.lean`

- `SimpleClosedC1Curve`, `area` (shoelace formula)

Repository Structure

`IsoperimetricInequality/Basic.lean`

- Fourier series orthogonality lemmas
- Uniform convergence via Weierstrass M-test
- **Parseval's identity**
- **Wirtinger's inequality**

`IsoperimetricInequality/Adolf_Hurwitz_proof.lean`

- `SimpleClosedC1Curve`, `area` (shoelace formula)
- Arc-length parameterization and periodicity lemmas

Repository Structure

`IsoperimetricInequality/Basic.lean`

- Fourier series orthogonality lemmas
- Uniform convergence via Weierstrass M-test
- **Parseval's identity**
- **Wirtinger's inequality**

`IsoperimetricInequality/Adolf_Hurwitz_proof.lean`

- `SimpleClosedC1Curve`, `area` (shoelace formula)
- Arc-length parameterization and periodicity lemmas
- Integration by parts, AM-GM bound

Repository Structure

`IsoperimetricInequality/Basic.lean`

- Fourier series orthogonality lemmas
- Uniform convergence via Weierstrass M-test
- **Parseval's identity**
- **Wirtinger's inequality**

`IsoperimetricInequality/Adolf_Hurwitz_proof.lean`

- `SimpleClosedC1Curve`, `area` (shoelace formula)
- Arc-length parameterization and periodicity lemmas
- Integration by parts, AM-GM bound
- `isoperimetric_inequality` — main theorem

The Main Theorem in Lean

```
theorem isoperimetric_inequality
  (γ :  SimpleClosedC1Curve 2)
  (hdc :  Continuous (deriv γ.curve))
  (h_arc :  IsArcLengthParametrized γ)
  ...
  :  area γ 0 γ.length ≤
    γ.length^ 2 / (4 * Real.pi) := by
```

... – 26 lemmas, no sorry

The Main Theorem in Lean

```
theorem isoperimetric_inequality
  (γ :  SimpleClosedC1Curve 2)
  (hdc :  Continuous (deriv γ.curve))
  (h_arc :  IsArcLengthParametrized γ)
  ...
  :  area γ 0 γ.length ≤
    γ.length^ 2 / (4 * Real.pi) := by
  ... – 26 lemmas, no sorry
```

Verified by

Lean 4 kernel + Mathlib4 — every logical step is machine-checked.

Acknowledgement

This work is supported by the **NSF HBCU-UP Implementation Project**, Award No. 2510537.

Thank you!

References I



S. Banerjee et al.

LLMs will always hallucinate, and we need to live with this.

arXiv preprint arXiv:2409.05746, 2024.



L. de Moura and S. Ullrich.

The Lean 4 theorem prover and programming language.

In *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer, 2021.



T. Hubert, R. Mehta, L. Sartran, et al.

Olympiad-level formal mathematical reasoning with reinforcement learning.

Nature, 651:607–613, 2025.



T. mathlib Community.

The Lean mathematical library.

In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, pages 367–381, 2020.

References II



R. Osserman.

The isoperimetric inequality.

Bulletin of the American Mathematical Society, 84(6):1182–1238, 1978.



M. Rothgang.

Differential geometry in Mathlib: Present and future.

Seminar slides, Université Paris-Saclay (Orsay), 2025.



M. Samarakkody.

Formalization of the isoperimetric inequality in Lean 4.

arXiv preprint arXiv:2603.14663, 2026.



Z. Xu et al.

Hallucination is inevitable: An innate limitation of large language models.

arXiv preprint arXiv:2401.11817, 2024.

v2, revised 13 Feb 2025.



K. Yang, G. Poesia, J. He, W. Li, K. Lauter, S. Chaudhuri, and D. Song.

Formal mathematical reasoning: A new frontier in AI.

arXiv preprint arXiv:2412.16075, 2024.