MA-ProofBench: A Two-Tiered Evaluation of LLMs for Theorem Proving in Mathematical Analysis

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have made notable progress in automated theorem proving, yet existing formal benchmarks remain limited in both mathematical coverage and difficulty. Most are concentrated in areas that are easier to formalize, such as algebra and elementary number theory, and provide limited coverage of subfields that require deeper reasoning, including mathematical analysis. To address this gap, we introduce ***MA-ProofBench***, to the best of our knowledge, the first formal theorem-proving benchmark dedicated to **M**athematical **A**nalysis. The benchmark contains 200 formalized theorems covering 6 core topics and 27 subcategories, including measure and integration theory, complex analysis, and functional analysis. The problems are divided into two difficulty levels, an undergraduate level (Level I, 100 problems) and a Ph.D. qualifying level (Level II, 100 problems), to evaluate how well LLMs perform formal reasoning at different mathematical depths. We evaluate a range of recent general-purpose reasoning models and formal theorem provers on MA-ProofBench. However, most models perform poorly: even the best-performing model, GPT-5.5, achieves only 16% Pass@8 on Level I and 5% on Level II, while most models stay close to 0% on Level II. Further analysis identifies Mathlib hallucinations and incomplete proofs as the two dominant failure modes, while an evaluation on the natural-language version of the benchmark exposes a clear gap between informal and formal reasoning. MA-ProofBench is intended to serve as a reliable reference for tracking progress in formal mathematical reasoning in advanced domains.

1 Introduction

The rapid advancement of Large Reasoning Models (LRMs) (Jaech et al., 2024; Guo et al., 2025) has driven significant progress in end-to-end automated theorem proving. As natural language proofs become increasingly long and intricate, manual verification by human experts has emerged as a major bottleneck for AI-assisted mathematics. Formal verification via interactive theorem provers such as Lean 4 (Moura and Ullrich, 2021) offers a scalable and reliable alternative. Building on this approach, recent models (Lin et al., 2025; Ren et al., 2025; Chen et al., 2025b) have demonstrated the ability to generate non-trivial formal proofs across diverse benchmarks, including challenging problems from the International Mathematical Olympiad (IMO).

Despite this progress, existing formal benchmarks fall short of providing a comprehensive assessment of model capabilities in complex mathematical reasoning. The discriminative power of some traditional benchmarks is diminishing; for instance, Seed-Prover (Chen et al., 2025b) has effectively saturated miniF2F (Zheng et al., 2022), achieving a 100% success rate. Moreover, the quality of the formal statements directly affects the reliability of evaluation. Recent studies (Ammanamanchi and Bhat, 2025; Ospanov et al., 2026) have shown that some existing datasets contain semantic flaws, imprecise statements, or formalizations that do not fully match the intended mathematical meaning of the original problems. More importantly, current benchmarks, such as FIMO (Liu et al., 2023), Putnam (Tsoukalas et al., 2024), ProofNet (Azerbayev et al., 2023), and FormalMATH (Yu et al., 2025), exhibit an uneven topic distribution. Many problems are concentrated in areas that are comparatively easier to formalize, such as algebra, elementary number theory, combinatorics, or discrete structures, while fields that require reasoning jointly about continuity, limits, and topological structures, such as measure theory, complex analysis, and functional analysis, remain underrepresented.

Mathematical Analysis (MA) is a central branch of modern mathematics, studying continuity, limits, and infinite processes. Theorem proving in this

domain is often demanding, as it requires both an understanding of the key structures involved and the ability to decompose a goal into a sequence of verifiable intermediate steps, making it a particularly challenging target for current formal systems. To address the limited coverage of analysis in existing formal benchmarks, we introduce ***MA-ProofBench***, a high-quality, broad-coverage, two-tiered formal benchmark dedicated to mathematical analysis.

The problems in MA-ProofBench are mainly collected from widely used undergraduate analysis textbooks and publicly available Ph.D. qualifying exams, yielding two difficulty levels: *Level I* contains foundational exercises from the standard undergraduate curriculum, while *Level II* comprises more sophisticated analysis problems from doctoral qualifying examinations. To ensure the mathematical fidelity of the formal statements, we employ a human-led, LLM-assisted formalization pipeline coupled with a rigorous independent expert review stage. The resulting dataset provides a robust framework for evaluating the formal reasoning capabilities of models in mathematical analysis.

We evaluate both general-purpose reasoning models and formal theorem provers on MA-ProofBench. The evaluation reveals that both general-purpose reasoning models and formal theorem provers struggle with MA-ProofBench: the best-performing general-purpose reasoning model, GPT-5.5, achieves only 16% Pass@8 on Level I and 5% on Level II, while the strongest theorem prover, DeepSeek-Prover-V2-671B, reaches just 6.86% and 0.44%, respectively. These results show that current models still struggle with formal proofs in advanced analysis. Further analysis attributes the majority of failures to Mathlib hallucinations and incomplete proofs, while an evaluation on the natural-language version of the benchmark exposes a clear gap between informal and formal reasoning.

Our main contributions are summarized as follows:

- We introduce ***MA-ProofBench***, the first formal benchmark dedicated to mathematical analysis. It comprises two difficulty levels (undergraduate and Ph.D.), each with 100 problems, covering 6 core topics and 27 subcategories.

- We propose a human-led, LLM-assisted formalization workflow that addresses the semantic and syntactic challenges of translating advanced analysis problems into Lean 4, thereby ensuring the quality of the resulting formal statements.

- We evaluate a broad range of general-purpose reasoning models and formal theorem provers on MA-ProofBench. The results identify Mathlib hallucinations and incomplete proofs as the dominant failure modes, and further reveal a substantial gap between models' informal and formal reasoning abilities.

2 Related Works

2.1 Formal Theorem Proving

The field of automated theorem proving is evolving from single-paradigm search toward more sophisticated search-augmented and agent-based methods. Since OpenAI's GPT-f (Polu and Sutskever, 2020) adopted best-first search and DeepMind's AlphaProof (Hubert et al., 2025) achieved strong results on IMO-style problems, the open-source community has explored several technical directions. Tree-search-based methods such as InternLM2.5-StepProver (Wu et al., 2025), BFS-Prover (Xin et al., 2025), DeepSeek-Prover-V1.5 (Xin et al., 2024), and HunyuanProver (Li et al., 2024) explore tactic-level tree search algorithms such as Monte Carlo Tree Search. In contrast, whole-proof generation methods such as Kimina-Prover (Wang et al., 2025), DeepSeek-Prover-V2 (Ren et al., 2025), and Goedel-Prover (Lin et al., 2025) generate the entire proof in a single pass. Another line of work augments formal proof search and verification by building agent-based systems, including Seed-Prover (Chen et al., 2025b,a), Aristotle (Achim et al., 2025), Ax-Prover (**?**), and Numina-Lean-Agent (Liu et al., 2026).

