

LeanGeo: Formalizing Competitional Geometry Problems in Lean

Chendong Song^{*1} Zihan Wang^{*13} Frederick Pu²⁴ Haiming Wang¹ Xiaohan Lin¹
Junqi Liu¹⁵ Jia Li² Zhengying Liu^{†1}

¹Moonshot AI ²Numina ³Peking University ⁴University of Toronto

⁵University of Chinese Academy of Sciences

songcd1212@gmail.com, zihanwang@stu.pku.edu.cn,

liuzhengying@moonshot.cn

Abstract

Geometry problems constitute a crucial testbed for AI reasoning. Most existing geometry solving systems rely on domain-specific formal languages that cannot integrate with other mathematical fields, and their dependence on graphical intuition makes rigorous verification particularly challenging. To address these limitations, we introduce **LeanGeo**, a unified formal framework for expressing and proving competition-level geometry problems within the Lean 4 theorem prover. LeanGeo provides a comprehensive library of 260 high-level geometric theorems grounded in Lean’s foundational logic, enabling rigorous proof verification and seamless integration with Mathlib. We further present **LeanGeo-Bench**, a benchmark of 122 formally verified geometry problems ranging from foundational exercises to International Mathematical Olympiad (IMO) challenges. Our evaluation of state-of-the-art Large Language Models on this benchmark reveals both capabilities and fundamental limitations in automated geometric reasoning. We open-source the theorem library and benchmark at <https://github.com/project-numina/LeanGeo>.

1 Introduction

Large Language Models (LLMs) have achieved significant progress in mathematical reasoning, particularly in automated theorem proving (Bibel, 2013). Lean 4 (Moura & Ullrich, 2021), supported by the extensive Mathlib library (mathlib community, 2020), provides a robust foundation for algebra and number theory and has become widely adopted for formal verification.

However, Euclidean geometry—an essential component of mathematical reasoning and a frequent focus of competitions—remains underexplored in Lean 4, Mathlib, and automated theorem provers. This stems from the inherent difficulty of geometric problems, which demand graphical intuition that resists straightforward formalization.

Advanced geometry systems such as AlphaGeometry (Trinh et al., 2024), TongGeometry (Zhang et al., 2024a), and SeedGeometry (Chen et al., 2025) achieve impressive results on IMO-level problems but operate within geometry-specific formal systems isolated from Lean. This isolation prevents integration with other mathematical domains, making it impossible to express geometric inequalities or positional relations. Moreover, their reliance on graphical verification and unordered formal systems can compromise logical soundness (see Table 3 and Appendix E).

Within Lean 4 itself, formalized geometry remains scarce. Mathlib’s geometry support is largely algebraic with little provision for synthetic reasoning. Myers (Zhang et al., 2022)

^{*}Equal contribution.

[†]Corresponding author.

formalized a single IMO geometry problem—an impressive but isolated result that does not develop a reusable geometric library.

Equally important is establishing benchmarks for evaluating geometric reasoning. Existing benchmarks such as GeoEval (Zhang et al., 2024b), GeoQA (Chen et al., 2021), Geometry3K (Hiyouga, 2025), and FormalMath (Yu et al., 2025) primarily emphasize numerical computation rather than genuine geometric proof. LLMs also exhibit unsatisfactory performance on Lean 4 geometry benchmarks such as MATP-Bench (He et al., 2025), partly due to the absence of a geometry theorem library.

To address these gaps, we introduce **LeanGeo**, a framework for formalizing and solving geometric problems in Lean 4. Building upon LeanEuclid (Murphy et al., 2024), LeanGeo provides a comprehensive theorem library specifically curated for competition-level challenges, with seamless Mathlib integration (see Table 1).

Table 1: Comparison of problem representation in AlphaGeometry and LeanGeo

Natural Language	In a triangle ABC, side $AB = AC$, prove that $\angle ACB = \angle ABC$. Solution. Choose D as the midpoint of side BC. Then $\triangle ABD$ and $\triangle ACD$ are congruent. Therefore, $\angle ACB = \angle ACD = \angle ABD = \angle ABC$
AlphaGeometry	$a\ b = \text{segment } a\ b; c = \text{on_circle } c\ a\ b\ ?\ \text{eqangle } b\ a\ b\ c\ b\ c\ c\ a$ Solution. * From theorem premises: $A\ B\ C : \text{Points}$ $\text{cong } A\ C\ A\ B\ [00]$ * Auxiliary Constructions: $: \text{Points}$ * Proof steps: 001. $\text{cong } A\ C\ A\ B\ [00] \Rightarrow \text{eqangle } A\ C\ B\ C\ B\ C\ A\ B$
LeanGeo	$\text{theorem isoTriangle_imp_eq_angles} : \forall (A\ B\ C : \text{Point}),$ $\text{IsoTriangle } A\ B\ C \rightarrow \angle A:B:C = \angle A:C:B := \text{by}$ euclid_intros $\text{euclid_apply exists_midpoint } B\ C\ \text{as } D$ $\text{euclid_apply line_from_points } B\ C\ \text{as } BC$ $\text{euclid_apply coll_angles_eq}$ $\text{euclid_apply congruentTriangles_SSS } D\ B\ A\ D\ C\ A$ $\text{euclid_apply coll_angles_eq}$ euclid_finish

Based on this library, we propose **LeanGeo-Bench**, the first formalized geometry benchmark in Lean 4, comprising 122 problems including all IMO geometry problems since 2000. The primary contributions of this work are:

- We present the first Lean 4 framework capable of expressing and reasoning about competition-level geometry in a human-readable manner, featuring an extensive library of high-level definitions and tactics based on theorems used by IMO competitors. Its Lean integration facilitates cross-domain formalization involving geometry and other fields such as combinatorics.
- We introduce a comprehensive geometry benchmark formalized in Lean 4, capable of representing most IMO geometry problems. This benchmark provides a standardized testbed for evaluating formal mathematics systems, accompanied by baseline results from several state-of-the-art LLMs.

2 Related Work

2.1 Automated Theorem Proving

Interactive theorem provers span a spectrum of foundations: HOL4 (Slind & Norrish, 2008) and Isabelle/HOL (Paulson, 1994) employ simply-typed higher-order logic, while Coq (Barras et al., 1999) and Lean (De Moura et al., 2015) use dependent type theory.

Search-based provers have been developed to enhance automated reasoning. LEGO-Prover (Wang et al., 2023a) constructs a reusable skill library for modular proof synthesis. DT-Solver (Wang et al., 2023b) introduces dynamic-tree Monte Carlo search, and BFS-Prover (Xin et al., 2025) achieves state-of-the-art performance via best-first search.

More recently, whole-proof generation approaches have emerged, where a language model produces entire proofs in a single pass, including DeepSeek-Prover (Ren et al., 2025), Goedel-Prover (Lin et al., 2025), and Kimina-Prover (Wang et al., 2025). Agentic methods such as Delta Prover (Zhou et al., 2025) integrate reflective decomposition with iterative repair. Seed-Prover (Chen et al., 2025) combines multi-stage reinforcement learning with agent-based strategies, solving 4 of 6 IMO 2025 problems.

2.2 LeanEuclid

LeanEuclid (Murphy et al., 2024) pioneered the formalization of plane geometry within Lean by integrating SMT (Barrett & Tinelli, 2018) solving with SystemE (Avigad et al., 2009). It provides an autoformalization benchmark covering the first chapter of Euclid’s *Elements* and 125 simple problems from UniGeo.

LeanGeo substantially extends LeanEuclid. Since LeanEuclid formalizes only the 49 propositions in *Elements* I, its expressive power is insufficient for standard competition geometry. LeanGeo shares the same axiomatic foundation but provides a significantly richer collection of theorems, definitions, and geometric structures, along with an improved SMT backend (Table 2).

Table 2: Comparison between LeanEuclid and LeanGeo

	LEANEUCLID	LEANGEO
Axiom Number	107	116
Theorem Number	106	260
Geometry Structure Number	12	50
Average Proof Length	20.27	16.20
Average number of quote lemma	3.80	3.43
SMT Method	Hard-coded rules	LeanSMT
Level	Euclid’s Element	Competitional Geometry

2.3 Geometry Problem Solving

Classical algebraic methods—Wu’s characteristic set (Wu, 1986) and Gröbner bases (Bose, 1995)—reduce geometry to polynomial ideal membership with broad coverage of textbook theorems.

AlphaGeometry (Trinh et al., 2024) integrates a neural language model with a symbolic deduction engine, solving 25 of 30 IMO-level problems. Built on Chou et al. (2000), its point-centered formal system enables fast symbolic deduction but has notable limitations: it lacks support for geometric inequalities, trigonometric reasoning, and positional relations, and its unsound formal system precludes rigorous proof verification. In contrast, LeanGeo prioritizes completeness, rigor, and structural expressivity, though at the cost of more elaborate reasoning (Table 3; Appendix E).

Table 3: Comparison between AlphaGeometry and LeanGeo

Category	Feature	AG	LeanGeo
Expressivity	Inequality & Trigonometric Functions	×	✓
	Metric Relation (Perp., Parallel, Equal)	✓	✓
	Positional Relation (Inside, Between, etc.)	×	✓
	Existential Proposition	×	✓
	Linear Computation	✓	✓
	Non-linear Computation	×	✓
Verifiability	Proof Verifiability	×	✓
Axiom System	Soundness	×	✓
	Extensibility	×	✓

2.4 Geometry and Lean Benchmarks

Several Lean-based benchmarks have been developed for automated theorem proving. MiniF2F (Zheng et al., 2021) evaluates systems on high-school algebra and number theory. Geometry benchmarks such as GeoEval (Zhang et al., 2024b), GeoQA (Chen et al., 2021), Geometry3K (Hiyouga, 2025), and FormalMath (Yu et al., 2025) assess computational reasoning but underrepresent classical Euclidean proof. AlphaGeometry’s IMO-30 and JGEX-231 benchmarks emphasize problem-solving without supporting verifiable formal proofs. MATP-Bench (He et al., 2025) aggregates Lean 4 geometry problems, yet the absence of a comprehensive theorem library hinders LLM performance. A detailed comparison is provided in Table 4.

