MATHADV: BEYOND PROOF ACCURACY IN ADVANCED MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

A verified proof establishes correctness for a formal statement, but aggregate proof accuracy offers limited insight into the capabilities behind model success or failure. We introduce MathAdv to probe these capabilities across advanced mathematics and test whether proof success persists under equivalent reformulations. The benchmark contains 321 undergraduate- and graduate-level problems across 13 domains, including 298 with Lean 4 statements. Alongside theorem proving, suitable problems have up to three auxiliary tasks: multiple-choice questions probing mathematical knowledge, direct-answer questions assessing informal reasoning without Lean proof construction, and expert-crafted reformulations testing robustness to problem presentation. Our evaluations reveal limitations that proof accuracy alone obscures: models can identify promising mathematical approaches yet fail to construct valid Lean proofs, and their performance varies substantially across subjects and equivalent formulations. Natural-language guidance further exposes model-dependent differences, helping general-purpose LLMs but sometimes hindering proof-specialized models. Together, these findings motivate evaluating not only whether models produce valid proofs, but also the mathematical capabilities underlying their performance and the conditions under which they succeed. The dataset and evaluation scripts are available at `https://github.com/mathadv912-design/mathadv-iclr.git`.

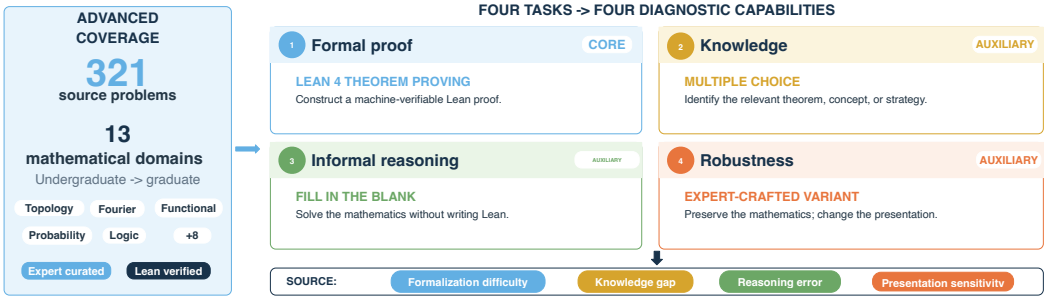


Figure 1: MathAdv at a glance.

1 INTRODUCTION

Mathematical reasoning is a fundamental benchmark for assessing AI because it requires abstract understanding, logical deduction, and multi-step reasoning beyond memorization Hendrycks et al. (2021); Mirzadeh et al. (2025). It is also central to scientific discovery and other domains requiring reliable, verifiable conclusions, making its rigorous evaluation increasingly important Lewkowycz et al. (2022); Ríos-García et al. (2026). Early benchmarks primarily assessed informal natural-language solutions by their final answers, providing limited guarantees about the validity of intermediate reasoning Lewkowycz et al. (2022); Luo et al. (2025). Formal theorem proving offers a more rigorous alternative: models construct machine-verifiable proofs in assistants such as Lean Moura & Ullrich (2021), Coq Bertot & Castéran (2004), and Isabelle Nipkow et al. (2002), enabling automatic verification and precise feedback on invalid proof attempts Li et al. (2024); Hubert et al. (2026). Recent advances in language models have further strengthened proof-generation systems and intensified interest in formal reasoning Lin et al. (2025b); Ren et al. (2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

However, verifying a proof and understanding a model's mathematical capabilities are different evaluation goals. A proof score records success on formal statements, but leaves three questions unresolved. **First, what capabilities underlie success or failure?** Theorem proving draws on mathematical knowledge, logical deduction, and formal proof construction. Aggregate accuracy alone cannot distinguish a knowledge gap from a reasoning error or difficulty expressing an argument in Lean. **Second, across which mathematical domains do these capabilities hold?** Existing benchmarks have largely focused on competition-level problems drawn from high-school and undergraduate mathematics, with an emphasis on algebra and number theory Liu et al. (2023); Tsoukalas et al. (2024), leaving broader advanced mathematics less explored. **Third, does success persist when the formulation changes?** Evaluations generally use a single, fixed formulation, although language-model reasoning can be sensitive to minor changes in problem presentation Mirzadeh et al. (2025). Concerns about data contamination and memorization further motivate testing beyond familiar formulations Gardner et al. (2020); Kaushik et al. (2020); Sakaguchi et al. (2021). Together, these questions call for evaluation beyond proof accuracy: probing the capabilities behind a score, their domain coverage, and their robustness.

We introduce MathAdv, a benchmark designed to investigate these three questions within a unified evaluation framework. **First, paired tasks probe the capabilities behind proof performance.** Alongside Lean theorem proving, multiple-choice questions probe mathematical knowledge and direct-answer questions assess informal reasoning without requiring Lean proof construction. Comparing these task outcomes helps identify potential knowledge gaps, reasoning errors, and formal proof construction difficulties. **Second, broad coverage reveals domain-specific strengths and limitations.** The benchmark contains 321 undergraduate- and graduate-level problems across 13 domains, including topology, Fourier analysis, and functional analysis. Of these, 298 have Lean 4 statements; the remaining 23 are retained for auxiliary evaluations pending adequate Mathlib support. **Third, equivalent reformulations test the stability of proof success.** Expert-crafted variants preserve the underlying mathematical content while substantially changing its presentation. Each suitable problem has up to three auxiliary evaluations, connecting capability-level analysis with domain coverage and robustness testing.

Constructing MathAdv poses two main technical challenges: designing paired tasks that probe complementary capabilities while preserving their connection to the original mathematics, and producing Lean statements that are both type-correct and semantically faithful despite uneven Mathlib coverage. We address the first through domain-expert curation and independent review: experts select nonredundant problems, construct the auxiliary tasks, and verify that transformed variants retain the reasoning required by the originals. We address the second with an LLM-assisted, verifier-in-the-loop formalization pipeline that combines compiler checks with independent semantic review. An LLM drafts each Lean statement, compiler feedback guides iterative correction, an independent LLM and a domain expert check semantic fidelity, and a second expert performs final review.

Our evaluations show why these complementary views matter: proof performance alone obscures gaps between mathematical reasoning and formal proof construction, as well as sensitivity to domain and presentation. **First, a promising mathematical approach does not guarantee a valid Lean proof.** Models often identify a useful proof direction but fail to formalize it successfully. Reasoning hints improve general-purpose LLMs but can degrade proof-specialized theorem provers, further showing that informal guidance does not translate uniformly into proof success. **Second, mathematical competence is uneven across domains.** Performance varies substantially by subject, so aggregate accuracy conceals domain-specific strengths and weaknesses. **Third, proof success is sensitive to equivalent reformulations.** Most models solve original problems substantially more often than their expert-crafted variants, limiting the extent to which success on one presentation establishes robustness. These findings support evaluating the capabilities and conditions behind proof success, rather than treating a single proof score as a complete account of mathematical reasoning.

Our contributions are as follows:

1. **An evaluation framework beyond proof accuracy:** paired tasks that probe mathematical knowledge, informal reasoning, and formal proof construction, with equivalent reformulations to test robustness.
2. **An expert-reviewed benchmark spanning advanced mathematics:** 321 problems across 13 domains, including 298 Lean 4 statements checked for compilability and semantic fidelity, enabling domain-level analysis of these capabilities.

3. **A systematic evaluation of contemporary theorem provers:** evidence that promising mathematical approaches can fail at formal proof construction, performance is uneven across domains, and proof success is sensitive to equivalent reformulations and model-dependent use of guidance.