2.2 Mathematical Benchmarks

Existing mathematical benchmarks can be broadly categorized as informal or formal. Informal benchmarks including GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), AIME (MAA), OlympiadBench (He et al., 2024), Omni-MATH (Gao et al., 2024), IMO-AnswerBench (Luong et al., 2025), and AMO-Bench (An et al., 2025) focus mainly on numerical problem solving with final-answer evaluation, while IMO-ProofBench (Luong et al., 2025) and ProofBench (Ma et al., 2025) evaluate natural language proof generation. In formal mathematics, early benchmarks such as miniF2F (Zheng et al., 2022) and FIMO (Liu

(a) Category distribution for Level I

(b) Category distribution for Level II

Figure 1: Category distribution of MA-ProofBench across Level I and Level II problems. The inner ring represents high-level mathematical topics, while the outer ring shows their finer-grained subcategories. For readability, only subcategories with relatively high frequency are annotated in the outer ring, although all subcategories are included in the proportional areas. A detailed breakdown of the category distribution is provided in Appendix A.

et al., 2023) primarily cover competition-level problems, ranging from high school competitions to the IMO. Subsequent benchmarks such as ProofNet (Azerbayev et al., 2023), PutnamBench (Tsoukalas et al., 2024), and FormalMATH (Yu et al., 2025) expand the scope to the undergraduate level, but their content remains concentrated in areas such as algebra, topology, and elementary calculus. Recent benchmarks also target specific domains, such as CombiBench (Liu et al., 2025), FATE (Jiang et al., 2026), and LeanCat (Xu et al., 2025). MA-ProofBench follows this domain-specific line of work and focuses on mathematical analysis, a subfield that remains underrepresented in existing formal benchmarks.

3 MA-ProofBench Construction and Characteristics

3.1 Benchmark Overview

The problems in MA-ProofBench are collected primarily from widely used undergraduate textbooks in mathematical analysis and publicly accessible Ph.D. qualifying examination papers. Specifically, the problems are organized according to the Mathematics Subject Classification (MSC), encompassing 6 core categories of analysis: Real Functions, Measure and Integration, Functions of a Complex Variable, Sequences, Series, and Summability, Functional Analysis, and Operator Theory. These are further subdivided into 27 subcategories, with the detailed distribution illustrated in Table 1 and Figure 1.

Level	Description	Source	Count
Level I	Undergrad.	Basic Textbook Exercises	100
Level II	Ph.D.	Top-Tier University Exam Problems	100

Table 1: Difficulty Level Distribution

In terms of topic distribution, Level I primarily focuses on foundational topics in analysis, such as functions of one variable and classical measure theory. In contrast, Level II emphasizes deeper abstract structures, with a strong focus on advanced topics such as linear function spaces in functional analysis, the general theory of linear operators, advanced measure theory, and geometric function theory. Figure 2 shows representative examples for each level.

3.2 Curation Workflow

We constructed the benchmark through a four-stage workflow driven by human-LLM collaboration, as illustrated in Figure 3:

1. **Problem Collection:** We collected raw problems and conducted several rounds of cleaning and proofreading to correct inaccurate statements, standardize the mathematical content, and unify the formatting, removing ambiguous or incomplete entries along the way. This yielded a candidate pool of about 500 problems, from which we selected a representative subset

Example of Level I

Question:
Suppose $f \in L^1(\mu)$. Prove that for each $\epsilon > 0$ there exists a $\delta > 0$ such that

$$\int_E |f| \, d\mu < \epsilon$$

whenever $\mu(E) < \delta$.

Formal Statement:

```
import Mathlib

open MeasureTheory

theorem ma_proofbench_l1_85 {α : Type*} [
    MeasurableSpace α] {μ : Measure α} {f :
     α → ℝ}
    (hf : Integrable f μ) :
    ∀ ε : ℝ, 0 < ε → ∃ δ : ENNReal, 0 < δ
    ∧
      ∀ E : Set α, MeasurableSet E → μ E <
     δ → (∫ x in E, ‖f x‖ ∂μ) < ε := by
  sorry
```

Example of Level II

Question:
Let X be a reflexive complex Banach space, $C \subset X$ be a closed convex subset, show that for all $x_0 \in X$, there exists $y_0 \in C$ such that

$$\|x_0 - y_0\| = \inf_{y \in C} \|x_0 - y\|.$$

Formal Statement:

```
import Mathlib

theorem ma_proofbench_l2_64
  (X : Type*) [NormedAddCommGroup X] [
    NormedSpace ℂ X] [CompleteSpace X]
  (hX_reflexive : Function.Surjective (
    NormedSpace.inclusionInDoubleDual ℂ X))
     (C : Set X)
  (hC_closed : IsClosed C) (_hC_convex :
    Convex ℝ C) (hC_nonempty : C.Nonempty)
    :
  ∀ x₀ : X, ∃ y₀ ∈ C, ‖x₀ - y₀‖ = ⨅ y : C,
    ‖x₀ - (y : X)‖ := by
  sorry
```

Figure 2: Comparison of problem difficulty. The Level I example involves a basic absolute continuity theorem in measure theory with straightforward typing. In contrast, the Level II example requires setting up heavy mathematical machinery, such as reflexive Banach spaces and double duals, illustrating the increased conceptual and syntactic complexity.

of 200 problems.

2. **Formalization:** Human experts first translated each natural-language problem into a draft Lean 4 statement, leaving the proof as `sorry`. They then iterated in a "formalize–feedback–refine" loop, using Lean 4 compiler messages (e.g., type mismatches or unknown identifiers) to fix the code and add missing imports. For unfamiliar Mathlib APIs, experts consulted LLM suggestions and cross-checked them against the official Mathlib documentation. Final decisions were always made by the human experts, ensuring that the formal statements remained faithful to the original problems.

3. **Independent Expert Review:** Each formal statement was then independently reviewed by three additional experts. Each reviewer was required to "reverse-translate" the Lean 4 code back into its intended mathematical statement, ensuring that no implicit assumption was dropped and no condition was inadvertently weakened. A theorem was accepted only if at least two reviewers voted in favor; otherwise, it was sent back to Stage 2 for revision.

4. **Difficulty Grading:** Based on the intrinsic mathematical complexity and the difficulty of formalization, experts scored and categorized the problems into Level I and Level II. The evaluation criteria include:

   - The number of critical sub-tasks or lemmas required for the proof.
   - The level of conceptual abstraction (e.g., elementary limit calculations versus the construction of abstract measure spaces).