Table 4: Comparison of Geometry and Lean Benchmarks

Benchmark	Size	Verifiable	Geo. Formal Proving %	Theorem Library	Level
miniF2F (Zheng et al., 2021)	488	✓	0%	Mathlib	Middle School
Geometry3K-test (Hiyouga, 2025)	601	✓	0%	×	Middle School
LeanEuclid (Murphy et al., 2024)	173	✓	0%	SystemE	Elementary
AG-IMO-30 (Trinh et al., 2024)	30	×	100%	DD rules	Olympiad
MATP-Bench (He et al., 2025)	1056	✓	~20%	×	Mixed-level
LeanGeo-Bench	122	✓	100%	LeanGeo	Mixed-level

3 LeanGeo

LeanGeo is a manually formalized system of plane geometry theorems and proofs in the Lean 4 proof assistant. It builds upon the axiomatic framework of SystemE (Avigad et al., 2009) and inherits most foundational geometric objects and relations from LeanEuclid (Murphy et al., 2024), with minor modifications (Appendix ??). At its core, LeanGeo leverages LeanSMT (Mohamed et al., 2025) to effectively discharge low-level proof obligations.

3.1 Theorem Library

To enhance expressive power and align with standard geometric terminology, we introduced 52 new definitions for geometric structures—such as Midpoint, Circumcenter, and RadicalAxis—using abbrev (Listing 1). These definitions make problem statements more concise without increasing SMT processing overhead.

```
abbrev Cyclic (A B C D: Point) : Prop :=
  ∃ (O: Circle), A.onCircle O ∧ B.onCircle O ∧ C.onCircle O ∧ D.onCircle O
```

Listing 1: Example of abbreviation

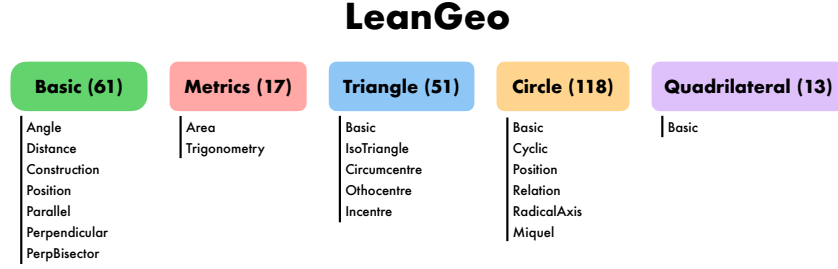


Figure 1: Structure of the LeanGeo Theorem Library

With these definitions, LeanGeo comprises 260 geometric theorems (Figure 1), all manually written, formally proved, and machine-verified by Lean 4 and LeanSMT. The library systematically covers topics from foundational geometry to IMO-level results such as Menelaus’s theorem and Miquel’s theorem, including properties of triangles (congruence, similarity), circles (inscribed angles, power of a point, radical axis), quadrilaterals, and key geometric points (circumcenter, orthocenter).

A key design principle is that most proofs reference previously established theorems via the `euclid_apply` tactic, mirroring the human process of building geometric theory from axioms to complex structures (Listing 5). As the library grows, these reusable lemmas substantially shorten higher-level proofs. Experiments (Appendix A.2) confirm that this modular structure is essential: inlining lemmas increases compilation time severely, whereas separate lemmas enable shared arguments to be compiled once and reused.

LeanGeo integrates seamlessly with Mathlib, enabling the use of trigonometric identities and advanced inequalities for problems beyond the reach of purely axiomatic systems. As demonstrated in Appendix D.2, Mathlib trigonometric theorems are applied to prove IMO_2001_P1, a geometric inequality problem difficult to express in most geometric formal systems.

A distinctive challenge in formal geometry is describing positional relationships without visual aids. As illustrated in Figure 2, natural-language proofs and systems like AlphaGeometry typically consider only a single configuration. Lean’s stringent rigor requires LeanGeo proofs to account for all possible cases explicitly. While this increases proof complexity, it ensures a higher level of correctness. To mitigate overly cumbersome case analyses, we make extensive use of SMT solvers to simplify classification and discharge trivial goals.

3.2 LeanSMT Integration

LeanGeo invokes the CVC5 (Barbosa et al., 2022) SMT solver to discharge goals considered trivial in natural-language proofs. Unlike LeanEuclid, which embeds SystemE axioms as hard-coded SMT commands, LeanGeo employs the `esmt` tactic that directly passes all local hypotheses from the current tactic state—together with SystemE inference axioms and the negated goal—to CVC5 for an unsatisfiability check.

For performance, parsed axiom expressions are cached and a global metavariable dependency graph is maintained, dynamically updated upon encountering definitions or axioms annotated with `@[euclid]` (Listing 2; Appendix B).

```
@[euclid]
axiom zero_segment_if :
  ∀ (a b : Point), |(a - b)| = 0 → a = b
```

Listing 2: tactic usage]Example of `@[euclid]` tactic usage

The `@[euclid]` annotation makes LeanGeo more extensible than LeanEuclid: new definitions such as `sin` and `cos` can be incorporated without manual translator modifications. LeanGeo also inherits LeanEuclid’s tactic interface (`euclid_intros`, `euclid_apply`, `euclid_finish`), with automatic LeanSMT invocation (see Appendix D.1).

Natural Language:

The Circle O_1 and O_2 intersect at K and B . A line through B intersects the circle O_1 at A and circle O_2 at C . Prove that triangle KO_1O_2 and KAC are similar.

Formal Statement in LeanGeo:

theorem intersectCircles_similarTriangles_of_one_secant : $\forall (O_1 O_2 A B C K : \text{Point}) (\Omega_1 \Omega_2 : \text{Circle}), \Omega_1 \neq \Omega_2 \wedge O_1.\text{isCentre } \Omega_1 \wedge O_2.\text{isCentre } \Omega_2 \wedge \text{CirclesIntersectAtTwoPoints } \Omega_1 \Omega_2 B K \wedge A.\text{onCircle } \Omega_1 \wedge C.\text{onCircle } \Omega_2 \wedge \text{Coll } A B C \wedge A \neq B \wedge B \neq C \wedge A \neq K \wedge C \neq K \rightarrow \text{SimilarTriangles } O_1 O_2 K A C K := \text{by}$

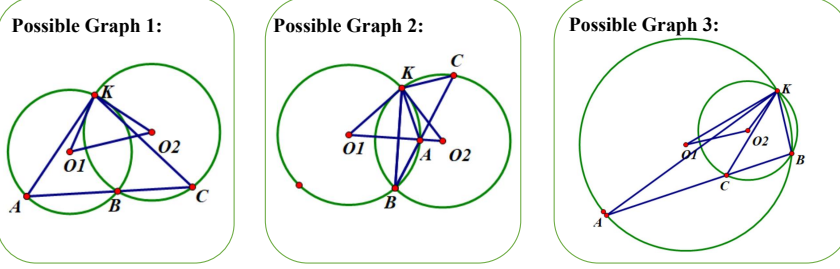


Figure 2: Different diagrams sharing the same formal statement

We analyze scalability across four controlled experiments varying the number of geometry elements, assumptions, proof length, and `euclid` tactic invocations (Appendix A.1). Both heartbeats and compilation time exhibit near-linear growth with problem size, with no pathological slowdowns observed.

4 LeanGeo-Bench

4.1 Benchmark Construction

LeanGeo-Bench is a formal benchmark for contest-level plane geometry in Lean 4. As shown in Table 5, it comprises 122 problems from diverse sources. Difficulty ranges from

Table 5: Composition of LeanGeo-Bench

Section	N	Source	Method
UniGeo (UG)	10	LeanEuclid	Manually written
Library (LB)	10	LeanGeo Library	Manually written
Synthetic (SP)	20	LeanGeo Library	Generated by Gemini
High School Competition (HSC)	20	NuminaMath	Autoformalized + expert review
Olympiad Problems (OP)	19	Evan Chen’s textbook	Autoformalized + expert review
IMO	43	AoPS	Autoformalized + expert review
Total	122		

foundational to competition-level. The benchmark includes 20 introductory problems from UniGeo (Chen et al., 2022) and the LeanGeo library, 20 synthetically generated problems produced by Gemini-2.5 Pro, and 82 advanced problems from high-school curricula, NuminaMath (Li et al., 2024), Evan Chen’s textbook Chen (2021), and all IMO geometry problems since 2000 from AoPS (Art of Problem Solving). Advanced problems were developed via human-in-the-loop methodology: each problem was first autoformalized by an LLM through prompt engineering, then rigorously reviewed and corrected by two human experts.

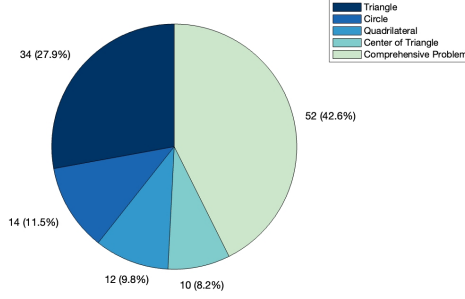


Figure 3: Category Distribution of LeanGeo-Bench

The benchmark covers triangles, circles, quadrilaterals, and triangle centers (incenter, circumcenter), as shown in Figure 3. Problem types include traditional proofs, angle and length derivations, three geometric inequality problems, and two moving-point problems.

We provide 43 formally verified solutions, including two IMO problems, all machine-checked in Lean. For problems without formal proofs, correctness is validated via negation-based methods combined with independent expert review.

4.2 Evaluation Method

We design a comprehensive prompt comprising a custom declarative DSL for LeanGeo, error-and-correction examples, construction rules, the full theorem library, and few-shot examples. Evaluation follows the `online_one_stage` Fine-Eval method from CombiBench (Liu et al., 2025): we first verify consistency with the original formal statement, then feed the result into a Lean server with the pre-built theorem library for formal verification.

4.3 Baseline Results

We evaluated Gemini 2.5 Pro (DeepMind, 2025), o4-mini (OpenAI, 2025), Grok 4 (xAI, 2025), Kimi K2 (MoonshotAI, 2025), Claude 4 (Anthropic, 2025), and Qwen3-235B-A22B (Yang et al., 2025) across different sample budgets. Results are shown in Table 6.