2 Related work

Formal mathematics reasoning. Formal mathematics encodes statements and proofs within a logical system implemented by a proof assistant such as Lean Moura & Ullrich (2021). Mathematical objects and definitions must be built from concepts recognized by the system, and every derivation must follow its logical rules. The proof assistant therefore verify each step and certify that the conclusion follows from the stated assumptions.

Existing benchmark datasets. Recent benchmarks evaluate formal mathematical reasoning in large language models across competition mathematics, informal-to-formal translation, formal theorem collections etc., but typically focus on specific sources, domains, or capabilities Zheng et al. (2021); Azerbayev et al. (2023a); Liu et al. (2023); Tsoukalas et al. (2024); Yu et al. (2025); Liu et al. (2026); Biyani et al. (2025). In contrast, MathAdv spans 13 domains in undergraduate- and graduate-level mathematics and includes direct-answer, multiple-choice, and transformed questions for diagnosis beyond theorem-proving accuracy alone. Appendix A.1 provides a detailed comparison.

Formal theorem provers Recent work has developed a wide range of large language model approaches for both informal mathematical reasoning and formal theorem proving. Formal theorem provers range from direct proof generators to systems that use search, verifier feedback, large formal training corpora, or multiple reasoning agents Polu & Sutskever (2020); Azerbayev et al. (2023b); Ying et al. (2024); Lin et al. (2024); Shen et al. (2025); Lin et al. (2025a); Xin et al. (2024); Wang et al. (2024); Gao et al. (2024); Wang et al. (2025). These approaches differ primarily in how they generate candidate proofs and recover from errors; Appendix A.2 provides a detailed review.

3 MathAdv: advanced benchmarking of formal math reasoning

Section 3.1 introduces MathAdv and describes three complementary diagnostic question types that test natural-language problem solving, knowledge of relevant theorems and proof strategies, and robustness to equivalent reformulations. These diagnostic tasks allow us to disentangle these component abilities from formal proof construction, which requires combining them to produce a precise, verifiable proof in Lean. In Section 3.2, we present the human-in-the-loop pipeline used to translate problems into Lean 4.

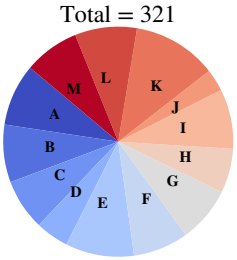
3.1 MathAdv

MathAdv contains 321 problems drawn from undergraduate- and graduate-level mathematics textbooks or contributed by domain experts. The problems span 13 domains—number theory, linear algebra, abstract algebra, calculus, real analysis, complex analysis, Fourier analysis, functional analysis, probability, topology, geometry, combinatorics, and logic—and are selected to maximize coverage while minimizing redundancy. We formalize 298 problems in Lean 4; the remaining 23 are retained for auxiliary evaluations and deferred for future formalization because of current gaps in Mathlib. For each suitable problem, we construct up to three auxiliary tasks: a direct-answer problem, a multiple-choice reasoning question, and an expert-crafted transformed problem. Figure 2 shows the domain distribution alongside an example of the diagnostic tasks.

Direct-answer problems. We recast suitable proof problems as direct-answer questions that ask the model to compute a target quantity or expression without writing a Lean proof. These questions test whether the model can solve the underlying mathematics in natural language, thereby helping separate informal problem solving from formal proof construction. Their domain distribution is shown in Figure 8.

Multiple-choice problems. We ask models to identify the theorem, concept, or reasoning strategy most relevant to the original problem from several plausible options. These questions probe whether a model has the background knowledge needed to approach the proof in natural language. We construct MC questions for 293 problems, excluding those whose solutions rely primarily on direct

computation or basic facts rather than a central theorem or proof strategy. Their domain distribution is shown in Figure 6. An example of direct-answer and multiple-choice questions is shown in Example 1.



A. Abstract Algebra: 28
B. Calculus: 26
C. Combinatorial: 23
D. Complex analysis: 15
E. Fourier Analysis: 31
F. Functional Analysis: 25
G. Geometry: 25
H. Linear Algebra: 20
I. Logic: 27
J. Number Theory: 10
K. Probabilities: 38
L. Real Analysis: 28
M. Topology: 25

Example 1: Diagnostic tasks for a probability problem.

Original problem. Let $S_n = X_1 + \cdots + X_n$ be a random walk, where the X_i are independent and identically distributed with $\mathbb{P}(X_i = 1) = p$, $\mathbb{P}(X_i = -1) = 1 - p$, and $p \neq \frac{1}{2}$. For integers $a \leq -1$ and $b \geq 1$, let

$$\tau = \min\{n \geq 1 : S_n = a \text{ or } S_n = b\}.$$

Show that

$$\mathbb{E}\tau = \frac{a(1 - r^b) + b(r^a - 1)}{(r^a - r^b)(2p - 1)}, \qquad r = \frac{1 - p}{p}.$$

Direct-answer problem. Under the same assumptions, compute $\mathbb{E}\tau$.

Multiple-choice reasoning question. Which result or concept is most useful for solving this problem?

(a) Doob decomposition
(b) Doob's martingale convergence theorems
(c) Optional stopping theorem
(d) Limiting distribution of a Markov chain
(e) Distribution of arrival times for Poisson processes

Figure 2: Left: Distribution of the 321 problems across mathematical domains. Right: An original probability problem with its direct-answer and multiple-choice diagnostic tasks.

Transformed problems. Domain experts manually reformulate selected problems so that they appear substantially different while preserving the same underlying mathematical reasoning. Comparing performance across the original and transformed versions tests whether models reason consistently rather than relying on familiar wording or memorized patterns Mirzadeh et al. (2025).

Example 2 illustrates such a transformation. Although the two formulations appear distinct, they rely on the same mathematical ideas, and recognizing this connection requires a deep understanding of the underlying concepts. In total, we construct transformed versions of 30 problems, with their distribution across domains shown in Figure 7.

Example 2: Transformed questions in real analysis.

Natural language problem. Let $S = \{(0, 0), (2, 0), (0, 1)\}$. Prove that

$$\{\lambda_1(0, 0) + \lambda_2(2, 0) + \lambda_3(0, 1) : \lambda_1, \lambda_2, \lambda_3 \geq 0, \lambda_1 + \lambda_2 + \lambda_3 = 1\} = \{(x, y) \in \mathbb{R}^2 : x \geq 0, y \geq 0, \frac{x}{2} + y \leq 1\}.$$

Transformed problem. Let $S = \{(0, 0), (2, 0), (0, 1)\}$. Prove that the smallest convex set containing S is

$$\{(x, y) \in \mathbb{R}^2 : x \geq 0, y \geq 0, x/2 + y \leq 1\}.$$

Data collection. To provide broad coverage while minimizing redundancy and exposing meaningful model failure modes, we prioritize problems that emphasize distinct aspects of the underlying theory when several candidates involve similar concepts. Contributors also interact with models such as ChatGPT, Gemini, and DeepSeek to assess each problem's diagnostic value. All contributors are PhD students who have completed graduate coursework or conduct research in the relevant domain, and a second expert independently reviews every problem statement and its auxiliary questions. Because the distinction between undergraduate- and graduate-level problems is subjective and varies across institutions and curricula, we do not assign these labels. A complete list of sources is provided in Appendix C.