3.3 Formalization Standards

To ensure the semantic fidelity of each formal statement to its informal counterpart, MA-ProofBench adheres to a unified set of formalization conventions:

Function Definitions. In informal mathematics, a function is sometimes defined only on a certain subset. In formalization, however, we typically represent it as a total function on the ambient space and make the actual domain of interest explicit through a predicate or a set condition. This treatment keeps

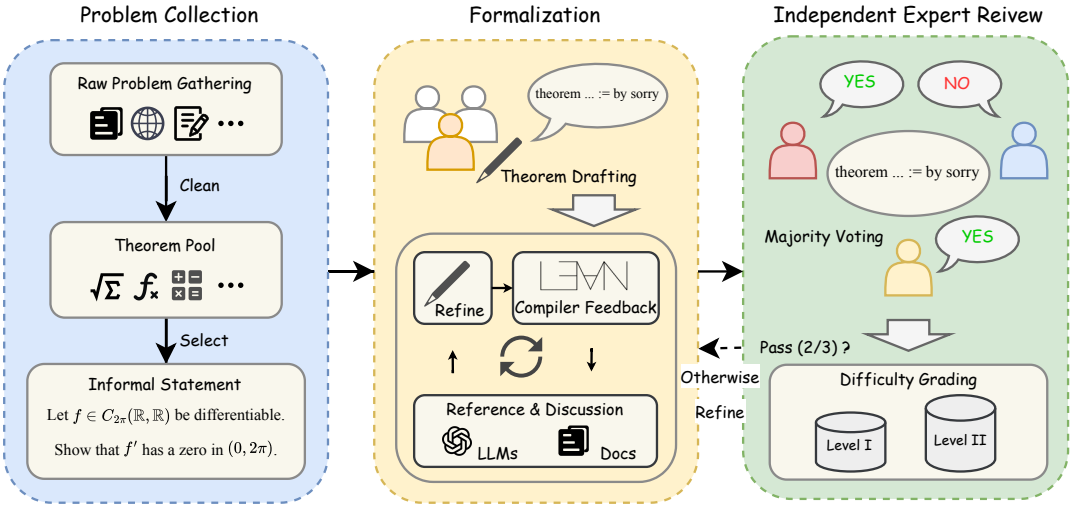


Figure 3: Overview of the curation workflow of MA-ProofBench, comprising Problem Collection, Formalization, Independent Expert Review, and Difficulty Grading.

the function type uniform and avoids frequent coercions between subtypes and the ambient space, thereby allowing us to directly reuse Mathlib's general definitions and theorems for notions such as continuity, differentiability, and harmonicity. In this setting, the values outside the specified subset do not carry mathematical content; the relevant statements are asserted only on the prescribed set or in its neighborhood.

Integration Framework. For integrals in natural-language problem statements whose values lie in a normed linear space, in particular for real- or complex-valued integrals, we formalize them as Bochner integrals in Lean. This convention applies only when the target space carries the relevant normed linear structure; for objects such as extended nonnegative real-valued integrals, we still use Mathlib's dedicated framework for nonnegative Lebesgue integration.

Spatial Structures. Natural-language problem statements sometimes leave implicit the distinction between L^p equivalence classes and pointwise-defined functions, whereas formalization requires this distinction to be made explicit. We therefore choose the representation according to the mathematical structure on which each statement depends. For properties intrinsic to L^p spaces, such as norms, distances, convergence, and bounded linear operators, functions are formalized as equivalence classes modulo almost-everywhere equality, which preserves the standard normed space structure of L^p. For statements involving values at specific points, pointwise inequalities, or pointwise regularity, we instead work with concrete function representatives.

Explicit Premises. Implicit mathematical assumptions are stated explicitly in the formal theorem. Because Lean quantifies over every object satisfying the declared premises, non-degeneracy assumptions that are commonly left tacit in informal writing, such as nonemptiness captured by `[Nonempty X]`, are added directly to the statement. Doing so prevents the formal theorem from being trivially or vacuously satisfied in boundary or degenerate cases, and keeps its semantic scope aligned with that of the original problem.

4 Experiments

4.1 Experimental Setting

We evaluated a range of models on MA-ProofBench, including formal theorem provers (DeepSeek-Prover-V2 (7B and 671B) (Ren et al., 2025), Kimina-Prover-72B (Wang et al., 2025), and Goedel-Prover-V2 (8B and 32B) (Lin et al., 2025)); open-source general-purpose reasoning models, including DeepSeek-V3.2-Thinking (DeepSeek-AI et al., 2025), GLM-5.1 (Z.ai, 2026), Qwen3.5-397B-A17B (Qwen Team, 2026), Qwen3-235B-Thinking-2507 (Team, 2025), Nemotron-3-Nano-30B-A3B (NVIDIA, 2025), and GPT-OSS-120B (High) (OpenAI, 2025); and proprietary models, namely GPT-5.5 (xhigh) (OpenAI, 2026), Gem-

ini 3.1 Pro (High) (Google DeepMind, 2026), and Claude Sonnet 4.6 (High) (Anthropic, 2026). The generation parameters were configured with a maximum output length of 32k tokens and a temperature of 1.0. For evaluation, we used the Pass@k metric (Chen et al., 2021), which is formally defined as follows:

$$\text{Pass@}k = \frac{1}{N}\sum_{i=1}^{N}\left(1 - \frac{\binom{n-c_i}{k}}{\binom{n}{k}}\right),$$

where N is the total number of problems in the benchmark, n denotes the number of candidate proofs generated per problem, and c_i denotes the number of successful proofs among the n candidates for the i-th problem.

For all open-source models, we generated $n = 32$ candidate proofs per problem, while for the proprietary GPT-5.5 (xhigh), Gemini 3.1 Pro (High), and Claude Sonnet 4.6 (High), we used $n = 8$ due to API budget constraints. We therefore adopt Pass@8 as the primary metric for cross-model comparison, and report Pass@32 only as a supplementary result for the models sampled with $n = 32$. All evaluations were run on Mathlib 4.28.0 (mathlib Community, 2020), with Kimina Lean Server (Santos et al., 2025) as the compilation backend for its efficiency. We establish the following criteria for a proof to be considered successful:

- The proof must pass compiler verification without generating any errors.
- The proof must not contain any `sorry` placeholders.
- The proof must include all prerequisite components (e.g., definitions) required by the original problem, while leaving the theorem statement itself strictly unaltered.

The exact prompts used in our evaluation are provided in Appendix C.

4.2 Main Results

Table 2 summarizes the main evaluation results, from which we highlight three observations.

- **Closed-source reasoning models lead overall.** GPT-5.5 (xhigh) attains the highest Pass@8 on Level I (16%) and ties Gemini 3.1 Pro (High) for the best Pass@8 on Level II (5%).
- **Formal theorem provers remain strong on Level I.** DeepSeek-Prover-V2-671B reaches 6.86% Pass@8 and 9% Pass@32 on Level I, outperforming all open-source general-purpose reasoning models on this split. A separate observation is that, despite having only ∼3B active parameters, Nemotron-3-Nano-30B-A3B reaches 2.12% Pass@1 and 3.59% Pass@8 on Level I, clearly above several open-source reasoning models of similar or larger scale. Given that its post-training stage incorporates large-scale Lean 4 formal-proof SFT examples, this result suggests that formal-proof data may help smaller models acquire direct proof-generation ability. On Level II, however, formal theorem provers reach at most 1% Pass@32 and no longer outperform the best open-source general-purpose reasoning model, DeepSeek-V3.2-Thinking, which reaches 2%. This indicates that the theorem prover advantage observed on Level I does not carry over to problems requiring deeper analytical reasoning.
- **Level II exposes a substantial capability gap.** All model families perform much worse on Level II, and even the strongest models solve only a small fraction of the problems. This suggests that this split stresses capabilities beyond local tactic selection or short proof completion, including long-range proof planning, the construction of domain-specific auxiliary lemmas, and reliable use of deeper Mathlib knowledge in mathematical analysis.