Table 6: Evaluation on LeanGeo-Bench

Model	Overall Success Rate (%)			Success Count (pass@4)					
	pass@1	pass@2	pass@4	UG	LB	SP	HSC	OP	IMO
Gemini 2.5 Pro	17.21	22.95	27.05	10	4	13	6	0	0
o4-mini	19.67	21.31	22.13	7	9	8	3	0	0
Grok 4	16.39	21.31	24.59	10	6	11	3	0	0
Kimi K2	9.02	9.02	9.84	1	9	2	0	0	0
Claude 4	4.92	9.02	10.66	1	5	7	0	0	0
Qwen3-235B-A22B	3.28	4.10	5.74	0	6	1	0	0	0
Total				10	10	20	20	19	43

The results reveal substantial performance differences across models. o4-mini achieves the highest pass@1 (19.67%), while Gemini 2.5 Pro leads at pass@4 (27.05%). Category-level analysis reveals complementary strengths: Gemini excels on synthetic and high-school competition problems, suggesting stronger adaptability to unseen reasoning patterns, whereas o4-mini demonstrates greater proficiency on library problems, indicating better utilization of the theorem library in the prompt.

Notably, all models plateau below 30% overall, and *none* solve any of the 62 Olympiad-level problems, indicating fundamental limitations in handling complex geometric proofs requiring sophisticated logical reasoning and formal verification.

5 Discussion and Future Work

While LeanGeo demonstrates the viability of a declarative, human-readable approach for competition-level geometry, several challenges remain.

Automation Capabilities. General-purpose SMT solvers lack geometry-specific heuristics, causing solving speed to degrade as problem complexity increases. Embedding domain-specific proof automation—such as the Area Method (Janicic et al., 2012) or algebraic geometry techniques—into the tactic framework could significantly extend LeanGeo’s reach.

Knowledge Integration for LLMs. Currently, the entire theorem library is included in the prompt to ensure correct theorem citation. However, excessively long prompts may degrade model performance. Developing methods to selectively retrieve relevant theorems or to internalize geometric knowledge within the model remains an important direction.

Foundational Soundness. Strengthening the axiomatic foundation and extending the library to cover additional geometric domains (e.g., projective geometry, non-Euclidean geometry) would broaden the system’s applicability.

6 Conclusion

We present LeanGeo, the first Lean-based framework capable of formalizing and solving competition-level geometry problems, together with LeanGeo-Bench, a 122-problem benchmark spanning foundational theorems to IMO challenges. LeanGeo’s declarative, human-readable proofs, deep Mathlib integration, and extensible library enable rigorous cross-domain reasoning beyond the reach of existing geometry systems.

Our evaluations reveal that while current LLMs achieve partial success, they fall far short on the hardest problems—none solving any Olympiad-level challenge—underscoring the need for stronger geometric reasoning and proof search capabilities. By providing a rich formal library and a challenging benchmark, LeanGeo establishes a scalable testbed for advancing automated geometry theorem proving.

7 Reproducibility Statement

To reproduce experiments or run the benchmark evaluation, please clone the repository at <https://github.com/project-numina/LeanGeo> and follow the instructions in `README.md`. Our evaluation toolkit provides an end-to-end benchmark harness that reproduces the results reported in this paper.

References

- Anthropic. Introducing claude 4. <https://www.anthropic.com/news/claude-4>, July 2025. Accessed on 2025-07-25.
- Art of Problem Solving. Art of problem solving. Website. URL <https://artofproblemsolving.com>. Accessed: 2025-08-12.
- Jeremy Avigad, Edward Dean, and John Mumma. A formal system for euclid’s elements. *The Review of Symbolic Logic*, 2(4):700–768, 2009.
- Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, et al. cvc5: A versatile and industrial-strength smt solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 415–442. Springer, 2022.

- Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye, Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, et al. The coq proof assistant reference manual. *INRIA, version*, 6(11):17–21, 1999.
- Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of model checking*, pp. 305–343. Springer, 2018.
- Wolfgang Bibel. *Automated theorem proving*. Springer Science & Business Media, 2013.
- NK Bose. Gröbner bases: An algorithmic method in polynomial ideal theory. In *Multidimensional systems theory and applications*, pp. 89–127. Springer, 1995.
- Evan Chen. *Euclidean geometry in mathematical olympiads*, volume 27. American Mathematical Soc., 2021.
- Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric P Xing, and Liang Lin. Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning. *arXiv preprint arXiv:2105.14517*, 2021.
- Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. Unigeo: Unifying geometry logical reasoning via reformulating mathematical expression. *arXiv preprint arXiv:2212.02746*, 2022.
- Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, Cheng Ren, Jiawei Shen, Wenlei Shi, Tong Sun, He Sun, Jiahui Wang, Siran Wang, Zhihong Wang, Chenrui Wei, Shufa Wei, Yonghui Wu, Yuchen Wu, Yihang Xia, Huajian Xin, Fan Yang, Huaiyuan Ying, Hongyi Yuan, Zheng Yuan, Tianyang Zhan, Chi Zhang, Yue Zhang, Ge Zhang, Tianyun Zhao, Jianqiu Zhao, Yichi Zhou, and Thomas Hanwen Zhu. Seed-prover: Deep and broad reasoning for automated theorem proving, 2025. URL <https://arxiv.org/abs/2507.23726>.
- Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning*, 25(3):219–246, 2000.
- Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *International Conference on Automated Deduction*, pp. 378–388. Springer, 2015.
- DeepMind. Gemini 2.5 pro. <https://deepmind.google/models/gemini/pro/>, July 2025. Accessed on 2025-07-25.
- Zhitao He, Zongwei Lyu, Dazhong Chen, Dadi Guo, and Yi R Fung. Matp-bench: Can mllm be a good automated theorem prover for multimodal problems? *arXiv preprint arXiv:2506.06034*, 2025.
- Hiyouga. Geometry3k dataset, 2025. URL <https://huggingface.co/datasets/hiyouga/geometry3k>.
- Predrag Janicic, Julien Narboux, and Pedro Quaresma. The Area Method : a Recapitulation. *Journal of Automated Reasoning*, 48(4):489–532, 2012. doi: 10.1007/s10817-010-9209-7. URL <https://hal.science/hal-00426563>.
- Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.
- Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, et al. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint arXiv:2508.03613*, 2025.

- Junqi Liu, Xiaohan Lin, Jonas Bayer, Yael Dillies, Weijie Jiang, Xiaodan Liang, Roman Soletskyi, Haiming Wang, Yunzhou Xie, Beibei Xiong, et al. Combibench: Benchmarking llm capability for combinatorial mathematics. *arXiv preprint arXiv:2505.03171*, 2025.
- The mathlib community. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020*, pp. 367–381. ACM, 2020. doi: 10.1145/3372885.3373824.
- Abdalkhman Mohamed, Tomaz Mascarenhas, Harun Khan, Haniel Barbosa, Andrew Reynolds, Yicheng Qian, Cesare Tinelli, and Clark Barrett. Lean-smt: An smt tactic for discharging proof goals in lean. In *International Conference on Computer Aided Verification*, pp. 197–212. Springer, 2025.
- MoonshotAI. Kimi k2: Open agentic intelligence. <https://moonshotai.github.io/Kimi-K2/>, July 2025. Accessed on 2025-07-25.
- Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *International Conference on Automated Deduction*, pp. 625–635. Springer, 2021.
- Logan Murphy, Kaiyu Yang, Jialiang Sun, Zhaoyu Li, Anima Anandkumar, and Xujie Si. Autoformalizing euclidean geometry. *arXiv preprint arXiv:2405.17216*, 2024.
- OpenAI. Announcing openai o3 and o4-mini. <https://openai.com/index/introducing-o3-and-o4-mini/>, July 2025. Accessed on 2025-07-25.
- Lawrence C Paulson. *Isabelle: A generic theorem prover*. Springer, 1994.
- ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanxia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.
- Konrad Slind and Michael Norrish. A brief overview of hol4. In *International Conference on Theorem Proving in Higher Order Logics*, pp. 28–32. Springer, 2008.
- Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.
- Haiming Wang, Huajian Xin, Chuanyang Zheng, Lin Li, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, et al. Lego-prover: Neural theorem proving with growing libraries. *arXiv preprint arXiv:2310.00656*, 2023a.
- Haiming Wang, Ye Yuan, Zhengying Liu, Jianhao Shen, Yichun Yin, Jing Xiong, Enze Xie, Han Shi, Yujun Li, Lin Li, et al. Dt-solver: Automated theorem proving with dynamic-tree sampling guided by proof-level value function. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12632–12646, 2023b.
- Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, et al. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025.
- Wen-Tsun Wu. Basic principles of mechanical theorem proving in elementary geometries. *Journal of automated Reasoning*, 2(3):221–252, 1986.
- xAI. Grok 4 — xai. <https://x.ai/news/grok-4>, July 2025. Accessed on 2025-07-25.
- Ran Xin, Chenguang Xi, Jie Yang, Feng Chen, Hang Wu, Xia Xiao, Yifan Sun, Shen Zheng, and Kai Shen. Bfs-prover: Scalable best-first tree search for llm-based automatic theorem proving. *arXiv preprint arXiv:2502.03438*, 2025.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

- Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, et al. Formalmath: Benchmarking formal mathematical reasoning of large language models. *arXiv preprint arXiv:2505.02735*, 2025.
- Chi Zhang, Jiajun Song, Siyu Li, Yitao Liang, Yuxi Ma, Wei Wang, Yixin Zhu, and Song-Chun Zhu. Proposing and solving olympiad geometry with guided tree search. *arXiv preprint arXiv:2412.10673*, 2024a.
- Hanting Zhang, Daniel Selsam, and Joseph Myers. imo geometry topic in the lean zulip chat archive. <https://leanprover-community.github.io/archive/stream/219941-Machine-Learning-for-Theorem-Proving/topic/IMO.20Geometry.html>, 2022. LeanProver Community Chat, Apr 2022.
- Jiaxin Zhang, Zhongzhi Li, Mingliang Zhang, Fei Yin, Chenglin Liu, and Yashar Moshfeghi. Geoeval: benchmark for evaluating llms and multi-modal models on geometry problem-solving. *arXiv preprint arXiv:2402.10104*, 2024b.
- Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.
- Yichi Zhou, Jianqiu Zhao, Yongxin Zhang, Bohan Wang, Siran Wang, Luoxin Chen, Jiahui Wang, Haowei Chen, Allan Jie, Xinbo Zhang, Haocheng Wang, Luong Trung, Rong Ye, Phan Nhat Hoang, Huishuai Zhang, Peng Sun, and Hang Li. Solving formal math problems by decomposition and iterative reflection. 2025. URL <https://arxiv.org/abs/2507.15225>.