3.2 Human-in-the-loop autoformalization and challenges

To formalize the natural-language problems in MathAdv, we use the LLM-assisted, human-in-the-loop pipeline illustrated in Figure 3. This design compensates for the inability of current LLMs to independently ensure both valid Lean 4 code and semantic fidelity to the original problems.

Following an interactive, feedback-intensive workflow guided by human experts, the process comprises four stages: *Initial Autoformalization, Syntax Correction, Dual Semantic Verification*, and *Final Expert Review*. An LLM first generates a Lean 4 statement, and compiler errors are returned to the model for iterative correction until the statement type-checks. The resulting formalization is

then checked for semantic fidelity by an independent LLM and a human expert, followed by a final review by a second expert. Further details are provided in Appendix D. This design choice is motivated by two main considerations: the current strengths and limitations of LLMs in mathematical formalization and the distinctive complexity of MathAdv.

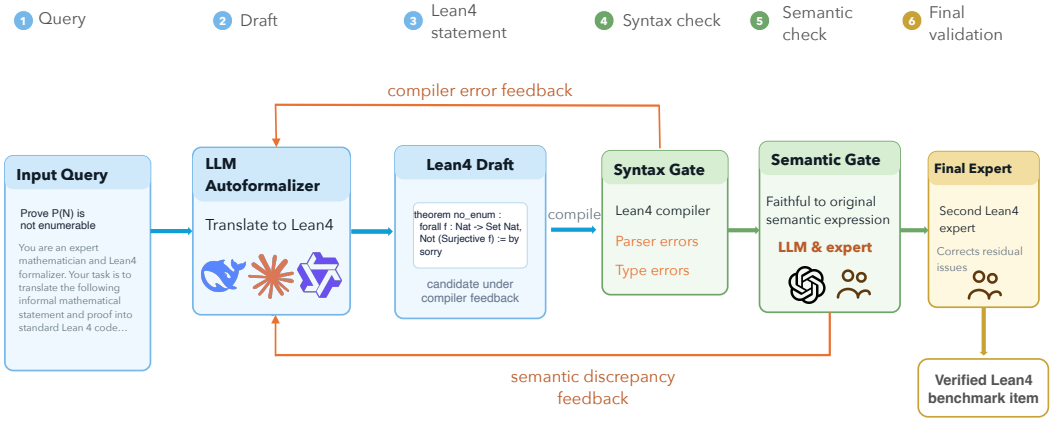


Figure 3: Human-in-the-loop autoformalization process for MathAdv.

First, we observe that general-purpose LLMs can often explain the mathematical meaning of the code and identify semantic issues, but still incorrectly predict that the code does not compile. However, without actually running the Lean compiler, they are much less reliable at judging whether the code will be accepted by the Lean type checker. Using James Hanson's Lean 4 "junk theorems", semantically misleading snippets that nevertheless compile, we test whether models can predict compilation and explain the code's meaning Hanson. The results show that LLMs often misjudge what Lean accepts; full details are provided in Appendix D.1.

This observation motivates the design of our formalization pipeline. We use LLMs for the tasks where they are relatively strong, such as checking whether a formal statement matches the intended mathematical meaning. At the same time, we do not rely on them as syntax or type-checking judges. Instead, we place the Lean verifier in the loop: Lean provides direct compiler feedback on whether the code type-checks, while the LLMs help interpret and revise the formalization at the semantic level.

A second major obstacle to fully automated translation of MathAdv is the limited availability of prerequisite definitions and lemmas in Mathlib at the time of writing. Because MathAdv includes problems from advanced and specialized areas of mathematics, some of the basic objects needed to state these problems have not yet been formalized in Lean by the open-source community. For example, some candidate problems in Riemannian geometry cannot yet be stated or proved in Lean because Mathlib lacks the required specialized definitions and supporting results, such as the Bishop–Gromov inequality. This illustrates how gaps in the current formal library constrain immediate formalization; Appendix D.3 provides a detailed example.

To assess whether problems were immediately formalizable, we systematically surveyed Mathlib using its official documentation and LeanSearch, supplemented by Loggle when the required types were known. Expert judgment remains necessary to identify structural gaps and determine whether a translation is viable. Problems requiring prohibitive upstream library development are retained for auxiliary evaluations and deferred to future iterations of MathAdv as the Lean 4 ecosystem matures.

4 Diagnostic Evaluation of Theorem Provers

We evaluate a diverse set of theorem-proving and general-purpose LLMs on MathAdv. Evaluated models and protocols are described in Section 4.1. Section 4.2 measures end-to-end Lean 4 proof construction. Section 4.3 assesses natural-language problem solving and knowledge of relevant theorems and proof strategies. Section 4.4 evaluates whether models remain robust under mathematically equivalent reformulations. Together, these evaluations provide a fine-grained view of model capabilities, revealing whether failures arise from mathematical reasoning, background knowledge, formal proof construction, or sensitivity to problem formulation.

4.1 SETTINGS

We evaluate two types of proof-generation approaches on MathAdv: proof-step generation models and whole-proof generation models. Proof-step generation based models usually utilize verifier to obtain information on the current tactic state and often times construct valid proofs with tree-searching schemes. For this category, we evaluate MathAdv on InternLM2-Math-Plus-7B Ying et al. (2024), InternLM2-Step-ProverWu et al. (2024), DeepSeek-Prover-V1.5-RL + RMaxTS Xin et al. (2024), DeepSeek-Prover + MA-LoT, Goedel-Prover + MA-LoT Wang et al. (2025), Goedel-Prover-V2-32B Lin et al. (2025b).

Whole-proof generation models in contrast produce an entire proof code given the prompts, without involving communication between prover model and the verifier. The evaluation metrics used for whole-proof generation models are usually pass@K by measuring the number of problems for which at least one valid proof is found among the top K generated attempts. From this category, we evaluated on Goedel-Prover-SFT, Goedel-Prover-DPO Lin et al. (2025a), DeepSeek-Prover-V1.5, DeepSeek-Prover-V1.5-SFT and DeepSeek-Prover-V1.5-RLXin et al. (2024).

In addition to the open-sourced models, we also evaluate MathAdv on GPT-5.4 Singh et al. (2025) Claude-sonnet-5 Anthropic (2026), DeepSeek-V3.2DeepSeek-AI et al. (2025), DeepSeek-R1 Guo et al. (2025). We further evaluate three AI agents with iterative theorem-proving capabilities: Opengauss Math, Inc. (2026) and Codex with GPT-5.4. These agents leverage feedback from the Lean environment to iteratively generate and refine formal proofs. For details on the computational budget used for evaluation of each model, see Appendix E.

4.2 FORMAL THEOREM-PROVING PERFORMANCE

Table 1 presents the performance of theorem-proving LLMs on MathAdv in Lean 4. Overall, theorem-proving-specific training and search improve Lean 4 proving performance, but the overall success rate remains limited on MathAdv.

Model	Accuracy (%)
DeepSeek-Prover-V1.5 Base	1.25
DeepSeek-Prover-V1.5 SFT	9.06
DeepSeek-Prover-V1.5 RL	11.25
DeepSeek-Prover-V1.5 RL + RMaxTS	16.56
InternLM2-Math-Plus-7B	13.12
InternLM2.5-StepProver	10.31
Goedel-Prover-SFT	10.62
Goedel-Prover-DPO	11.25
Goedel-Prover-V2	21.88
DeepSeek-Prover + MA-LoT	10.00
Goedel-Prover + MA-LoT	11.88
GPT-5.4	10.62
DeepSeek-R1	10.94
DeepSeek-V3.2	5.31
Opengauss (GPT-5.4)	8.13
Opengauss (sonnet 5)	25.63
Codex (GPT-5.4)	14.06
Claude Sonnet 5	20.31

Table 1: Performance comparison of theorem prover LLMs on MathAdv in Lean 4.