5 Further Analysis

5.1 Error Classification in Formal Proofs

To better understand why models fail, we perform an error attribution analysis based on Lean compiler feedback and manual inspection, and summarize the failed proofs into four major error types.

Mathlib Hallucinations. These errors stem from the model referencing fictitious or mismatched Mathlib definitions, theorems, and identifiers, which trigger `Unknown Constant` or `Unknown Identifier` compiler errors. Two common cases are *namespace omission*, where the referenced theorem exists in Mathlib but the model fails to `open` the corresponding namespace, and *name fabrication*, where the model "guesses" and fabricates non-existent theorem names based on Mathlib's naming conventions in an attempt to complete the proof.

Model	Level I			Level II		
	Pass@1	Pass@8	Pass@32	Pass@1	Pass@8	Pass@32
Closed-Source Reasoning Models						
GPT-5.5 (xhigh)	6.50	16.00	–	1.75	5.00	–
Gemini 3.1 Pro (High)	4.38	13.00	–	1.63	5.00	–
Claude Sonnet 4.6 (High)	3.50	6.00	–	0.50	3.00	–
Open-Source Reasoning Models						
DeepSeek-V3.2-Thinking	2.09	5.56	7.00	0.66	1.85	2.00
GLM-5.1	0.06	0.44	1.00	0.00	0.00	0.00
Qwen3.5-397B-A17B	0.87	3.14	5.00	0.00	0.00	0.00
Qwen3-235B-Thinking-2507	0.03	0.25	1.00	0.00	0.00	0.00
Nemotron-3-Nano-30B-A3B	2.12	3.59	4.00	0.00	0.00	0.00
GPT-OSS-120B (High)	0.06	0.50	2.00	0.00	0.00	0.00
Theorem Provers						
DeepSeek-Prover-V2-671B	3.22	6.86	9.00	0.06	0.44	1.00
Kimina-Prover-72B	1.12	3.11	4.00	0.00	0.00	0.00
Goedel-Prover-V2-32B	2.84	5.05	6.00	0.00	0.00	0.00
Goedel-Prover-V2-8B	1.97	3.84	5.00	0.00	0.00	0.00
DeepSeek-Prover-V2-7B	0.94	3.46	6.00	0.00	0.00	0.00

Table 2: Performance evaluation of different models on MA-ProofBench. Accuracies (%) are reported under Pass@1, Pass@8, and Pass@32 settings. Pass@8 is used as the primary cross-model comparison; Pass@32 is reported only for the models sampled with $n = 32$.

Type System Errors. The type of an expression does not match the type expected in context. In analysis, the most common case is confusing $\mathbb{R}$ with `ENNReal` (extended non-negative reals), which suggests that the model does not clearly separate metric values from measure values. This category also covers mismatched function signatures and typeclass synthesis failures, where the model cannot supply the structures required for the proof (e.g., `MetricSpace` or `MeasurableSpace`).

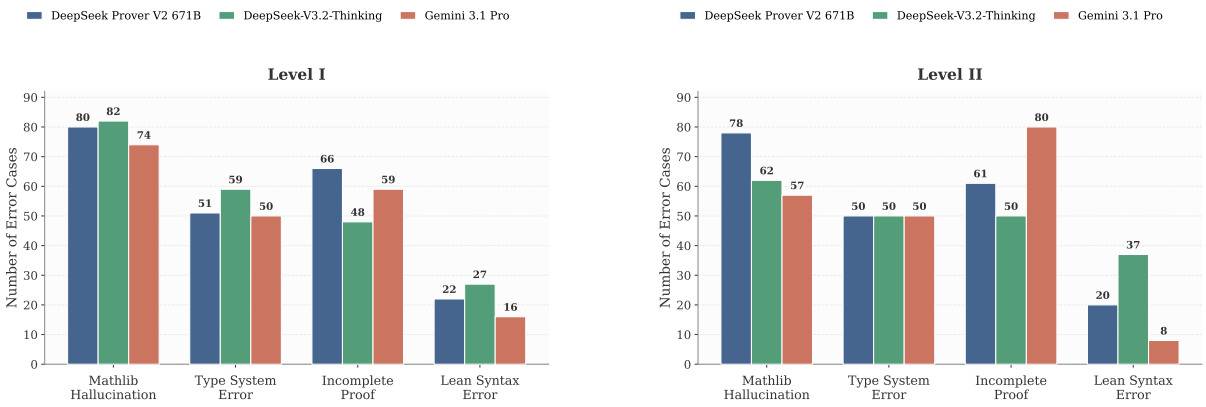
Incomplete Proofs. These proofs contain unresolved sub-goals or `sorry` placeholders. Typical patterns include *tactic misuse*, where the model employs general-purpose tactics for tasks beyond their capabilities, such as using `linarith` for complex non-linear inequalities or expecting `simp` to automate deep analytical reasoning, and *explicit sorry insertion*, where the model deliberately inserts `sorry` to bypass substantial logical leaps or complex sub-proofs.

Lean Syntax Errors. This category refers to code that violates Lean 4 syntax rules, causing the parser to fail. Typical phenomena include *mode confusion*, where the model fails to distinguish the syntactic boundaries between Term Mode and Tactic Mode, and *illegal identifiers*, such as the misuse of reserved words or special symbols (e.g., using λ in inappropriate contexts), or code structures that do not conform to the Lean 4 macro system specifications.

Furthermore, we conducted a random sampling of the failure cases produced by DeepSeek-Prover-V2-671B, DeepSeek-V3.2-Thinking, and Gemini 3.1 Pro across both Level I and Level II problems, with the resulting error distribution summarized in Figure 4. Overall, Mathlib hallucinations and incomplete proofs constitute the primary sources of error. This trend is particularly pronounced for DeepSeek-Prover-V2-671B and DeepSeek-V3.2-Thinking, which yielded 80 and 82 such instances in Level I, and 78 and 62 instances in Level II, respectively. Gemini 3.1 Pro shows a different pattern on Level II: its dominant failure mode is incomplete proofs (80 instances), much higher than the other models, suggesting that it tends to leave proofs unfinished on harder problems. Type system errors occur at a fairly stable rate across models and difficulty levels. Although Lean syntax errors are the least frequent overall, DeepSeek-V3.2-Thinking has more of these errors on Level II (37 instances), suggesting lower syntactic stability on harder problems.

5.2 Gap Between Informal and Formal Reasoning

An inspection of the reasoning traces shows that, on most problems, the models follow a two-stage strategy: an informal proof is first developed within the chain-of-thought, and only then translated into Lean 4 code. Under this pipeline, our main results reveal a phenomenon worth highlighting: several open-source models with otherwise strong general



(a) Error category distribution on Level I

(b) Error category distribution on Level II

Figure 4: Distribution of error categories across representative models on MA-ProofBench. The figure reports the number of failed proof attempts assigned to each error type, including Mathlib hallucination, type system error, incomplete proof, and Lean syntax error.