A Analysis of Scalability

A.1 Scalability of SMT

We conduct a series of supplementary experiments to evaluate the scalability of LeanGeo, examining how four key factors influence both compilation time and the number of heartbeats required for proof execution:

- [(1)] The number of basic geometric elements (points, lines, and circles),
- [(2)] The number of given conditions,
- [(3)] The length of the proof, and
- [(4)] The number of applications of the `euclid` tactics.

The scaling curves are shown in Figure 4.

Across all four experiments, LeanGeo exhibits approximately linear scaling: as we increase assumptions, conditions, proof length, or the number of Euclid tactics, both heartbeats and compilation time grow in a strongly correlated, near-linear manner. The only noticeable rises occur when the logical structure becomes denser (e.g., deeper lemma dependencies), which naturally increases the amount of proof search. Overall, the results show that LeanGeo is practically scalable, with performance determined primarily by the expected positive correlation between heartbeats and compilation effort rather than by any pathological geometric cases.

A.2 Complexity versus Lemma Granularity

Our experiments demonstrate that coarse lemma granularity leads to severe blow-ups in both compilation time and heartbeats. When large all-in-one lemmas are inlined directly into a theorem, many nearly identical reasoning steps must be recompiled repeatedly, causing exponential-like scaling.

In contrast, extracting commonly reused intermediate results into separate lemmas keeps the compilation cost close to linear in the dependency depth, because each lemma is compiled

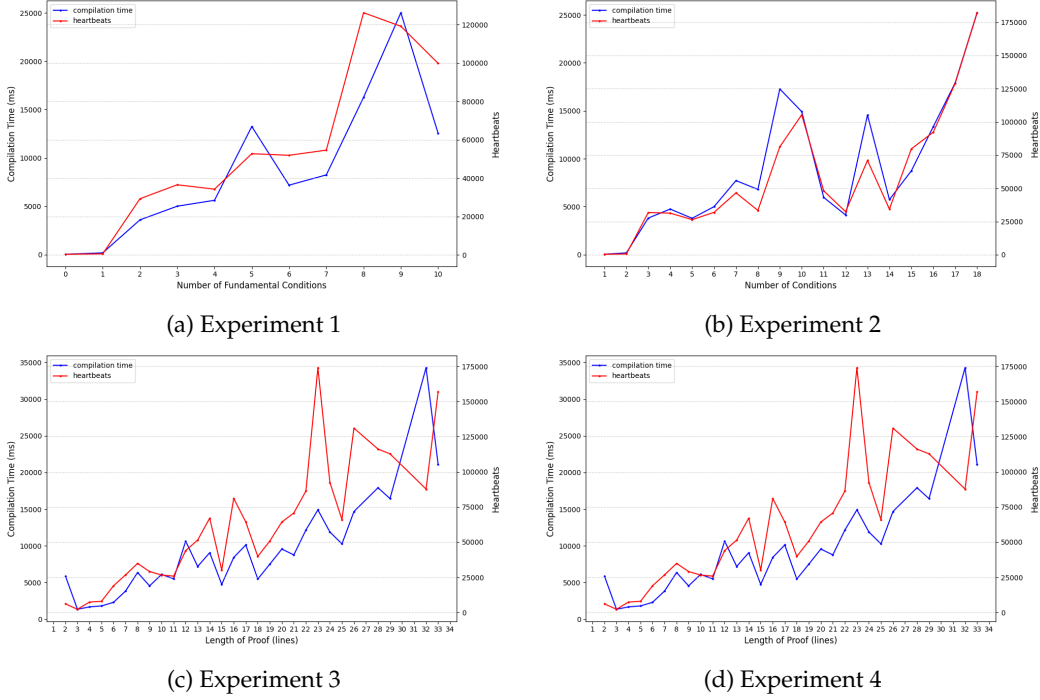


Figure 4: Scaling behavior of heartbeats and compilation time across four experimental settings.

once and then reused. This is precisely why Leans modular proof structure is essential for scalability.

The Table 7 illustrates this effect using Miquels Theorem at different lemma-dependency depths:

Lemma Depth	Compiled One Time		Compiled Multiple Times	
	Heartbeats	Time(ms)	Heartbeats	Time(ms)
0	1.7×10^5	2.1×10^4	1.7×10^5	2.1×10^4
1	6.3×10^5	8.8×10^4	7.6×10^5	1.0×10^5
2	1.4×10^6	2.1×10^5	2.7×10^6	4.0×10^5
3	2.5×10^6	3.6×10^5	8.9×10^6	1.3×10^6
5	4.6×10^6	7.5×10^5	1.0×10^8	1.8×10^7
8	7.5×10^6	1.3×10^6	1.4×10^9	2.2×10^8

Table 7: Scaling behavior of heartbeats and compilation time under different lemma depths.

In the table, “lemma depth” refers to the depth of dependencies referenced back from the current theorem, where a “lemma depth” of 0 indicates the current theorem itself. The right side represents the total compilation resource consumption at that lemma depth. If intermediate theorems are not extracted, a single theorem may need to be written and compiled multiple times. Conversely, extracting them ensures that the intermediate result is compiled only once.

B Command Caching

```

/--
Adds a command for a new constant to the SMT command cache and updates the dependency
graph.

* `oldAxiomExprs`: the expressions corresponding to the types of all currently cached
  axioms.
* `cName`: the name of the axiom to be added to the cache.
* `initialState`: the current state of the global dependency graph.

Returns a tuple of the form `(new global dependency graph, new list of cached axioms,
  list of SMT commands for all of the axioms)`.
-/
def addCommandForConstant
  (oldAxiomExprs : List Expr)
  (cName : Name)
  (initialState : QueryBuilderM.State)
  : MetaM (QueryBuilderM.State × List Expr × List Command) := do
  let constInfo ← getConstInfo cName
  let constExpr := mkConst cName (constInfo.levelParams.map Level.param)
  let (_, st), r ←
    QueryBuilderM.buildDependencyGraph (mkConst `True)
    |>.run { toDefine := oldAxiomExprs ++ [constExpr] : QueryBuilderM.Config }
    |>.run initialState
    |>.run { uniqueFVarNames := {} : TranslationM.State }
  let (_, cmds) ← StateT.run (st.graph.orderedDfs (oldAxiomExprs ++ [constExpr])
    (emitVertex st.commands)) []
  return (st, oldAxiomExprs ++ [constExpr], cmds)

```

Figure 5: Command caching code for SystemE axioms.

C Changes to SystemE formalism

There are some discrepancies between how SystemE axioms are described in the LeanEuclid lean theory vs how they are passed into the SMT solver. In particular degree and length and area are defined directly as functions from Points to a real number. That is the types Angle and Segment do not exist in the SMT query. If a rule involves substituting a function into application into a forall statement it will double the search depth required to obtain that proof. For example if angle degree is defined as Angle.degree (Angle.ofPoints a b c) the smt's search procedure would have to first apply Angle.ofPoints to points a, b, c and then apply Angle.degree to that resultant angle. By contrast, if degree is defined as the measure of three points only a single application is required to obtain the term degree $a b c$. By changing the definition of degree to be a function on three points it halves the search depth required to achieve the same term. Since we generally never reason about segments or angles outside of their measures this simplification is acceptable and segment congruence is defined uniquely by length. For Triangles it is not possible to get rid of the type entirely since Triangle congruence. We can however define a function area' which behaves as an area function on points. When then define Triangle.area (Triangle.ofPoints a b c) = area' a b c. And tag it as a simp lemma. Thus, since simplification is applied before passing into the smt solver, the Triangle type will disappear by the time the smt solver is invoked.

```

opaque area' : Point → Point → Point → ℝ

inductive Triangle
| ofPoints (a b c : Point)

@[simp]
abbrev Triangle.area : Triangle → ℝ :=
  fun x =>
    match x with
    | ofPoints a b c => area' a b c

```

```

notation: max  $\triangle$  a : b : c : 66 => Triangle.ofPoints a b c

instance : Coe Triangle  $\mathbb{R}$  :=
  ⟨Triangle.area⟩

```

Listing 3: Triangle Definition

Besides, to broaden SystemEs applicability to the wider field of geometry, we add nine axioms to LeanGeo covering circles, triangles, similar triangles, and triangle areas, which cannot be derived within the original SystemE.

```

axiom triangle_area_foot :  $\forall$  (a b c d : Point) (BC : Line), b.onLine BC  $\wedge$  c.onLine BC  $\wedge$ 
  (Triangle a b c)  $\wedge$  Foot a d BC  $\rightarrow$  ( $\triangle$  a b c).area = |(a-d)| * |(b-c)|/2

axiom threePoints_existCircle :  $\forall$  (A B C : Point),
  Triangle A B C  $\rightarrow$ 
   $\exists$  ( $\Omega$  : Circle),
  (A.onCircle  $\Omega$   $\wedge$  B.onCircle  $\Omega$   $\wedge$  C.onCircle  $\Omega$ )

axiom exists_centre :  $\forall$  (O : Circle),  $\exists$  (C : Point), C.isCentre O

axiom rightAngle_eq_pi_div_two :  $\perp$  = Real.pi / 2

axiom rightTriangle_sin :  $\forall$  (A B C : Point), RightTriangle A B C  $\rightarrow$  Real.sin ( $\angle$ A:B:C) =
  |(A-C)| / |(B-C)|

axiom rightTriangle_cos :  $\forall$  (A B C : Point), RightTriangle A B C  $\rightarrow$  Real.cos ( $\angle$ A:B:C) =
  |(A-B)| / |(B-C)|

axiom similar_AA :  $\forall$  (A B C D E F : Point), Triangle A B C  $\wedge$  Triangle D E F  $\wedge$   $\angle$ 
  A:B:C =  $\angle$  D:E:F  $\wedge$   $\angle$  B:A:C =  $\angle$  E:D:F  $\rightarrow$  SimilarTriangles A B C D E F

axiom similar_SAS :  $\forall$  (A B C D E F : Point), Triangle A B C  $\wedge$  Triangle D E F  $\wedge$   $\angle$ 
  A:B:C =  $\angle$  D:E:F  $\wedge$  |(A-B)| * |(E-F)| = |(B-C)| * |(D-E)|  $\rightarrow$  SimilarTriangles A B C D
  E F

axiom similar_SSS :  $\forall$  (A B C D E F : Point), Triangle A B C  $\wedge$  Triangle D E F  $\wedge$  |(A-B)| *
  |(E-F)| = |(B-C)| * |(D-E)|  $\wedge$  |(B-C)| * |(F-D)| = |(C-A)| * |(E-F)|  $\rightarrow$ 
  SimilarTriangles A B C D E F

```

Listing 4: Additional Axioms in LeanGeo

The nine axioms form three functional classes.