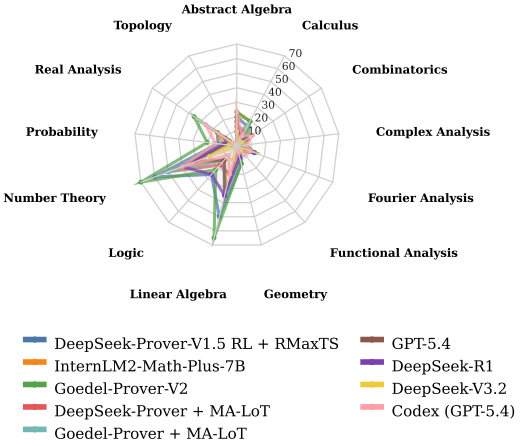


Figure 4: Accuracy by domain in MathAdv for selected models.

Among DeepSeek-Prover-V1.5 variants, accuracy increases from 1.25% for the base model to 11.25% with reinforcement learning, and further to 16.56% with RMaxTS, suggesting the benefit of search-based inference in the sparse-reward setting of formal theorem proving. We also observe smaller gains from the MA-LoT framework across both DeepSeek-Prover and Goedel-Prover. In contrast, general-purpose LLMs such as GPT-5.4 and DeepSeek-R1 achieve only modest accuracy despite their larger scale and broad mathematical knowledge, highlighting the gap between mathematical reasoning and producing successful proofs in Lean 4.

Goedel-Prover-V2 achieves the best performance at 21.88%, followed by DeepSeek-Prover-V1.5 RL + RMaxTS. Notably, both systems employ verifier-guided search, treating theorem proving as

an interactive process with Lean rather than a one-shot generation task. Their strong performance highlights the importance of effective interaction with the proof assistant for end-to-end theorem proving.

Unbalanced performance across different domains.

We also observe substantial variation across mathematical domains, as shown in Figure 4. Goedel-Prover-V2 remains strongest in number theory and linear algebra. Codex (GPT-5.4) shows a different profile: it performs particularly well in number theory, real analysis, and abstract algebra, and obtains the highest accuracy among the evaluated systems in combinatorics and functional analysis. Nevertheless, all evaluated models, including Codex, score 0% on topology. These domain-level differences may reflect both the composition of model training data and the availability of relevant formal library resources. The concentration of Goedel-Prover-V2 and DeepSeek-Prover variants in number theory and linear algebra may be related to their emphasis on formal and competition-style mathematics during training Lin et al. (2025a); Xin et al. (2024). Codex improves the overall accuracy of GPT-5.4, but the improvement is not uniform across domains. This mixed result suggests that agentic execution may be able to help overcome some failures in library use and proof construction, but does not consistently remove domain-specific weaknesses. The performance of each model across all domains can be found in Tables 7-22 in Appendix G.

4.3 Direct-answer and multiple-choice mathematical reasoning performance

We evaluate models on direct-answer (DA) and multiple-choice (MC) problems to isolate natural-language mathematical reasoning and background knowledge in our assessment. Table 2 reports the results. We exclude systems such as MA-LoT that do not support DA or MC answer generation. Details of DA and MC answer generation and evaluation are provided in Appendices F.1 and F.2.

Direct-answer problems. We observe that most models achieve higher accuracy on DA problems than on the corresponding Lean 4 theorem-proving tasks. This gap indicates that models can often solve the underlying mathematical problem, but still struggle to express the solution as a formal proof accepted by Lean. Goedel-Prover is a notable exception: despite its strong theorem-proving performance, it performs comparatively worse on DA tasks, suggesting that its strengths are more closely tied to Lean proof search and theorem application than to general mathematical answer generation in natural language.

Multiple-choice problems. On MC problems, we also see that most models achieve substantially higher accuracy than Lean 4 theorem-proving accuracy. Goedel-Prover remains an exception, consistent with its stronger specialization toward Lean proof search than general answer selection. Together with the DA results, these findings reinforce that formal proof construction involves challenges beyond mathematical background knowledge, natural-language reasoning, and answer generation.

Model	DA (%)	MC (%)
DeepSeek-Prover-V1.5 Base	25.9	53.08
DeepSeek-Prover-V1.5 SFT	32.1	32.53
DeepSeek-Prover-V1.5 RL	37.0	33.90
DeepSeek-Prover-V1.5 RL + RMaxTS	42.0	38.70
InternLM2-Math-Plus-7B	39.5	64.04
Goedel-Prover-SFT	4.9	21.23
Goedel-Prover-DPO	4.9	19.52
DeepSeek-R1	58.0	51.71
DeepSeek-V3.2	66.7	75.00
GPT-5.4	64.2	82.88
Codex (GPT-5.4)	54.2	83.56
Claude Sonnet 5	45.83	84.59

Table 2: Overall accuracy on direct-answer and multiple-choice reasoning questions.

Model	O✓/T×	O×/T✓
DeepSeek-Prover-V1.5 Base	0	0
DeepSeek-Prover-V1.5 SFT	2	2
DeepSeek-Prover-V1.5 RL	4	2
DeepSeek-Prover-V1.5 RL + RMaxTS	4	2
InternLM2-Math-Plus-7B	2	2
InternLM2.5-StepProver	3	2
DeepSeek-Prover + MA-LoT	4	2
Goedel-Prover + MA-LoT	4	2
Goedel-Prover-SFT	4	2
Goedel-Prover-DPO	4	2
Goedel-Prover-V2	6	0
GPT-5.4	2	1
DeepSeek-R1	4	0
DeepSeek-V3.2	3	1
Opengauss (GPT-5.4)	3	1
OpenGauss (Sonnet 5)	7	3
Codex (GPT-5.4)	5	1
Claude Sonnet 5	8	1

Table 3: Original vs. transformed comparison on theorem proving questions. Entries are counts only.

4.4 Robustness to mathematically equivalent transformations

A distinctive feature of MathAdv is its transformed theorem-proving problems in both natural language and Lean, which domain experts manually construct to preserve the underlying mathematical reasoning while changing the problem formulation. Table 3 compares model performance on the original and transformed versions. We use O✓/T× to denote cases where the original version is solved but the transformed version is not, and O×/T✓ to denote the reverse.

The results reveal a consistent robustness gap. For most models, O✓/T× is larger than O×/T✓, meaning that models more often solve the original problem but fail on an equivalent transformed version than vice versa. This effect is especially pronounced for Goedel-Prover-V2 and DeepSeek-R1, which show multiple O✓/T× cases but no O×/T✓ cases. Because the transformations preserve the underlying mathematics, failures on reformulated versions show that current theorem provers depend on how a problem is presented. Changes in wording or structure can disrupt learned proof patterns, theorem retrieval, or proof-search strategies. Although the absolute counts are small, the same asymmetry appears across model families, motivating training that rewards consistency across equivalent formulations rather than reliance on surface cues. Overall, these evaluations show that current models are limited not only by mathematical knowledge, but also by formal proof construction, uneven domain coverage, and sensitivity to equivalent problem formulations.

5 Failure analysis and improvement strategies

In this section, we compare failed and successful proofs to identify what separates plausible attempts from valid Lean proofs in Section 5.1. Section 5.3 examines what agentic interaction can and cannot repair. Finally, Section 5.2 examines how natural-language reasoning hints affect formal proof construction across model types.