Model	Level I	Level II
Qwen3-235B-Thinking-2507	66	42
DeepSeek-V3.2-Thinking	85	66
GLM-5.1	90	75

Table 3: Number of fully correct Pass@1 informal proofs per level, as judged by GPT-5.5.

reasoning ability, such as GLM-5.1 and Qwen3-235B-Thinking-2507, score close to zero in the formal setting. A natural question is whether this gap originates from a deficiency in the underlying mathematical reasoning, or from the difficulty of converting an informal proof into Lean 4.

To address this question, we evaluate the same models on the natural-language version of MA-ProofBench. Each model produces a single informal proof per problem under Pass@1, which is then graded by GPT-5.5 on a three-level scale (0, 0.5, or 1) following the protocol of DeepSeekMath-V2 (Shao et al., 2025). Table 3 reports the number of fully correct proofs that received a score of 1, the judging prompt is given in Appendix C.

Juxtaposing these results with the formal evaluation in Table 2 reveals a clear gap between informal and formal reasoning: GLM-5.1 produces 90 and 75 fully correct informal proofs across the two levels, while its Pass@32 in the formal setting remains close to zero, and the same pattern is observed for Qwen3-235B-Thinking-2507. For these models, the bottleneck lies in faithfully expressing such reasoning as Lean 4 code that the compiler accepts.

At the same time, formal performance is not fully determined by informal proficiency. DeepSeek-V3.2-Thinking scores below GLM-5.1 on the informal task, yet remains the strongest open-source general-purpose reasoning model in the formal setting, indicating that familiarity with Mathlib, handling of the type system, and the ability to discharge remaining subgoals all contribute substantially to the final formal score.

6 Conclusion

In this paper, we introduce ***MA-ProofBench***, a formal theorem-proving benchmark dedicated to mathematical analysis, comprising 200 theorems across two difficulty levels (Level I and Level II). Our human-led, LLM-assisted formalization pipeline, combined with independent expert review, ensures broad coverage of analytical subfields, high-quality formal statements, and substantial difficulty for current systems. Both general-purpose reasoning models and formal theorem provers struggle with MA-ProofBench, with Mathlib hallucinations and incomplete proofs as the main bottlenecks, while an evaluation on the natural-language version of the benchmark further exposes a substantial translation gap between informal reasoning and verifiable Lean 4 proofs. These findings suggest that future systems will require better grounding in Mathlib, stronger proof-completion capabilities for analysis-heavy Lean proofs, and more reliable translation from informal proofs into compilable Lean 4 code. We hope MA-ProofBench will serve as a rigorous testbed for advancing formal reasoning in advanced mathematics.

Limitations

Our evaluation may constrained by sampling budget and computational cost, and the reported results may be influenced by different evaluation configurations and inference settings. Future evaluations with larger-scale sampling and more standardized protocols may provide a more comprehensive understanding of model capabilities.

Furthermore, MA-ProofBench mainly evaluates end-to-end Lean proof generation, without explicitly disentangling the different sub-skills involved in formal theorem proving. Future work may introduce more fine-grained diagnostic tasks targeting specific capabilities such as lemma retrieval, premise selection, subgoal decomposition, tactic prediction, and informal-to-formal translation.

References

Tudor Achim, Alex Best, Alberto Bietti, Kevin Der, Mathïs Fédérico, Sergei Gukov, Daniel Halpern-Leistner, Kirsten Henningsgard, Yury Kudryashov, Alexander Meiburg, and 1 others. 2025. Aristotle: Imo-level automated theorem proving. *arXiv preprint arXiv:2510.01346*.

Pawan Sasanka Ammanamanchi and Siddharth Bhat. 2025. Faults in our formal benchmarks. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*.

Shengnan An, Xunliang Cai, Xuezhi Cao, Xiaoyu Li, Yehao Lin, Junlin Liu, Xinxuan Lv, Dan Ma, Xuanlin Wang, Ziwen Wang, and Shuang Zhou. 2025. Amo-bench: Large language models still struggle in high school math competitions. *Preprint*, arXiv:2510.26768.

Anthropic. 2026. Introducing Claude Sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *Preprint*, arXiv:2302.12433.

Jiangjie Chen, Wenxiang Chen, Jiacheng Du, Jinyi Hu, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Wenlei Shi, and 1 others. 2025a. Seed-prover 1.5: Mastering undergraduate-level theorem proving via learning from experience. *arXiv preprint arXiv:2512.17260*.

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, and 1 others. 2025b. Seed-prover: Deep and broad reasoning for automated theorem proving. *arXiv preprint arXiv:2507.23726*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *Preprint*, arXiv:2410.07985.

Google DeepMind. 2026. Gemini 3.1 Pro: Model Card. https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain

Masoom, and 1 others. 2025. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, pages 1–3.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Jiedong Jiang, Wanyi He, Wang Yuefeng, Guoxiong Gao, Yongle Hu, Jingting Wang, Nailin Guan, Peihao Wu, Bryan Dai, Liang Xiao, and Bin Dong. 2026. FATE: A formal benchmark series for frontier algebra of multiple difficulty levels. In *The Fourteenth International Conference on Learning Representations*.

Yang Li, Dong Du, Linfeng Song, Chen Li, Weikang Wang, Tao Yang, and Haitao Mi. 2024. Hunyuan-prover: A scalable data synthesis framework and guided tree search for automated theorem proving. *arXiv preprint arXiv:2412.20735*.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. 2025. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *Preprint*, arXiv:2508.03613.

Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, Ming Zhang, and Qun Liu. 2023. Fimo: A challenge formal dataset for automated theorem proving. *Preprint*, arXiv:2309.04295.

Junqi Liu, Xiaohan Lin, Jonas Bayer, Yael Dillies, Weijie Jiang, Xiaodan Liang, Roman Soletskyi, Haiming Wang, Yunzhou Xie, Beibei Xiong, and 1 others. 2025. Combibench: Benchmarking llm capability for combinatorial mathematics. *arXiv preprint arXiv:2505.03171*.

Junqi Liu, Zihao Zhou, Zekai Zhu, Marco Dos Santos, Weikun He, Jiawei Liu, Ran Wang, Yunzhou Xie, Junqiao Zhao, Qiufeng Wang, and 1 others. 2026. Numina-lean-agent: An open and general agentic reasoning system for formal mathematics. *arXiv preprint arXiv:2601.14027*.

Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu Hoang Trinh, Quoc V Le, and Junehyuk Jung. 2025. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35418–35442, Suzhou, China. Association for Computational Linguistics.

Wenjie Ma, Andrei Cojocaru, Neel Kolhe, Robin Sharif, Haihan Zhang, Vincent Zhuang, Matei Zaharia, and Sewon Min. 2025. Reliable fine-grained evaluation of natural language math proofs. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*.

MAA. American invitational mathematics examination-aime.

The mathlib Community. 2020. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, page 367–381, New York, NY, USA. Association for Computing Machinery.

Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In *International Conference on Automated Deduction*, pages 625–635. Springer.

NVIDIA. 2025. Nemotron 3 Nano: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for Agentic reasoning. Technical report.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

OpenAI. 2026. Introducing GPT-5.5. `https://openai.com/index/introducing-gpt-5-5/`.

Azim Ospanov, Farzan Farnia, and Roozbeh Yousefzadeh. 2026. minif2f-lean revisited: Reviewing limitations and charting a path forward. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *Preprint*, arXiv:2504.21801.