1. **Definitions** to fix numeric meanings such as a “right angle equals $/2$ ” and “a triangle’s area equals base height $/2$ ”; these are definitional extensions of SystemE, hence conservative and consistency-preserving by the standard metatheorem.
2. **Bridge axioms** stating that in a right triangle $\sin = \text{opposite/hypotenuse}$ and $\cos = \text{adjacent/hypotenuse}$; SystemE has no trigonometry, so these connect its synthetic angles to Mathlib’s analytic $\sin/\cos$, adding a new metric. They are not conservative but hold trivially in the standard $\mathbb{R}^2$ model.
3. **$\mathbb{R}^2$ -valid, SystemE-hard theorems** that for any three points there exists a circle passing through them, that every circle has a center, and the AA/SAS/SSS similarity criteria. These are standard Euclidean facts, yet we could not derive them within SystemE in practice: for each, an expert spent at least three hours attempting a derivation, none of which yielded a compiling proof. We therefore adopt them as axioms. We believe that they’re non-conservative, but could not prove it.

All nine axioms are selected to be valid in the standard real-coordinate model.

D Examples of Formalization

D.1 Examples in Theorem Library

Here is a proof example from the LeanGeo theorem library.

```

theorem angle_lt_outsideCircle: ∀ (A B C D : Point) (AB : Line) (Ω : Circle),
  A.onCircle Ω ∧ B.onCircle Ω ∧ distinctPointsOnLine A B AB ∧ C.onCircle Ω ∧ C ≠
  A ∧ C ≠ B ∧ D.sameSide C AB ∧ ∠ A:D:B < ∠ A:C:B → D.outsideCircle Ω := by
  euclid_intros
  have h1 : ¬ (D.onCircle Ω) := by
    by_contra
    euclid_apply cyclic_eqAngle A B C D AB Ω
    euclid_finish
  have h2: ¬ (D.insideCircle Ω) := by
    by_contra
    euclid_apply line_from_points A D as AD
    euclid_apply intersection_circle_line_extending_points Ω AD D A as E
    have h3: ∠ B:C:A = ∠ B:E:A := by
      euclid_apply cyclic_eqAngle A B C E AB Ω
      euclid_finish
    euclid_apply triangle_exteriorAngle E D B A
    have h4: ∠ A:E:B = ∠ D:E:B := by
      euclid_apply angle_between_transfer A D E B
      euclid_finish
    euclid_finish
  euclid_finish

```

Listing 5: Example of Theorem Library

LeanGeo proofs are structured to mirror the step-by-step, declarative style of traditional, natural-language geometry proofs. This design choice results in simple, readable proof scripts that are particularly amenable to machine learning techniques. The proof development relies on a small set of core tactics:

- `euclid_intros`
This is an initialization tactic that begins the proof. It processes the theorem's statement, automatically introducing all universally quantified variables (e.g., 'A', 'B', 'C', 'D', 'Ω') and hypotheses (e.g., 'A.onCircle Ω', 'D.sameSide C AB') into the local proof context.
- `euclid_apply <rule> <args>`
Given a rule <rule> with type of the form $\forall (<args> : \text{Types}) \dots P \rightarrow Q$, this tactic attempts to prove premise P from the local proof and attempts to prove premise Q from the local proof context using an SMT solver. If successful, proposition Q is added to the proof context.
In this example, `euclid_apply cyclic_eqAngle A B C D AB` refers to the former theorem in the library (in `Circle.lean`)

```

theorem cyclic_eqAngle: ∀ (A B C D: Point) (AB:Line) (Ω : Circle),
  distinctPointsOnLine A B AB ∧ C ≠ A ∧ D ≠ A ∧ C ≠ B ∧ D ≠ B ∧
  A.onCircle Ω ∧ B.onCircle Ω ∧ C.onCircle Ω ∧ D.onCircle Ω ∧
  C.sameSide D AB → ∠ B:C:A = ∠ B:D:A := by ...

```

LeanGeo automatically checks whether all of the premises of `cyclic_eqAngle`, i.e. `distinctPointsOnLine A B AB`, `C ≠ A`, `D ≠ A` ... are satisfied. If yes, then its result, $\angle B:C:A = \angle B:D:A$ will be added in the proof context.

- `euclid.apply <rule> with <args> as <x, h>`
A forward-reasoning tactic designed to apply theorems and construction rules. Given a rule, typically of the form $\forall \dots, P \exists x, Q(x)$ This tactic instantiates it with the provided arguments <args>. It then employs an SMT solver to automatically prove the premise 'P' using hypotheses from the

local context. If successful, the tactic introduces the newly constructed object 'x' and its property 'Q(x)' (named 'h') into the context. This command streamlines geometric constructions and deductions by combining the application of a rule with the automated verification of its preconditions, making the proof script more declarative and readable.

- `euclid.finish`
A terminal tactic that invokes an SMT solver to automatically prove the current goal using the set of available hypotheses in the local context. This tactic is effective for discharging goals that are either direct assumptions or straightforward logical consequences of the premises, requiring minimal search from the solver.
- `have hP : P := by`
A construct for structuring proofs by introducing an intermediate lemma 'P' (named 'hP'). This allows a complex proof to be decomposed into a sequence of smaller, more manageable sub-proofs. This methodology not only enhances the readability and maintainability of the proof script but also improves the SMT solver's performance by reducing its search space. The solver can tackle the smaller lemma in isolation and then utilize the proven result 'hP' in the main proof.

D.2 Formalization of IMO 2001 P1

Problem statement:

Let ABC be an acute-angled triangle with O as its circumcenter. Let P on line BC be the foot of the altitude from A . Assume that $\angle BCA \geq \angle ABC + 30^\circ$. Prove that $\angle CAB + \angle COP < 90^\circ$.

Proof of LeanGeo:

```
import Mathlib
import SystemE
import LeanGeo
open LeanGeo Real

--Consider an acute-angled Triangle ABC. Let P be the Foot of the altitude of
--Triangle ABC issuing from the vertex A, and let O be the circumcenter of
--Triangle ABC. Assume that  $\angle C \geq \angle B + 30^\circ$ . Prove that  $\angle A + \angle COP < 90^\circ$ .
--To Trigonometry.lean
--To Triangle.lean
set_option maxHeartbeats 0

theorem sin_inequality(B C : ℝ)
  (hB : 0 < B ∧ B < π) (hC : 0 < C ∧ C < π)
  (hC1 : C ≥ B + π/6) : 4 * sin B * cos C ≤ 1 := by
  rcases hB with ⟨hB1, hB2⟩
  rcases hC with ⟨hC11, hC22⟩
  have h1 : cos C ≤ cos (B + π / 6) := by
    have h2 : C ≥ B + π / 6 := hC1
    have h3 : C < π := by linarith [hC22]
    have h4 : 0 < B + π / 6 := by
      linarith [hB1, Real.pi_pos]
    have h5 : B + π / 6 < π := by
      nlinarith [hB2, hC11, hC22, Real.pi_pos]
    have h6 : cos C ≤ cos (B + π / 6) := by
      apply Real.cos_le_cos_of_nonneg_of_le_pi
      all_goals
        nlinarith [Real.pi_pos, hB1, hB2, hC11, hC22, Real.pi_pos]
    linarith
  have h2 : sin B * cos (B + π / 6) ≤ 1 / 4 := by
    have h21 : cos (B + π / 6) = cos B * cos (π / 6) - sin B * sin (π / 6) := by
      rw [Real.cos_add]
    have h22 : cos (π / 6) = Real.sqrt 3 / 2 := by
      rw [cos_pi_div_six]
    have h23 : sin (π / 6) = 1 / 2 := by
```

```

    rw [sin_pi_div_six]
  have h24 : sin B * cos (B + π / 6) = (Real.sqrt 3 / 2) * sin B * cos B - (1 / 2) *
    sin B ^ 2 := by
    rw [h21, h22, h23]
    ring_nf
  have h25 : (Real.sqrt 3 / 2) * sin B * cos B - (1 / 2) * sin B ^ 2 ≤ 1 / 4 := by
    nlinarith [sq_nonneg (sin B - 1 / 2), sq_nonneg (cos B - Real.sqrt 3 / 2),
      sq_nonneg (sin B ^ 2 - 1 / 4), sq_nonneg (sin B - Real.sqrt 3 / 2),
      sq_nonneg (cos B ^ 2 - 1 / 4), sq_nonneg (cos B - 1 / 2),
      Real.sqrt_pos.mpr (by linarith : (0 : ℝ) < (3 : ℝ)),
      Real.sqrt_nonneg 3, Real.sq_sqrt (show (0 : ℝ) ≤ (3 : ℝ) by linarith),
      Real.sin_sq_add_cos_sq B, mul_nonneg (show 0 ≤ (0 : ℝ) by linarith) (show 0
        ≤ (0 : ℝ) by linarith),
      Real.sin_pos_of_pos_of_lt_pi hB1 (by linarith : B < Real.pi)]
  linarith [h24, h25]
  have h3 : 0 < sin B := by
    apply sin_pos_of_pos_of_lt_pi
  all_goals linarith [hB1, hB2, Real.pi_pos]
  nlinarith [h1, h2, h3, Real.sin_sq_add_cos_sq B, Real.sin_sq_add_cos_sq C,
    Real.pi_pos]

theorem sin_range (A : ℝ) (hA : 0 < A ∧ A < π/2) : sin A < 1 ∧ sin A > 0 := by
  have h1 : 0 < A := hA.1
  have h2 : A < π / 2 := hA.2
  have h3 : sin A < 1 := by
    have h4 : sin (π / 2) = 1 := by
      rw [sin_pi_div_two]
    have h5 : sin A < sin (π / 2) := by
      apply sin_lt_sin_of_lt_of_le_pi_div_two
    all_goals linarith [Real.pi_pos, Real.pi_gt_three, h1, h2]
  linarith [h4, h5]
  have h6 : sin A > 0 := by
    have h7 : sin (0 : ℝ) = 0 := by
      simp [Real.sin_zero]
    have h8 : sin (0 : ℝ) < sin A := by
      apply sin_lt_sin_of_lt_of_le_pi_div_two
    all_goals linarith [Real.pi_pos, Real.pi_gt_three, h1, h2]
  linarith [h7, h8]
  constructor
  · linarith [h3]
  · linarith [h6]
--To Triangle, Generated b

theorem IMO_2001_P1 :
  ∀ (A B C P O : Point) (AB BC CA : Line),
    formAcuteTriangle A B C AB BC CA ∧
    Foot A P BC ∧
    Circumcentre O A B C ∧
    ∠ A:C:B ≥ ∠ C:B:A + ∟/3 →
    ∠ B:A:C + ∠ C:O:P < ∟ := by
  euclid_intros
  euclid_apply rightAngle_eq_pi_div_two
  euclid_apply acuteTriangle_circumcentre_insideTriangle A B C O AB BC CA
  euclid_apply circle_from_points O B as Ω
  euclid_apply circumcentre_inscribedAngle_comp B C A O BC Ω
  have h0: 4 * sin (∠ B:A:C) * sin (∠ A:B:C) * cos (∠ A:C:B) < 1 := by
    have h1: 0 < ∠ A:B:C ∧ ∠ A:B:C < π := by
      euclid_finish
    have h2: 0 < ∠ A:C:B ∧ ∠ A:C:B < π := by
      euclid_finish
    have h3: (sin (∠ B:A:C) < 1) ∧ (sin (∠ B:A:C) > 0) := by
      euclid_apply sin_range (∠ B:A:C)
    euclid_finish