5.1 Success requires precise Lean execution

Comparing failed and successful attempts reveals a clear divide. Many failed proofs begin with a reasonable mathematical idea and even resemble valid Lean code, but break because a necessary step is missing, an unsuitable tactic or nonexistent lemma is used, or the proof repeats without progress. Successful proofs avoid these failures by either retrieving the correct Mathlib result or building the argument explicitly, step by step, while tracking the remaining goals. This suggests that improving formal theorem provers requires not only strong mathematical reasoning, but also better control of the proof state, more reliable library retrieval, and the ability to recognize whether each step makes progress. We describe these patterns in more detail below.

Error pattern analysis. We identify four recurring ways in which proofs fail. First, an *incomplete proof* may follow a reasonable approach but prove only a weaker statement, skip a required case, or treat a difficult step as already finished; Lean therefore still has an unresolved claim at the end (see Appendix H.1). Second, a model may choose a proof command that is valid Lean code but cannot establish the current claim. For example, it may use a command intended for linear arithmetic when the proof actually requires reasoning about divisibility (Appendix H.2). Third, models sometimes invent names for results that sound as though they belong to Lean's mathematical library but do not actually exist, preventing the proof from being checked. Finally, some proofs repeat the same rewriting or command many times without changing what remains to be proved (Appendix H.3). These failures show that a plausible overall idea is insufficient unless the model also checks what each step proves and whether it moves the proof forward.

Successful proof patterns. We observe three common ways in which proofs succeed. First, some successful proofs are short because they find an existing result in Lean's mathematical library that closely matches the statement and already captures most of the required argument (Appendix I.1). Second, some proofs translate a step-by-step mathematical argument into Lean by introducing intermediate claims, separating cases, and providing the required objects. Examples *Q4* and *Q5* illustrate this pattern in Appendix I.2. Third, some problems that appear difficult in ordinary mathematical language become simple after their formal definitions are unfolded; the proof then needs only a suitable example and a few basic checks (Appendix I.3). In contrast to failed attempts, successful proofs either find a result that completes the argument or ensure that each explicit step brings the proof closer to completion.

5.2 Model-dependent effects of natural-language hints

We conduct an ablation in which models receive natural-language reasoning hints during Lean 4 proof construction. This experiment is motivated by the results in Section 4.3: most models can identify relevant mathematical knowledge and proof strategies in multiple-choice questions, yet still struggle with end-to-end theorem proving. We therefore test whether providing this knowledge as a hint can help bridge the gap. We evaluate DeepSeek-V3.2, DeepSeek-R1, Goedel-Prover-SFT, and Goedel-Prover-DPO with hints derived from the corresponding multiple-choice questions. We use the same theorem-proving setup, adding a standardized hint based on each question and its correct answer to the prompt.

Hints reveal a clear split between general-purpose and proof-specialized models. They improve the two DeepSeek models but substantially reduce the performance of Goedel-Prover-SFT and Goedel-Prover-DPO. This suggests that natural-language reasoning guidance can help general-purpose models, but can interfere with proof-specialized models whose strengths depend more on learned formal proof patterns and Lean-specific theorem application. Prompting and training strategies should therefore be model-dependent: informal reasoning hints may help general-purpose models, whereas proof-specialized systems may benefit more from formal, Lean-aligned guidance.

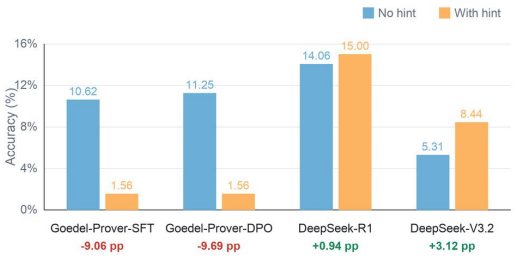


Figure 5: Performance on Lean 4 formal proving with and without hints from MC questions.

5.3 What agentic interaction can and cannot repair

Building on our error analysis and the benefits of interaction observed in Section 4.2, we therefore first conduct a controlled experiment that isolates the effect of verifier feedback. Lean error messages are returned to the model for up to $T = 3$ turns under a fixed budget $K = N \times T$. Table 5 in the appendix shows that feedback improves both evaluated models. Qualitatively, it is most effective at repairing local syntax and elaboration errors. Once these errors are removed, however, failures often shift to unresolved goals, incorrect theorem applications, and type mismatches. Verifier feedback can therefore correct some local implementation mistakes, but does not reliably repair an incomplete or incorrect proof strategy.

Codex and Gauss extend this interaction with library search and file inspection. Rather than revising the original proof directly, an agent may avoid its failure by finding an applicable Mathlib result or constructing a different formalization. Codex finds valid proofs for 21 problems that non-agentic GPT-5.4 fails to solve, while OpenGauss contributes four additional solves beyond the same baseline. Most successful agent proofs are short and direct that introduce no intermediate lemmas. Only seven Codex proofs and four OpenGauss proofs build chains of three or more intermediate results. This suggests that the agents are most effective when the main mathematical idea is already accessible but must be translated into valid Lean, for example by finding an applicable library theorem or choosing a simpler formalization. They are less successful when a problem requires a new multi-step argument.

6 Conclusion and Limitations

In this work, we introduced MathAdv, a comprehensive diagnostic benchmark for formal mathematical reasoning in Lean 4 that spans 13 mathematical domains. By moving beyond aggregate theorem-proving accuracy, it provides a fine-grained view of why models succeed or fail. In addition to formal theorem-proving tasks, MathAdvincludes three auxiliary tasks that allow us to distinguish models' abilities in mathematical problem solving, background knowledge, formal proof construction, and robustness to equivalent reformulations.

MathAdv has two main limitations. First, it remains modest in size because creating and validating high-quality problems across many mathematical domains requires substantial expert effort. Second, the benchmark is constrained by the current coverage of Mathlib, so some advanced problems cannot yet be formalized in Lean 4. We plan to formalize these deferred problems as Mathlib expands.

AI USE STATEMENT

In this work, we used generative AI models both as objects of evaluation and as components of benchmark construction, including candidate autoformalization, semantic checking, and proof generation. We also used generative AI tools for language editing. All AI-assisted benchmark content and formal statements were reviewed by human experts, generated Lean code was checked with the Lean verifier, and the authors reviewed the final manuscript and take responsibility for its content.

REPRODUCIBILITY STATEMENT

We describe the benchmark design and diagnostic tasks in Section 3.1, the human-in-the-loop formalization process in Section 3.2, and the evaluation settings in Section 4.1. The appendix provides the data sources, prompts, computational budgets, evaluation procedures, and additional results needed to reproduce and extend our experiments. The data and evaluation code are included in https://github.com/mathadv912-design/mathadv-iclr.git.

REFERENCES

Anthropic. Introducing claude sonnet 5. https://www.anthropic.com/news/claude-sonnet-5, June 2026. Accessed: 2026-09-24.

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

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023b.

Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development: Coq'Art*. Springer, 2004.

Param Biyani, Shashank Kirtania, Yasharth Bajpai, Sumit Gulwani, and Ashish Tiwari. Indimathbench: Autoformalizing mathematical reasoning problems with a human touch. *arXiv preprint arXiv:2512.00997*, 2025.

Miklós Bóna. *Introduction to Enumerative and Analytic Combinatorics*. CRC Press, 2015.

James Ward Brown and Ruel V. Churchill. *Fourier Series and Boundary Value Problems*. 2011.