Marco Dos Santos, Hugues de Saxcé, Haiming Wang, Ran Wang, Mantas Baksys, Mert Unsal, Junqi Liu, Zhengying Liu, and Jia Li. 2025. Kimina lean server: A high-performance lean server for large-scale verification. *Preprint*, arXiv:2504.21230.

Zhihong Shao, Yuxiang Luo, Chengda Lu, Z. Z. Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, Xiaokang Zhang, and DeepSeek-AI. 2025. Deepseekmath-v2: Towards self-verifiable mathematical reasoning. *Preprint*, arXiv:2511.22570.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. 2024. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. In *Advances in Neural Information Processing Systems*, volume 37, pages 11545–11569. Curran Associates, Inc.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, and 1 others. 2025. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*.

Zijian Wu, Suozhi Huang, Zhejian Zhou, Huaiyuan Ying, Zheng Yuan, Wenwei Zhang, Dahua Lin, and Kai Chen. 2025. Internlm2. 5-stepprover: Advancing automated theorem proving via critic-guided search. In *2nd AI for Math Workshop@ ICML 2025*.

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. 2024. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *Preprint*, arXiv:2408.08152.

Ran Xin, Chenguang Xi, Jie Yang, Feng Chen, Hang Wu, Xia Xiao, Yifan Sun, Shen Zheng, and Ming Ding. 2025. Bfs-prover: Scalable best-first tree search for llm-based automatic theorem proving. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32588–32599.

Rongge Xu, Hui Dai, Yiming Fu, Jiedong Jiang, Tianjiao Nie, Hongwei Wang, Junkai Wang, Holiverse Yang, Jiatong Yang, and Zhi-Hao Zhang. 2025. Leancat: A benchmark suite for formal category theory in lean (part i: 1-categories). *arXiv preprint arXiv:2512.24796*.

Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, Yandong Wen, Ge Zhang, and Weiyang Liu. 2025. Formalmath: Benchmarking formal mathematical reasoning of large language models. *Preprint*, arXiv:2505.02735.

Z.ai. 2026. GLM-5: From vibe coding to agentic engineering. `https://github.com/zai-org/GLM-5`.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*.

A Category Distribution

Category / Subcategory Name	Level I	Level II
Real Functions		
Functions of One Variable	30	2
Functions of Several Variables	6	2
Polynomials, Rational Functions in Real Analysis	1	0
Inequalities in Real Analysis	6	8
Miscellaneous Topics in Real Functions	1	0
Measure and Integration		
Classical Measure Theory	13	15
Set Functions, Measures and Integrals with Values in Abstract Spaces	0	1
Set Functions and Measures on Spaces with Additional Structure	0	1
Functions of a Complex Variable		
General Properties of Functions of One Complex Variable	9	2
Series Expansions of Functions of One Complex Variable	1	0
Geometric Function Theory	5	10
Entire and Meromorphic Functions of One Complex Variable, and Related Topics	4	3
Miscellaneous Topics of Analysis in the Complex Plane	0	1
Sequences, Series, and Summability		
Convergence and Divergence of Infinite Limiting Processes	5	1
Functional Analysis		
Topological Linear Spaces and Related Structures	1	2
Normed Linear Spaces and Banach Spaces; Banach Lattices	2	6
Inner Product Spaces and Their Generalizations, Hilbert Spaces	2	0
Linear Function Spaces and Their Duals	7	14
Distributions, Generalized Functions, Distribution Spaces	0	7
Measures, Integration, Derivative, Holomorphy	2	1
Commutative Banach Algebras and Commutative Topological Algebras	0	1
Nonlinear Functional Analysis	1	0
Operator Theory		
General Theory of Linear Operators	3	12
Special Classes of Linear Operators	0	5
Ordinary Differential Operators	0	2
Integral, Integro-Differential, and Pseudodifferential Operators	0	4
Nonlinear Operators and Their Properties	1	0
Total	**100**	**100**

Table 4: Distribution of MA-ProofBench problems across MSC categories.

B Case Studies

B.1 Successful Proofs

Success Case (DeepSeek-V3.2-Thinking)

Question:
Let $f \in C_{2\pi}(\mathbb{R}, \mathbb{R})$ be differentiable. Show that f' has a zero in $(0, 2\pi)$.

Output:

```
import Mathlib

theorem ma_proofbench_l1_83 {f : ℝ → ℝ} (hper : Function.Periodic f (2 * Real.pi)) (hdiff :
    Differentiable ℝ f) :
    ∃ c ∈ Set.Ioo 0 (2 * Real.pi), deriv f c = 0 := by
  have πpos : 0 < Real.pi := Real.pi_pos
  have hlt : (0 : ℝ) < 2 * Real.pi := by linarith
  have hcont : ContinuousOn f (Set.Icc 0 (2 * Real.pi)) :=
    hdiff.continuous.continuousOn
  have hdiff_on : DifferentiableOn ℝ f (Set.Ioo 0 (2 * Real.pi)) :=
    hdiff.differentiableOn
```

```
  have h_eq : f (2 * Real.pi) = f 0 := by simpa using hper 0
  rcases exists_deriv_eq_slope f hlt hcont hdiff_on with ⟨c, hc, hc'⟩
  rw [h_eq, sub_self, zero_div] at hc'
  exact ⟨c, hc, hc'⟩
```

Analysis:
The model successfully proves the existence of a zero for the derivative of a differentiable periodic function. It accurately identifies and applies the Mean Value Theorem (`exists_deriv_eq_slope`) from Mathlib. By verifying the prerequisites of continuity and differentiability on the appropriate intervals, it cleverly uses the periodicity condition to deduce a zero slope, effectively concluding the proof.

Success Case (Gemini 3.1 Pro)

Question:
Let H be a Hilbert space and let U be a bounded linear operator acting on H. If U is a unitary operator, and $\sigma(U)$ denotes the spectrum of U, prove that $\sigma(U) \subseteq \{z \in \mathbb{C} : |z| = 1\}$.

Output:

```
import Mathlib

theorem ma_proofbench_l2_73 (H : Type*) [NormedAddCommGroup H] [InnerProductSpace ℂ H] [
    CompleteSpace H]
    (U : H →L[ℂ] H)
    (h₁ : U.adjoint * U = 1) (h₂ : U * U.adjoint = 1) :
    spectrum ℂ U ⊆ {z : ℂ | ‖z‖ = 1} := by
  -- In the C*-algebra of continuous linear maps `H →L[ℂ] H`, `star U` is definitionally `U.
    adjoint`
  have h1_star : star U * U = 1 := h₁
  have h2_star : U * star U = 1 := h₂

  -- We construct the proof that U belongs to the unitary submonoid
  have hu : U ∈ unitary (H →L[ℂ] H) := ⟨h1_star, h2_star⟩
  let u : unitary (H →L[ℂ] H) := ⟨U, hu⟩

  -- Apply the general C*-algebra theorem that bounds the spectrum of a unitary element
  -- to be within the metric sphere of radius 1 centered at 0.
  have h_spec : spectrum ℂ (u : H →L[ℂ] H) ⊆ Metric.sphere (0 : ℂ) 1 :=
    unitary.spectrum_subset_circle u

  -- Now we simply translate the `Metric.sphere` condition to the goal's norm condition
  intro z hz
  have hz' := h_spec hz

  -- `z ∈ Metric.sphere 0 1` means `dist z 0 = 1`.
  -- Since `dist z 0 = ‖z‖`, it simplifies exactly to `‖z‖ = 1`.
  rw [Metric.mem_sphere, dist_zero_right] at hz'
  exact hz'
```

Analysis:
This example demonstrates the model's proficiency in handling advanced functional analysis concepts. To bound the spectrum of a unitary operator on a Hilbert space, the model avoids reasoning from first principles and instead expertly navigates Mathlib's C*-algebra API. It explicitly constructs the unitary submonoid to apply the general theorem `unitary.spectrum_subset_circle`, and then smoothly translates the topological metric condition (`Metric.sphere`) into the algebraic norm condition required by the goal.