```

```

have h4:  $\angle A:C:B \geq \angle C:B:A + \pi/6$  := by
  euclid_finish
have h5:  $4 * \sin(\angle A:B:C) * \cos(\angle A:C:B) \leq 1$  := by
  euclid_apply sin_inequality ( $\angle A:B:C$ ) ( $\angle A:C:B$ )
  euclid_finish
nlinarith

have h1: between B P C := by
  euclid_apply acuteTriangle_foot_between A B C P BC
  euclid_finish
have h2:  $|P-C| < |P-O|$  := by
  have h3:  $|P-C| * |P-C| < |P-O| * |P-O|$  := by
    have h4:  $|O-C| * |O-C| - |O-P| * |O-P| = |P-B| * |P-C|$  := by
      euclid_apply ApolloniusTheorem_to_isoTriangle O B C P BC
      euclid_finish
    have h5:  $|P-C| = |A-C| * \cos(\angle A:C:P)$  := by
      euclid_apply rightTriangle_cos P C A
      euclid_finish
    have h6:  $|A-C| = 2 * |O-C| * \sin(\angle A:B:C)$  := by
      euclid_apply LawOfSines_radius B A C O
      euclid_finish
    have h7:  $|B-C| = 2 * |O-C| * \sin(\angle B:A:C)$  := by
      euclid_apply LawOfSines_radius A B C O
      euclid_finish
    have h8:  $\angle A:C:P = \angle A:C:B$  := by
      euclid_apply coll_angles_eq B P C A
      euclid_finish
    have h9:  $|P-C| * |B-C| < |O-C| * |O-C|$  := by
      rw [h5, h6, h7, h8]
    have h10:  $(|O-C| * |O-C|) > 0$  := by euclid_finish
    calc
      _ =  $(4 * \sin(\angle B:A:C) * \sin(\angle A:B:C) * \cos(\angle A:C:B)) * (|O-C| * |O-C|)$  :=
    by linarith
      _ <  $1 * (|O-C| * |O-C|)$  := by euclid_finish
      _ := by euclid_finish
  euclid_finish
euclid_assert  $|P-C| > 0$ 
euclid_assert  $|P-O| > 0$ 
nlinarith
euclid_assert Triangle O C P
euclid_apply triangle_gt_side_gt_angle P C O
have h_final:  $\angle P:C:O = \angle B:C:O$  := by
  euclid_apply coll_angles_eq B P C O
  euclid_finish
euclid_finish

```

Listing 6: Proof of LeanGeo for IMO 2001 P1

A significant advantage of LeanGeo is its seamless integration with Mathlib’s extensive mathematical library, enabling it to tackle a broader class of problems. This is particularly evident in its ability to formalize geometric inequalities, a domain where systems like AlphaGeometry face challenges due to their reliance on converting geometry into polynomial equations. The formalization of IMO 2001 P1, shown above, serves as a prime example. The proof strategy involves reducing the geometric inequality $\angle CAB + \angle COP < \frac{\pi}{2}$ to a trigonometric one: $4 \sin(\angle ABC) \cos(\angle BCA) \leq 1$, derived from the condition $\angle BCA \geq \angle ABC + \frac{\pi}{6}$.

This trigonometric lemma, ‘sin_inequality’, is proven not by geometric tactics. Annotators could obtain the proof from an open-sourced formal prover, Kimina-Prover Wang et al. (2025). The main geometric proof, orchestrated by LeanGeo’s ‘euclid...’ tactics, then imports and applies this analytical result to complete the formalization. This hybrid approach, combining high-level geometric reasoning with deep analytical capabilities from Mathlib, demonstrates LeanGeo’s power in unifying different mathematical domains to expand the scope of automated geometric theorem proving.

E Comparison with AlphaGeometry

E.1 Expressivity

Compared with LeanGeo, AlphaGeometry(Trinh et al., 2024) is built upon a significantly weaker axiomatic foundation. Its formal language cannot express many essential geometric notions, including:

1. inequality and quantitative relations,
2. positional relations (inside, outside, between, same side),
3. existential quantifiers and locus-type assertions,
4. trigonometric functions and general real-number computation,
5. ordered-angle semantics required for precise angular reasoning.

To quantify this gap, we analyzed all 260 theorems in the LeanGeo library and found that **56.% (148 theorems)** are completely inexpressible in AlphaGeometry, **21.2% (55 theorems)** are partially expressible but not semantically equivalent. Only **21.9% (57 theorems)** theorems in LeanGeo can be completely translated in AlphaGeometry’s pattern. On the other hand, **100%** of AlphaGeometry-expressible statements are expressible in LeanGeo.

Below are representative theorems from LeanGeo whose statements cannot be expressed in AlphaGeometry due to limitations of its formal system.

Example 1: Diameter is the longest chord.

```
theorem diameter_longest :
  ∀ (a b c d o : Point) (C : Circle),
    (Diameter a b o C) ∧ (c.onCircle C) ∧ (d.onCircle C)
    → |(a-b)| ≥ |(c-d)| := by
```

AlphaGeometry does not support inequalities, so relations such as $|AB| \geq |CD|$ cannot be expressed at all.

Example 2: Orthocenter of an acute triangle lies inside the triangle.

```
theorem orthocentre_of_acuteTriangle_insideTriangle :
  ∀ (A B C H D E F : Point) (AB BC CA : Line),
    (formAcuteTriangle A B C AB BC CA) ∧
    (Orthocentre H A B C D E F AB BC CA)
    → InsideTriangle H A B C AB BC CA := by
```

AlphaGeometry cannot express “inside/outside” relations or “acute/obtuse” distinctions, making this theorem inexpressible.

Example 3: Existence of a circumcenter

```
theorem exists_circumcentre :
  ∀ (A B C : Point), Triangle A B C →
    ∃ (O : Point), Circumcentre O A B C := by
```

AlphaGeometry lacks existential quantifiers such as “there exists”, so existence theorems cannot be stated.

Example 4: Law of sines (radius form)

```
theorem LawOfSines_radius :
  ∀ (A B C O: Point),
    Triangle A B C ∧ Circumcentre O A B C
    → |(B-C)| = 2 * Real.sin (\angle B:A:C) * |(A-O)| := by
```

AlphaGeometry does not include trigonometric functions and therefore cannot express any theorem involving sin, cos, or angle measure.

Example 5: Cyclic quadrilateral angle relations.

```
theorem cyclic_eq_angles' :
  ∀ (A B C D: Point) (AB : Line) (Ω : Circle),
    distinctPointsOnLine A B AB ∧
    C.sameSide D AB ∧
    A.onCircle Ω ∧ B.onCircle Ω ∧
    C.onCircle Ω ∧ D.onCircle Ω
    → \angle C:A:D = \angle C:B:D := by
```

AlphaGeometry uses unordered “full-angle” equality, which cannot distinguish positional relations or angle orientation, making this theorem not exactly expressible. In AlphaGeometry’s framework, this statement is expressed as “cyclic $A B P Q \Rightarrow \angle P A P B Q A Q B$ ”. This formulation does not account for changes in the relative positions of A, B, P, Q that may cause $\angle APB = \angle AQB$ or $\angle APB + \angle AQB = \pi$.

To further illustrate the differences between our formal system and that of AlphaGeometry in the shared subset of representation, we present the following two examples.

Example 6: Prove that the mid-segment of an isosceles trapezoid $ABCD$ is parallel to AB .