James Ward Brown and Ruel V. Churchill. *Complex Variables and Applications*. McGraw-Hill, 9 edition, 2013.

Theo Bühler and Dietmar A. Salamon. *Functional Analysis*. American Mathematical Society, 2018.

George Cain. *Complex Variables*. 2007.

Brandon Coya. Brandon's math blog, 2014. URL https://brandoncoya.wordpress.com/.

Paul Dawkins. Calculus i, 2022. URL https://tutorial.math.lamar.edu/Classes/CalcI/CalcI.aspx.

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, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige

Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Herbert B. Enderton. *A Mathematical Introduction to Logic*. Elsevier, 2001.

Joseph A. Gallian. *Contemporary Abstract Algebra*. Cengage Learning, 10 edition, 2021.

David Gamarnik. Advanced stochastic processes, 2013. Course notes.

Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. Herald: A natural language annotated lean 4 dataset. *arXiv preprint arXiv:2410.10878*, 2024.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating models' local decision boundaries via contrast sets. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1307–1323, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.117. URL https://aclanthology.org/2020.findings-emnlp.117/.

Charles Miller Grinstead and J. Laurie Snell. *Introduction to Probability*. American Mathematical Society, 2006.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin

Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

James Hanson. Junk theorems in lean. https://github.com/James-Hanson/junk-theorems-in-lean. GitHub repository, accessed 2026-06-14.

Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *NeurIPS 2021 Datasets and Benchmarks Track*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Nathan Hoell. Foundations of linear algebra.

Thomas Hubert, Rishi Mehta, Laurent Sartran, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651:607–613, 2026. doi: 10.1038/s41586-025-09833-y. URL https://doi.org/10.1038/s41586-025-09833-y.

David Kammler. *A First Course in Fourier Analysis*. Cambridge University Press, 2 edition, 2007.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Sklgs0NFvr.

Christopher C. Leary and Lars Kristiansen. *A Friendly Introduction to Mathematical Logic*. Milne Open Textbooks, 2 edition, 2015.

Jiří Lebl. *Basic Analysis I & II: Introduction to Real Analysis*. 2024a. URL https://www.jirka.org/ra/.

Jiří Lebl. *Guide to Cultivating Complex Analysis*. 2024b. URL https://www.jirka.org/ca/.

John M. Lee. *Introduction to Smooth Manifolds*. Springer, 2 edition, 2013.

Eric Lehman, F. Thomson Leighton, and Albert R. Meyer. *Mathematics for Computer Science*. Samurai Media Limited, 2017.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Zhaoyu Li, Jialiang Sun, Logan Murphy, Qidong Su, Zenan Li, Xian Zhang, Kaiyu Yang, and Xujie Si. A survey on deep learning for theorem proving, 2024. URL https://arxiv.org/abs/2404.09939.

Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. Lean-star: Learning to interleave thinking and proving. *arXiv preprint arXiv:2407.10040*, 2024.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-prover: A frontier model for open-source automated theorem proving, 2025a. URL https://arxiv.org/abs/2502.07640.

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. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction, 2025b. URL https://arxiv.org/abs/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. Fimo: A challenge formal dataset for automated theorem proving. *arXiv preprint arXiv:2309.04295*, 2023. doi: 10.48550/arXiv.2309.04295. URL https://arxiv.org/abs/2309.04295.

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

László Lovász. *Combinatorial Problems and Exercises*. North-Holland, 1979.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct, 2025. URL https://arxiv.org/abs/2308.09583.

David Marker. *Model Theory: An Introduction*. Springer, 2006.

Math, Inc. Opengauss, 2026. URL https://github.com/math-inc/OpenGauss. GitHub repository.

J. S. Milne. Algebraic number theory, 2020. URL https://www.jmilne.org/math/CourseNotes/ANT.pdf.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AjXkRZIvjB.

MIT OpenCourseWare. 18.06 linear algebra: Assignments, 2010.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings*, pp. 625–635, Berlin, Heidelberg, 2021. Springer-Verlag. ISBN 978-3-030-79875-8. doi: 10.1007/978-3-030-79876-5_37. URL https://doi.org/10.1007/978-3-030-79876-5_37.

Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002.

Peter Petersen. *Riemannian Geometry*. Springer, 2 edition, 2006.

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

Bjorn Poonen. Mit 18.785: Number theory, 2015. URL https://math.mit.edu/~poonen/785.html.

Michael Reed and Barry Simon. *Methods of Modern Mathematical Physics I: Functional Analysis*. Academic Press, 1980.

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. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition, 2025. URL https://arxiv.org/abs/2504.21801.

Martiño Ríos-García, Nawaf Alampara, Chandan Gupta, Indrajeet Mandal, Sajid Mannan, Ali Asghar Aghajani, N. M. Anoop Krishnan, and Kevin Maik Jablonka. Ai scientists produce results without reasoning scientifically. *arXiv preprint arXiv:2604.18805*, 2026.

Halsey L. Royden and Patrick Fitzpatrick. *Real Analysis*. Pearson, 4 edition, 2010.

Walter Rudin. *Functional Analysis*. McGraw-Hill, 2 edition, 1991.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Ziju Shen, Naohao Huang, Fanyi Yang, Yutong Wang, Guoxiong Gao, Tianyi Xu, Jiedong Jiang, Wanyi He, Pu Yang, Mengzhou Sun, et al. Real-prover: Retrieval augmented lean prover for mathematical reasoning. *arXiv preprint arXiv:2505.20613*, 2025.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, Alex Makelov, Alex Neitz, Alex Wei, Alexandra Barr, Alexandre Kirchmeyer, Alexey Ivanov, Alexi Christakis, Alistair Gillespie, Allison Tam, Ally Bennett, Alvin Wan, Alyssa Huang, Amy McDonald Sandjideh, Amy Yang, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrei Gheorghe, Andres Garcia Garcia, Andrew Braunstein, Andrew Liu, Andrew Schmidt, Andrey Mereskin, Andrey Mishchenko, Andy Applebaum, Andy Rogerson, Ann Rajan, Annie Wei, Anoop Kotha, Anubha Srivastava, Anushree Agrawal, Arun Vijayvergiya, Ashley Tyra, Ashvin Nair, Avi Nayak, Ben Eggers, Bessie Ji, Beth Hoover, Bill Chen, Blair Chen, Boaz Barak, Borys Minaiev, Botao Hao, Bowen Baker, Brad Lightcap, Brandon McKinzie, Brandon Wang, Brendan Quinn, Brian Fioca, Brian Hsu, Brian Yang, Brian Yu, Brian Zhang, Brittany Brenner, Callie Riggins Zetino, Cameron Raymond, Camillo Lugaresi, Carolina Paz, Cary Hudson, Cedric Whitney, Chak Li, Charles Chen, Charlotte Cole, Chelsea Voss, Chen Ding, Chen Shen, Chengdu Huang, Chris Colby, Chris Hallacy, Chris Koch, Chris Lu, Christina Kaplan, Christina Kim, CJ Minott-Henriques, Cliff Frey, Cody Yu, Coley Czarnecki, Colin Reid, Colin Wei, Cory Decareaux, Cristina Scheau, Cyril Zhang, Cyrus Forbes, Da Tang, Dakota Goldberg, Dan Roberts, Dana Palmie, Daniel Kappler, Daniel Levine, Daniel Wright, Dave Leo, David Lin, David Robinson, Declan Grabb, Derek Chen, Derek Lim, Derek Salama, Dibya Bhattacharjee, Dimitris Tsipras, Dinghua Li, Dingli Yu, DJ Strouse, Drew Williams, Dylan Hunn, Ed Bayes, Edwin Arbus, Ekin Akyurek, Elaine Ya Le, Elana Widmann, Eli Yani, Elizabeth Proehl, Enis Sert, Enoch Cheung, Eri Schwartz, Eric Han, Eric Jiang, Eric Mitchell, Eric Sigler, Eric Wallace, Erik Ritter, Erin Kavanaugh, Evan Mays, Evgenii Nikishin, Fangyuan Li, Felipe Petroski Such, Filipe de Avila Belbute Peres, Filippo Raso, Florent Bekerman, Foivos Tsimpourlas, Fotis Chantzis, Francis Song, Francis Zhang, Gaby Raila, Garrett McGrath, Gary Briggs, Gary Yang, Giambattista Parascandolo, Gildas Chabot, Grace Kim, Grace Zhao, Gregory Valiant, Guillaume Leclerc, Hadi Salman, Hanson Wang, Hao Sheng, Haoming Jiang, Haoyu Wang, Haozhun Jin, Harshit Sikchi, Heather Schmidt, Henry Aspegren, Honglin Chen, Huida Qiu, Hunter Lightman, Ian Covert, Ian Kivlichan, Ian Silber, Ian Sohl, Ibrahim Hammoud, Ignasi Clavera, Ikai Lan, Ilge Akkaya, Ilya Kostrikov, Irina Kofman, Isak Etinger, Ishaan Singal, Jackie Hehir, Jacob Huh, Jacqueline Pan, Jake Wilczynski, Jakub Pachocki, James Lee, James Quinn, Jamie Kiros, Janvi Kalra, Jasmyn Samaroo, Jason Wang, Jason Wolfe, Jay Chen, Jay Wang, Jean Harb, Jeffrey Han, Jeffrey Wang, Jennifer Zhao, Jeremy Chen, Jerene Yang, Jerry Tworek, Jesse Chand, Jessica Landon, Jessica Liang, Ji Lin, Jiancheng Liu, Jianfeng Wang, Jie Tang, Jihan Yin, Joanne Jang, Joel Morris, Joey Flynn, Johannes Ferstad, Johannes Heidecke, John Fishbein, John Hallman, Jonah Grant, Jonathan Chien, Jonathan Gordon, Jongsoo Park, Jordan Liss, Jos Kraaijeveld,