B.2 Failed Proofs

Failure Case: Mathlib Hallucination (DeepSeek-Prover-V2-671B)

Question:
Verify that $g : \mathbb{R}^2 \setminus \{0\} \to \mathbb{R}$ defined by

$$g(x) := \log |x|,$$

is harmonic in $\mathbb{R}^2 \setminus \{0\}$.

Output:

```
import Mathlib

theorem ma_proofbench_l1_50 : InnerProductSpace.HarmonicOnNhd (fun x : EuclideanSpace ℝ (Fin
    2) => Real.log ‖x‖) ({0}ᶜ) := by
  have h_main : ∀ (x : EuclideanSpace ℝ (Fin 2)), x ∈ ({0}ᶜ : Set (EuclideanSpace ℝ (Fin 2)))
      → ∃ (u : Set (EuclideanSpace ℝ (Fin 2))) (hu : u ∈ 𝒩[({0}ᶜ : Set (EuclideanSpace ℝ (
    Fin 2)))] x), InnerProductSpace.HarmonicOn (fun x : EuclideanSpace ℝ (Fin 2) => Real.
    log ‖x‖) u := by
    intro x hx
    have hx' : x ≠ 0 := by simpa using hx
    use ({0}ᶜ : Set (EuclideanSpace ℝ (Fin 2)))
    use?
    sorry
  intro x hx
  have h₁ : ∃ (u : Set (EuclideanSpace ℝ (Fin 2))) (hu : u ∈ 𝒩[({0}ᶜ : Set (EuclideanSpace ℝ
      (Fin 2)))] x), InnerProductSpace.HarmonicOn (fun x : EuclideanSpace ℝ (Fin 2) => Real.
    log ‖x‖) u := by
    apply h_main
    exact hx
  obtain ⟨u, hu, h₂⟩ := h₁
  exact ⟨u, hu, h₂⟩
```

Analysis:
This proof fails due to a **Mathlib Hallucination**. The model fabricates a nonexistent theorem identifier, `InnerProductSpace.HarmonicOn`. In Mathlib 4, the correct identifier for this context should be `InnerProductSpace.HarmonicOnNhd`. Additionally, the model is unable to resolve the neighborhood filter logic and ultimately resorts to an explicit `sorry` to bypass the sub-goal.

Failure Case: Type System Error (DeepSeek-Prover-V2-7B)

Question:
Suppose ν, ν_1, ν_2 are signed measures on $(X, \mathcal{M})$ and μ a (positive) measure on $\mathcal{M}$. Prove $\nu_1 \perp \nu_2$ implies $|\nu_1| \perp |\nu_2|$.

Output:

```
import Mathlib

theorem ma_proofbench_l1_98 {X : Type*} [MeasurableSpace X]
    (ν₁ ν₂ : MeasureTheory.SignedMeasure X)
    (h : MeasureTheory.VectorMeasure.MutuallySingular ν₁ ν₂) :
    (MeasureTheory.Measure.MutuallySingular ν₁.totalVariation ν₂.totalVariation) := by
  have h_main : MeasureTheory.Measure.MutuallySingular ν₁.totalVariation ν₂.totalVariation :=
    by
    -- Use the fact that ν₁ and ν₂ are mutually singular to find a set B such that ν₁(B) = 0
    and ν₂(Bᶜ) = 0
    have h₁ : ∃ (B : Set X), MeasurableSet B ∧ ν₁ B = 0 ∧ ν₂ Bᶜ = 0 := by
      exact h
    obtain ⟨B, hB_m, hB₁, hB₂⟩ := h₁
    -- Show that the total variation of ν₁ at B is 0
    have h₂ : ν₁.totalVariation B = 0 := by
      -- Compute the total variation of ν₁ at B
      have h₃ : ν₁.totalVariation B = |ν₁ B| := by
        -- Use the definition of total variation for a signed measure
        simp [MeasureTheory.SignedMeasure.totalVariation, abs_eq_self.mpr (le_of_lt (show (0 :
      ℝ≥0) < 1 by norm_num))]
      rw [h₃]
      -- Simplify |ν₁ B| since ν₁ B = 0
      have h₄ : |ν₁ B| = 0 := by
        simp [hB₁]
        <;> aesop
      rw [h₄]
    -- Show that the total variation of ν₂ at Bᶜ is 0
    have h₃ : ν₂.totalVariation Bᶜ = 0 := by
      -- Compute the total variation of ν₂ at Bᶜ
```

```
      have h₄ : ν₂.totalVariation Bᶜ = |ν₂ Bᶜ| := by
        -- Use the definition of total variation for a signed measure
        simp [MeasureTheory.SignedMeasure.totalVariation, abs_eq_self.mpr (le_of_lt (show (0 :
    ℝ≥0) < 1 by norm_num))]
      rw [h₄]
      -- Simplify |ν₂ Bᶜ| since ν₂ Bᶜ = 0
      have h₅ : |ν₂ Bᶜ| = 0 := by
        simp [hB₂]
        <;> aesop
      rw [h₅]
    -- Conclude that ν₁.totalVariation and ν₂.totalVariation are mutually singular
    refine' ⟨B, hB_m, _⟩
    simp_all [MeasureTheory.Measure.MutuallySingular, h₂, h₃]
    <;> aesop
  exact h_main
```

Analysis:
This represents a classic **Type System Error**. The model attempts to directly equate ν_1`.totalVariation B` with `|`ν_1 `B|`. However, the former evaluates to a measure value of type `ENNReal` (extended non-negative reals), whereas the latter evaluates to a standard metric value of type $\mathbb{R}$. Placing these two disparate types on opposite sides of an equality operator triggers a strict type mismatch failure during compiler verification.

Failure Case: Incomplete Proof (Claude Sonnet 4.6)

Question:
The function $f \in C^1(\mathbb{R}^2, \mathbb{R})$ satisfies $\partial_1 f = \partial_2 f$ and $f(0,0) = 0$. Show that there is a $g \in C(\mathbb{R}^2, \mathbb{R})$ such that $f(x,y) = g(x,y)(x+y)$ for $(x,y) \in \mathbb{R}^2$.