LeanGeo proof:

```
theorem trapezoid_midsegment_parallel_base :
  ∀ (A B C D E F: Point) (AB BC CD DA EF: Line),
    formQuadrilateral A B C D AB BC CD DA ∧
    (¬ AB.intersectsLine CD) ∧ distinctPointsOnLine E F EF ∧
    MidPoint B E C ∧ MidPoint A F D →
    (¬ EF.intersectsLine CD) := by
  euclid_intros
  euclid_apply line_from_points A E as AE
  euclid_apply intersection_lines CD AE as G
  have h1: |(A-E)| = |(E-G)| := by
    euclid_apply trapezoid_imp_similarTriangles_interior B A C G E AB CD
    euclid_apply similar_AA B A E C G E
    euclid_assert |(B-E)| = |(C-E)|
    euclid_apply congruentTriangles_ASA B E A C E G
    euclid_finish
  have h2: ¬ EF.intersectsLine CD := by
    euclid_apply triangleMidsegment_parallel_base A D G F E DA CD AE
    euclid_finish
  euclid_finish
```

Alphageometry Proof:

```
=====
* From theorem premises:
A B C D E F : Points
DC // AB [00]
A,E,C are collinear [01]
EA = EC [02]
F,B,D are collinear [03]
FB = FD [04]

* Auxiliary Constructions:
: Points

* Proof steps:
001. EA = EC [02] & FB = FD [04] ⇒ EA:EC = FB:FD [05]
002. CD // AB [00] & A,E,C are collinear [01] &
    F,B,D are collinear [03] & EA:EC = FB:FD [05]
    ⇒ EF // CD
```

The reason AlphaGeometry produces such a short proof is that its deductive database contains many relatively high-level secondary rules (as shown in step 002). These rules are treated as “axioms inside AlphaGeometry. In contrast, within the LeanGeo framework, we do not freely introduce such axioms. Instead, all basic theorems must be proved from more primitive axioms and inference tools. For instance, in this problem we introduce an auxiliary intersection point of CD and AE , and then complete the proof via congruence and similarity of triangles. As a consequence, our proof is longer but conceptually more instructive.

Example 7: IMO 2000 P1

Two circles G_1 and G_2 intersect at two points M and N . Let AB be the line tangent to these circles at A and B , respectively, so that M lies closer to AB than N . Let CD be the line parallel to AB and passing through the point M , with C on G_1 and D on G_2 . Lines AC and BD meet at E ; lines AN and CD meet at P ; lines BN and CD meet at Q . Show that $EP = EQ$.

Listing 7: IMO 2000 Problem 1

LeanGeo proof:

```
import Mathlib
import SystemE
import LeanGeo
namespace LeanGeo
set_option maxHeartbeats 0
--To circle
--Two circles  $G_1$  and  $G_2$  intersect at two points  $M$  and  $N$ . Let  $AB$  be the line tangent
--to these circles at  $A$  and  $B$ , respectively, so that  $M$  lies closer to  $AB$  than  $N$ .
--Let  $CD$  be the line parallel to  $AB$  and passing through the point  $M$ , with  $C$  on  $G_1$ 
--and  $D$  on  $G_2$ . Lines  $AC$  and  $BD$  meet at  $E$ ; lines  $AN$  and  $CD$  meet at  $P$ ; lines  $BN$ 
--and  $CD$  meet at  $Q$ . Show that  $EP = EQ$ .
theorem IMO_2000_P1 :
  ∀ (M N A B C D E P Q O1 O2 : Point) (G1 G2 : Circle) (AB CD AC BD AN BN : Line),
    CirclesIntersectAtTwoPoints G1 G2 M N ∧
    distinctPointsOnLine A B AB ∧
    TangentLineCircleAtPoint A O1 AB G1 ∧
    TangentLineCircleAtPoint B O2 AB G2 ∧
    ¬ AB.intersectsLine CD ∧
    distinctPointsOnLine M C CD ∧
    C.onCircle G1 ∧ C ≠ M ∧ C ≠ N ∧
    D.onCircle G2 ∧ between C M D ∧
    distinctPointsOnLine A C AC ∧
    distinctPointsOnLine B D BD ∧
    between E A C ∧ between E B D ∧
    distinctPointsOnLine A N AN ∧
    TwoLinesIntersectAtPoint AN CD P ∧
    distinctPointsOnLine B N BN ∧
    TwoLinesIntersectAtPoint BN CD Q →
    |(E-P)| = |(E-Q)| := by
      euclid_intros
      euclid_apply line_from_points M N as MN
      euclid_apply intersection_lines MN AB as T
      have midP_ATB: MidPoint A T B := by
        have h1: |(T-A)| * |(T-A)| = |(T-M)| * |(T-N)| := by
          euclid_apply TangentSecantTheorem T A M N O1 G1 AB
          euclid_finish
        have h2: |(T-B)| * |(T-B)| = |(T-M)| * |(T-N)| := by
          euclid_apply TangentSecantTheorem T B M N O2 G2 AB
          euclid_finish
        have h3: |(T-A)| * |(T-A)| = |(T-B)| * |(T-B)| := by
          rw[h1,h2]
```

```

euclid_assert |(T-A)| > 0
euclid_assert |(T-B)| > 0
have h4: |(T-A)| = |(T-B)| := by
  nlinarith
euclid_finish
have midP_PMQ : MidPoint P M Q := by
  have h1 : |(M-Q)| = |(M-P)| := by
    have h4: |(T-A)| = |(T-B)| := by euclid_finish
    have h5: |(M-Q)| * |(T-A)| = |(M-P)| * |(T-B)| := by
      euclid_apply triangle_parallel_bases_eq_ratio N T A M P B Q AB CD
    euclid_finish
  rw [h4] at h5
  have h6: |(T-B)| > 0 := by euclid_finish
  euclid_finish
  have h2: between P M Q := by
    euclid_finish
  euclid_finish
euclid_apply line_from_points E M as EM
have h_congr: CongruentTriangles A B E A B M := by
  have h1:  $\angle E:A:B = \angle M:A:B$  := by
    have h2:  $\angle E:A:B = \angle E:C:D$  := by
      euclid_apply parallel_imp_eq_alternateExteriorAngles B A D C E AB CD AC
    euclid_finish
  have h3:  $\angle M:A:B = \angle M:C:A$  := by
    euclid_apply line_from_points A M as AM
    have h4: M.sameSide B AC := by
      euclid_finish
    euclid_apply AlternateSegmentTheorem A M C B O1 G1 AM CD AC AB
    euclid_finish
  euclid_finish
  have h5:  $\angle E:B:A = \angle M:B:A$  := by
    have h6:  $\angle E:B:A = \angle E:D:C$  := by
      euclid_apply parallel_imp_eq_alternateExteriorAngles A B C D E AB CD BD
    euclid_finish
  have h7:  $\angle M:B:A = \angle M:D:B$  := by
    euclid_apply line_from_points B M as BM
    have h8: M.sameSide A BD := by
      euclid_finish
    euclid_apply AlternateSegmentTheorem B M D A O2 G2 BM CD BD AB
    euclid_finish
  euclid_finish
  euclid_apply congruentTriangles_ASA A B E A B M
  euclid_finish
have perp_EM_CD: PerpLine EM CD := by
  have h1: PerpBisector E M AB := by
    euclid_apply perpBisector_if_eq_dist E M A B AB
    euclid_finish
  euclid_apply perpBisector_imp_perpLine E M EM AB
  euclid_apply perp_parallel_imp_perp AB EM CD
  euclid_finish
have perpB: PerpBisector P Q EM := by
  euclid_apply (perpBisector_iff P Q EM).mpr
  euclid_finish
euclid_finish

```

Listing 8: Proof of LeanGeo for IMO_2000_P1

Alphageometry Proof:

* Formal statement:

a b = segment a b; c = on_tline c a a b; d = on_tline d b b a; e = on_circle e c a,
 on_circle e d b; f = on_circle f c a, on_circle f d b; g = on_pline g e a b,
 on_circle g c a; h = on_pline h e a b, on_circle h d b; i = on_line i a g, on_line
 i b h; j = on_line j a f, on_line j g h; k = on_line k b f, on_line k g h ? cong i
 j i k

```

=====
* From theorem premises:
A B C D E F G H I J K : Points
AC ⊥ AB [00]
BA ⊥ DB [01]
DE = DB [02]
CE = CA [03]
DF = DB [04]
CF = CA [05]
∠FAE = ∠FAE [06]
GE // AB [07]
CG = CA [08]
∠GAF = ∠GAF [09]
HE // AB [10]
DH = DB [11]
∠FBH = ∠FBH [12]
I,G,A are collinear [13]
I,B,H are collinear [14]
J,F,A are collinear [15]
J,G,H are collinear [16]
BF:BK = BF:BK [17]
G,K,H are collinear [18]
B,F,K are collinear [19]

* Auxiliary Constructions:
: Points

* Proof steps:
001. EG // AB [07] & EH // AB [10] ⇒ EH // EG [20]
002. EH // EG [20] ⇒ E,G,H are collinear [21]
003. DH = DB [11] & DF = DB [04] ⇒ D is the circumcenter of \Delta BHF [22]
004. D is the circumcenter of \Delta BHF [22] & DB ⊥ BA [01] ⇒ ∠ABH = ∠BFH [23]
005. D is the circumcenter of \Delta BHF [22] & DB ⊥ BA [01] ⇒ ∠ABF = ∠BHF [24]
006. E,G,H are collinear [21] & G,K,H are collinear [18] & ∠BFH = ∠ABH [23] & AB // EG [07] ⇒ ∠BFH = ∠KHB [25]
007. E,G,H are collinear [21] & G,K,H are collinear [18] & B,F,K are collinear [19] & ∠BHF = ∠ABF [24] & AB // EG [07] ⇒ ∠BHF = ∠HKB [26]
008. ∠BFH = ∠KHB [25] & ∠BHF = ∠HKB [26] (Similar Triangles) ⇒ BF:BH = BH:BK [27]
009. DF = DB [04] & DH = DB [11] & DE = DB [02] ⇒ E,B,F,H are concyclic [28]
010. DF = DB [04] & DE = DB [02] ⇒ D is the circumcenter of \Delta BFE [29]
011. D is the circumcenter of \Delta BFE [29] & DB ⊥ BA [01] ⇒ ∠EBA = ∠EFB [30]
012. E,G,H are collinear [21] & ∠EFB = ∠EBA [30] & AB // EG [07] ⇒ ∠EFB = ∠BEH [31]
013. E,B,F,H are concyclic [28] & ∠EFB = ∠BEH [31] ⇒ EB = BH [32]
014. CE = CA [03] & CG = CA [08] ⇒ C is the circumcenter of \Delta AEG [33]
015. C is the circumcenter of \Delta AEG [33] & AC ⊥ AB [00] ⇒ ∠BAE = ∠AGE [34]
016. I,G,A are collinear [13] & ∠BAE = ∠AGE [34] & EG // AB [07] ⇒ ∠IAB = ∠BAE [35]
017. DH = DB [11] & DE = DB [02] ⇒ D is the circumcenter of \Delta BHE [36]
018. D is the circumcenter of \Delta BHE [36] & DB ⊥ BA [01] ⇒ ∠ABH = ∠BEH [37]
019. I,B,H are collinear [14] & ∠ABH = ∠BEH [37] & EH // AB [10] ⇒ ∠ABE = ∠IBA [38]
020. ∠IAB = ∠BAE [35] & ∠ABE = ∠IBA [38] (Similar Triangles) ⇒ BI = BE [39]
021. ∠IAB = ∠BAE [35] & ∠ABE = ∠IBA [38] (Similar Triangles) ⇒ AI = AE [40]
022. BF:BH = BH:BK [27] & EB = BH [32] & BI = BE [39] ⇒ IB:BF = BK:IB [41]
023. B,F,K are collinear [19] & I,B,H are collinear [14] & ∠FBH = ∠FBH [12] ⇒ ∠KBI = ∠FBI [42]
024. IB:BF = BK:IB [41] & ∠KBI = ∠FBI [42] (Similar Triangles) ⇒ BK:IK = IB:IF [43]
025. E,B,F,H are concyclic [28] ⇒ ∠FEH = ∠FBH [44]
026. CF = CA [05] & CG = CA [08] & CE = CA [03] ⇒ E,G,F,A are concyclic [45]
027. E,G,F,A are concyclic [45] ⇒ ∠GEF = ∠GAF [46]
028. I,G,A are collinear [13] & I,B,H are collinear [14] & ∠FEH = ∠FBH [44] & EH // AB [10] & ∠GEF = ∠GAF [46] & EG // AB [07] ⇒ ∠IAF = ∠IBF [47]
029. ∠IAF = ∠IBF [47] ⇒ I,B,F,A are concyclic [48]
030. I,B,F,A are concyclic [48] ⇒ ∠IBA = ∠IFA [49]
031. I,B,F,A are concyclic [48] ⇒ ∠IFB = ∠IAB [50]

```

```

032. E,G,H are collinear [21] & G,K,H are collinear [18] & J,F,A are collinear [15] &  $\angle$ 
    IBA =  $\angle$ IFA [49] & I,B,H are collinear [14] &  $\angle$ ABH =  $\angle$ BEH [37] & EH // AB [10] &
    AB // EG [07]  $\Rightarrow$   $\angle$ BEK =  $\angle$ JFI [51]
033. CE = CA [03] & CF = CA [05]  $\Rightarrow$  C is the circumcenter of  $\Delta$ AEF [52]
034. C is the circumcenter of  $\Delta$ AEF [52] & AC  $\perp$  AB [00]  $\Rightarrow$   $\angle$ BAE =  $\angle$ AFE [53]
035. J,G,H are collinear [16] & E,G,H are collinear [21] &  $\angle$ BAE =  $\angle$ AFE [53] & AB // EG
    [07]  $\Rightarrow$   $\angle$ JEA =  $\angle$ AFE [54]
036. J,F,A are collinear [15] &  $\angle$ FAE =  $\angle$ FAE [06]  $\Rightarrow$   $\angle$ JAE =  $\angle$ FAE [55]
037.  $\angle$ JEA =  $\angle$ AFE [54] &  $\angle$ JAE =  $\angle$ FAE [55] (Similar Triangles) $\Rightarrow$  JA:EA = EA:FA [56]
038. EA:FA = JA:EA [56] & IA = EA [40]  $\Rightarrow$  IA:FA = JA:IA [57]
039. I,G,A are collinear [13] & J,F,A are collinear [15] &  $\angle$ GAF =  $\angle$ GAF [09]  $\Rightarrow$   $\angle$ IAF =
     $\angle$ IAJ [58]
040. IA:FA = JA:IA [57] &  $\angle$ IAF =  $\angle$ IAJ [58] (Similar Triangles) $\Rightarrow$   $\angle$ AIF =  $\angle$ IJA [59]
041. B,F,K are collinear [19] & E,G,H are collinear [21] & G,K,H are collinear [18] &
    J,F,A are collinear [15] &  $\angle$ AIF =  $\angle$ IJA [59] & I,G,A are collinear [13] &  $\angle$ IFB =  $\angle$ 
    IAB [50] & AB // EG [07]  $\Rightarrow$   $\angle$ BKE =  $\angle$ FJI [60]
042.  $\angle$ BEK =  $\angle$ JFI [51] &  $\angle$ BKE =  $\angle$ FJI [60] (Similar Triangles) $\Rightarrow$  BE:IF = BK:IJ [61]
043. BK:IK = IB:IF [43] & BE:IF = BK:IJ [61] & BI = BE [39]  $\Rightarrow$  BK:JI = BK:IK [62]
044. BF:BK = BF:BK [17] & BK:JI = BK:IK [62]  $\Rightarrow$  JI = IK
=====

```

Listing 9: Proof of AlphaGeometry for IMO 2000 Problem 1

AlphaGeometry presents the proof as a flat, linear sequence of 44 atomic deductions. While logically sound, this format obscures the underlying geometric narrative. It reads as a symbolic log where high-level concepts, without explicitly grouping these steps into a coherent subgoal.

In contrast, the LeanGeo proof is structured more hierarchically, perfectly reflecting the problems intrinsic geometric structure. The proof is organized into clear, self-contained logical blocks, such as proving ‘midP.ATB’ (T is the midpoint of AB) or ‘perp.EM.CD’. Each block is achieved by invoking powerful theorems in LeanGeo library like ‘TangentSecantTheorem’ and ‘AlternateSegmentTheorem’ mirroring the exact language a mathematician would use. Consequently, the LeanGeo proof is not only verifiable but also intelligible, bridging the gap between a machine-generated proof trace and a human-authored mathematical argument. It demonstrates a system that reasons in a manner remarkably close to natural geometric intuition.

E.2 Verifiability and Soundness

A fundamental requirement for any formal deductive system is **soundness**: every statement that can be derived within the system must be logically valid under the intended semantics. In other words, a proof system is sound if it never proves anything false.

One important limitation of AlphaGeometry is that it can only **generate** correct proofs, but cannot **verify** them. Each proof generated by AlphaGeometry implicitly corresponds to a specific geometric figure, and the deductions are valid only within that configuration. For other admissible figures satisfying the same hypotheses, the conclusion may fail.

Example 8: The internal angle bisector and the external angle bisector are perpendicular.

AlphaGeometry’s Proof:

```

Input:
b c d = triangle b c d; a = on_line a b d; e = angle_bisector e b a c; f =
    angle_bisector f c a d ? perp e a a f
=====
* From theorem premises:
  B C D A E F : Points
  D,A,B are collinear [00]
  \angle BAE = \angle EAC [01]
  \angle CAF = \angle FAD [02]
* Proof steps:

```

```

1. \angle CAF = \angle FAD [02] & D,A,B are collinear [00] ⇒ \angle CAF = \angle FAB [03]
2. \angle BAE = \angle EAC [01] & \angle CAF = \angle FAB [03] (Angle chase) ⇒ AE \perp AF
=====

```

However, if claim that A,E,F are collinear, AlphaGeometry produces a completely contradictory conclusion under exactly the same assumptions.

```

Input:
b c d = triangle b c d; a = on\_line a b d; e = angle\_bisector e b a c; f =
  angle\_bisector f c a d ? coll e a f
=====
* From theorem premises:
B C D A E F : Points
A,D,B are collinear [00]
\angle BAE = \angle EAC [01]
\angle CAF = \angle FAD [02]
* Auxiliary Constructions:
: Points

* Proof steps:
001. \angle CAF = \angle FAD [02] & A,D,B are collinear [00] ⇒ \angle CAF = \angle FAB [03]
002. \angle BAE = \angle EAC [01] & \angle CAF = \angle FAB [03] (Angle chase) ⇒ AE // AF [04]
003. AE // AF [04] ⇒ E,F,A are collinear
=====

```

The core issue is that many of AlphaGeometry’s built-in inference rules are not purely syntactic logical consequences of axioms; instead, they depend on properties of the internal geometric diagram. Since this diagram-based reasoning is not exposed or verified independently of the figure, ambiguous or under-specified statements may lead to incorrect deductions.

LeanGeo, however, is graph-free and handles positional relations with full logical rigor. This inevitably makes its proofs more complex, but we believe it more faithfully reflects the intrinsic nature of geometric reasoning.

Overall, AlphaGeometry is a *task-specialized solving system* tailored for IMO-style geometry problems: it is extremely powerful in problem solving, but this comes at the cost of sacrificing internal axiomatic rigor and omitting several components we believe are equally essential for geometry learners and researchers such as geometric inequalities, trigonometric reasoning, and positional or incidence relations. Its simplified formal system accelerates search and inference but loses part of the rigor and human interpretability. In contrast, our system aims to be more complete, rigorous, and structurally expressive, though this naturally results in more intricate and elaborate reasoning processes.

F LLM Acknowledgments

The authors acknowledge the use of an AI-powered language tool (e.g., ChatGPT, GPT-4) for enhancing the readability of this paper. We wish to clarify that all core ideas, research frameworks, and expressed opinions are original to the authors. Furthermore, all experimental results and data reported are authentic and based on real-world tests conducted by our team. The authors assume full responsibility for the content and integrity of this work.