Joseph Guay, Joseph Mo, Josh Lawson, Josh McGrath, Joshua Vendrow, Joy Jiao, Julian Lee, Julie Steele, Julie Wang, Junhua Mao, Kai Chen, Kai Hayashi, Kai Xiao, Kamyar Salahi, Kan Wu, Karan Sekhri, Karan Sharma, Karan Singhal, Karen Li, Kenny Nguyen, Keren Gu-Lemberg, Kevin King, Kevin Liu, Kevin Stone, Kevin Yu, Kristen Ying, Kristian Georgiev, Kristie Lim, Kushal Tirumala, Kyle Miller, Lama Ahmad, Larry Lv, Laura Clare, Laurance Fauconnet, Lauren Itow, Lauren Yang, Laurentia Romaniuk, Leah Anise, Lee Byron, Leher Pathak, Leon Maksin, Leyan Lo, Leyton Ho, Li Jing, Liang Wu, Liang Xiong, Lien Mamitsuka, Lin Yang, Lindsay McCallum, Lindsey Held, Liz Bourgeois, Logan Engstrom, Lorenz Kuhn, Louis Feuvrier, Lu Zhang, Lucas Switzer, Lukas Kondraciuk, Lukasz Kaiser, Manas Joglekar, Mandeep Singh, Mandip Shah, Manuka Stratta, Marcus Williams, Mark Chen, Mark Sun, Marselus Cayton, Martin Li, Marvin Zhang, Marwan Aljubeh, Matt Nichols, Matthew Haines, Max Schwarzer, Mayank Gupta, Meghan Shah, Melody Huang, Meng Dong, Mengqing Wang, Mia Glaese, Micah Carroll, Michael Lampe, Michael Malek, Michael Sharman, Michael Zhang, Michele Wang, Michelle Pokrass, Mihai Florian, Mikhail Pavlov, Miles Wang, Ming Chen, Mingxuan Wang, Minnia Feng, Mo Bavarian, Molly Lin, Moose Abdool, Mostafa Rohaninejad, Nacho Soto, Natalie Staudacher, Natan LaFontaine, Nathan Marwell, Nelson Liu, Nick Preston, Nick Turley, Nicklas Ansman, Nicole Blades, Nikil Pancha, Nikita Mikhaylin, Niko Felix, Nikunj Handa, Nishant Rai, Nitish Keskar, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Oona Gleeson, Pamela Mishkin, Patryk Lesiewicz, Paul Baltescu, Pavel Belov, Peter Zhokhov, Philip Pronin, Phillip Guo, Phoebe Thacker, Qi Liu, Qiming Yuan, Qinghua Liu, Rachel Dias, Rachel Puckett, Rahul Arora, Ravi Teja Mullapudi, Raz Gaon, Reah Miyara, Rennie Song, Rishabh Aggarwal, RJ Marsan, Robel Yemiru, Robert Xiong, Rohan Kshirsagar, Rohan Nuttall, Roman Tsiupa, Ronen Eldan, Rose Wang, Roshan James, Roy Ziv, Rui Shu, Ruslan Nigmatullin, Saachi Jain, Saam Talaie, Sam Altman, Sam Arnesen, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Sarah Yoo, Savannah Heon, Scott Ethersmith, Sean Grove, Sean Taylor, Sebastien Bubeck, Sever Banesiu, Shaokyi Amdo, Shengjia Zhao, Sherwin Wu, Shibani Santurkar, Shiyu Zhao, Shraman Ray Chaudhuri, Shreyas Krishnaswamy, Shuaiqi, Xia, Shuyang Cheng, Shyamal Anadkat, Simón Posada Fishman, Simon Tobin, Siyuan Fu, Somay Jain, Song Mei, Sonya Egoian, Spencer Kim, Spug Golden, SQ Mah, Steph Lin, Stephen Imm, Steve Sharpe, Steve Yadlowsky, Sulman Choudhry, Sungwon Eum, Suvansh Sanjeev, Tabarak Khan, Tal Stramer, Tao Wang, Tao Xin, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Degry, Thomas Shadwell, Tianfu Fu, Tianshi Gao, Timur Garipov, Tina Sriskandarajah, Toki Sherbakov, Tomer Kaftan, Tomo Hiratsuka, Tongzhou Wang, Tony Song, Tony Zhao, Troy Peterson, Val Kharitonov, Victoria Chernova, Vineet Kosaraju, Vishal Kuo, Vitchyr Pong, Vivek Verma, Vlad Petrov, Wanning Jiang, Weixing Zhang, Wenda Zhou, Wenlei Xie, Wenting Zhan, Wes McCabe, Will DePue, Will Ellsworth, Wulfie Bain, Wyatt Thompson, Xiangning Chen, Xiangyu Qi, Xin Xiang, Xinwei Shi, Yann Dubois, Yaodong Yu, Yara Khakbaz, Yifan Wu, Yilei Qian, Yin Tat Lee, Yinbo Chen, Yizhen Zhang, Yizhong Xiong, Yonglong Tian, Young Cha, Yu Bai, Yu Yang, Yuan Yuan, Yuanzhi Li, Yufeng Zhang, Yuguang Yang, Yujia Jin, Yun Jiang, Yunyun Wang, Yushi Wang, Yutian Liu, Zach Stubenvoll, Zehao Dou, Zheng Wu, and Zhigang Wang. Openai gpt-5 system card, 2025. URL https://arxiv.org/abs/2601.03267.