Output:

```
import Mathlib

theorem ma_proofbench_l1_02 {f : ℝ × ℝ → ℝ} (hf : ContDiff ℝ 1 f)
    (hpar : ∀ p : ℝ × ℝ, (fderiv ℝ f p) (1, 0) = (fderiv ℝ f p) (0, 1))
    (h0 : f (0, 0) = 0) :
    ∃ g : ℝ × ℝ → ℝ, Continuous g ∧
      ∀ x y : ℝ, f (x, y) = g (x, y) * (x + y) := by
  let g : ℝ × ℝ → ℝ := fun p =>
    if p.1 + p.2 = 0 then
      (fderiv ℝ f (0, 0)) (1, 1) / 2
    else
      f p / (p.1 + p.2)
  use g
  constructor
  · -- Continuity of g
    have hf_cont : Continuous f := hf.continuous
    have hf_diff : Differentiable ℝ f := hf.differentiable le_rfl
    sorry
  · intro x y
    by_cases h : x + y = 0
    · simp only [g, h, ite_true, mul_zero]
      sorry
    · simp only [g, h, ite_false]
      field_simp
```

Analysis:
This output exemplifies an **Incomplete Proof** characterized by explicit `sorry` insertion. While the model correctly defines the piecewise function $g(x,y)$ to satisfy the conditions, it struggles with the substantial logical leaps required to prove the continuity of this piecewise construction at the singularity boundary ($x + y = 0$). Consequently, it abandons the proof by inserting `sorry` placeholders.

Failure Case: Lean Syntax Errors (DeepSeek-V3.2-Thinking)

Question:
Let E be an n.v.s. and let $C \subset E$ be convex. Deduce that $\overline{C} = \overline{\operatorname{Int} C}$ whenever $\operatorname{Int} C \neq \varnothing$.

Output:

```
import Mathlib
open Set

theorem ma_proofbench_l1_88 {E : Type*} [NormedAddCommGroup E] [NormedSpace ℝ E]
    {C : Set E} (hC : Convex ℝ C) (hInt : (interior C).Nonempty) :
    closure (interior C) = closure C := by
  apply Set.Subset.antisymm
  · exact closure_mono (interior_subset C)
  · intro x hx
    rcases hInt with ⟨y, hy⟩
    set f : ℝ × E → E := fun p => (1 - p.1) • y + p.1 • p.2 with hf_def
    have h_cont : Continuous f := by
      unfold f
      continuity
    have h_cont_at : ContinuousAt f (1, x) := h_cont.continuousAt
    have h_closure : (1, x) ∈ closure (Ioo (0 : ℝ) 1 ×ˢ C) := by
      rw [closure_prod_eq]
      constructor
      · have : (1 : ℝ) ∈ closure (Ioo (0 : ℝ) 1) := by
          rw [closure_Ioo (by norm_num)]
          exact ⟨by norm_num, by norm_num⟩
        exact this
      · exact hx
    have h_image : f '' (Ioo (0 : ℝ) 1 ×ˢ C) ⊆ interior C := by
      rintro _ ⟨⟨λ, w⟩, ⟨hλ1, hλ2⟩, hw⟩, rfl⟩
      unfold f
      have H : λ • w + (1 - λ) • y ∈ interior C :=
        hC.combo_interior_self_subset_interior hw hy (by linarith) (by linarith) (by ring)
      simpa [add_comm] using H
    have H1 : f (1, x) ∈ closure (f '' (Ioo (0 : ℝ) 1 ×ˢ C)) :=
      h_cont_at.mem_closure_image h_closure
    have H2 : closure (f '' (Ioo (0 : ℝ) 1 ×ˢ C)) ⊆ closure (interior C) :=
      closure_mono h_image
    have : f (1, x) = x := by simp [f]
    rw [this] at H1
    exact H2 H1
```

Analysis:
This proof suffers from an Illegal Identifier syntax error. In the destructuring step `rintro _ ⟨⟨λ, w⟩⟩, ...`, the model attempts to bind a real number to the variable name λ. However, λ (`\lambda`) is a strictly reserved system keyword in Lean (utilized for lambda abstractions or anonymous functions). Utilizing it as a local variable identifier violates Lean 4 lexical rules and immediately breaks the parser.

C Prompts

Prompt for Goedel-Prover-V2 and DeepSeek-Prover-V2

```
Complete the following Lean 4 code:

```lean4
{formal_statement}
```
Before producing the Lean 4 code to formally prove the given theorem, provide a detailed proof
↪ plan outlining the main proof steps and strategies.
The plan should highlight key ideas, intermediate lemmas, and proof structures that will guide
↪ the construction of the final formal proof.
```

Prompt for Kimina-Prover

```
You are an expert in mathematics and proving theorems in Lean 4. Think about and solve the
↪  following problem step by step in Lean 4.
Problem:{informal_statement}

Formal statement:

```lean4
{formal_statement}
```
```

Prompt for General-Purpose Models

```
You are an expert in Lean 4 and Mathematics. Please finish the following proof in Lean4 code.
Do not change the original statement. Copy the final statement to prove exactly.
Please include the complete header (including imports and namespaces) so that your code can
↪  pass the Lean4 compiler. Please solve the statement step by step and provide your complete
↪  Lean4 code between ```lean4 and ``` after careful reasoning.
The statement for you to complete is:

```lean4
{formal_statement}
```
```

Prompt for GPT-5.5 Informal Proof Grading

```
## Instruction

Your task is to evaluate the quality of a solution to a problem. The problem may ask for a proof
↪  of statement, or ask for an answer. If finding an answer is required, the solution should
↪  present the answer, and it should also be a rigorous proof of that answer being valid.

Please evaluate the solution and score it according to the following criteria:
- If the solution is completely correct, with all steps executed properly and clearly
↪  demonstrated, then the score is 1
- If the solution is generally correct, but with some details omitted or minor errors, then the
↪  score is 0.5
- If the solution does not actually address the required problem, contains fatal errors, or has
↪  severe omissions, then the score is 0
- Additionally, referencing anything from any paper does not save the need to prove the
↪  reference. It's okay IF AND ONLY IF the solution also presents a valid proof of the
↪  reference argument(s); otherwise, if the solution omits the proof or if the proof provided
↪  is not completely correct, the solution should be scored according to the criteria above,
↪  and definitely not with a score of 1

Please carefully reason out and analyze the quality of the solution below, and in your final
↪  response present a detailed evaluation of the solution's quality followed by your score.
↪  Therefore, your response should be in the following format:

Here is my evaluation of the solution:
... // Your evaluation here. You are required to present in detail the key steps of the
↪  solution or the steps for which you had doubts regarding their correctness, and explicitly
↪  analyze whether each step is accurate: for correct steps, explain why you initially
↪  doubted their correctness and why they are indeed correct; for erroneous steps, explain
↪  the reason for the error and the impact of that error on the solution.

Based on my evaluation, the final overall score should be:
\boxed{...} // where ... should be the final overall score (0, 0.5, or 1, and nothing else)
↪  based on the above criteria

---

Here is your task input:

## Problem
{statement}
```

```
## Solution
{proof}
```