Elias M. Stein and Rami Shakarchi. *Fourier Analysis: An Introduction*. Princeton University Press, 2003.

William Stein. *Elementary Number Theory: Primes, Congruences, and Secrets*. Springer, 2009.

Gilbert Strang. Calculus online textbook. MIT OpenCourseWare, 2023. URL https://ocw.mit.edu/courses/res-18-001-calculus-fall-2023/pages/about/.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition, 2024. URL https://arxiv.org/abs/2407.11214.

University of California, Berkeley. Eecs16b note 14, 2024.

Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong Zhang. Theoremllama: Transforming general-purpose llms into lean4 experts. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11953–11974, 2024.

Ruida Wang, Rui Pan, Yuxin Li, Jipeng Zhang, Yizhen Jia, Shizhe Diao, Renjie Pi, Junjie Hu, and Tong Zhang. Ma-lot: Model-collaboration lean-based long chain-of-thought reasoning enhances formal theorem proving, 2025. URL https://arxiv.org/abs/2503.03205.

Zijian Wu, Suozhi Huang, Zhejian Zhou, Huaiyuan Ying, Jiayu Wang, Dahua Lin, and Kai Chen. Internlm2.5-stepprover: Advancing automated theorem proving via expert iteration on large-scale lean problems, 2024. URL https://arxiv.org/abs/2410.15700.

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. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search, 2024. URL https://arxiv.org/abs/2408.08152.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, Yudong Wang, Zijian Wu, Shuaibin Li, Fengzhe Zhou, Hongwei Liu, Songyang Zhang, Wenwei Zhang, Hang Yan, Xipeng Qiu, Jiayu Wang, Kai Chen, and Dahua Lin. Internlm-math: Open math large language models toward verifiable reasoning, 2024.

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.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

A Related Work

A.1 Existing Formal Mathematics Benchmarks

miniF2F Zheng et al. (2021) evaluates formal proof generation on competition-level mathematics problems, while ProofNet Azerbayev et al. (2023a) provides paired informal and formal statements to study translation from natural language into a proof assistant. FIMO Liu et al. (2023) and PutnamBench Tsoukalas et al. (2024) are also derived from mathematical competitions and concentrate largely on algebra, number theory, and analysis. FIMO uses IMO Shortlisted Problems from 2006–2021, whereas PutnamBench draws from the William Lowell Putnam Mathematical Competition.

FormalMath Yu et al. (2025) expands the scale and domain coverage of formally verified theorems. FormalNumina Liu et al. (2026) instead emphasizes structured symbolic and numerical reasoning, focusing on precise formal manipulation. IndiMathBench Biyani et al. (2025) extends evaluation to geometry problems involving diagrams and therefore requires both spatial and logical reasoning. Together, these benchmarks provide valuable tests of individual aspects of formal mathematics, but do not jointly diagnose mathematical knowledge, natural-language reasoning, formal proof construction, and robustness to equivalent reformulations.

A.2 Models for Formal Mathematics

One class of methods generates a complete solution or proof in a single pass. GPT-f Polu & Sutskever (2020) was an early application of language models to formal proof generation, while Llemma Azerbayev et al. (2023b) and InternLM Ying et al. (2024) build on stronger pretrained models and mathematical training data. These methods are simple and scalable, but cannot easily recover when an intermediate proof step is incorrect.

Search-based systems instead explore multiple proof paths and use intermediate feedback to revise their choices. Lean-STAR Lin et al. (2024), Real-Prover Shen et al. (2025), and Goedel-Prover Lin et al. (2025a) employ tree search, best-first search, or structured sampling to find valid proofs. This improves their ability to recover from errors, although it requires greater computation.

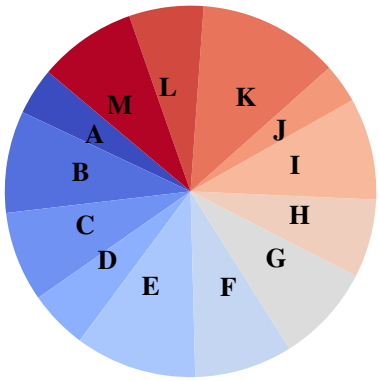
Other systems combine large formal training corpora, proof-assistant feedback, or collaborative reasoning. DeepSeek-Prover Xin et al. (2024) and TheoremLlama Wang et al. (2024) train on extensive formal data; Herald Gao et al. (2024) treats proof construction as a sequence of decisions over changing proof states; and MA-LoT Wang et al. (2025) uses multiple agents or iterative decomposition to address complex problems.

B Data statistics

C Data sources

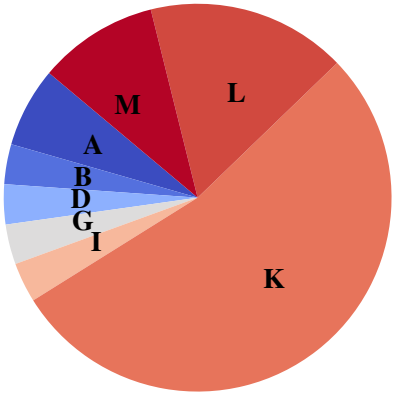
The following is a complete list of sources MathAdvdraws from:

- Probability: Gamarnik (2013); Grinstead & Snell (2006); Lehman et al. (2017)
- Geometry / Topology: Hatcher (2002); Lee (2013); Petersen (2006)
- Abstract Algebra: Gallian (2021); Coya (2014)
- Logic: Enderton (2001); Marker (2006); Leary & Kristiansen (2015)
- Real Analysis: Lebl (2024a); Royden & Fitzpatrick (2010)
- Number Theory: Stein (2009); Milne (2020); Poonen (2015)
- Complex Analysis: Cain (2007); Lebl (2024b); Brown & Churchill (2013)
- Linear Algebra: MIT OpenCourseWare (2010); University of California, Berkeley (2024); Hoell
- Fourier Analysis: Brown & Churchill (2011); Stein & Shakarchi (2003); Kammler (2007)
- Functional Analysis: Rudin (1991); Reed & Simon (1980); Bühler & Salamon (2018)
- Combinatorics: Bóna (2015); Lovász (1979)



A. Abstract Algebra: 12
B. Calculus: 26
C. Combinatorial: 23
D. Complex analysis: 15
E. Fourier Analysis: 31
F. Functional Analysis: 25
G. Geometry: 25
H. Linear Algebra: 20
I. Logic: 26
J. Number Theory: 10
K. Probabilities: 36
L. Real Analysis: 19
M. Topology: 25

Figure 6: Multiple-choice Reasoning Problems by Category (Total = 293).



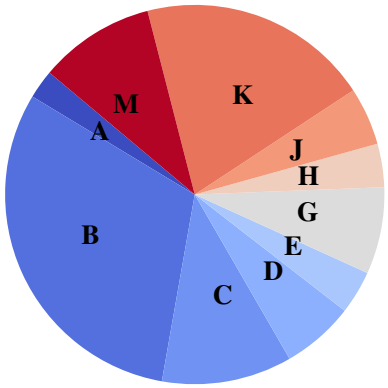
A. Abstract Algebra: 2
B. Calculus: 1
D. Complex analysis: 1
G. Geometry: 1
I. Logic: 1
K. Probabilities: 16
L. Real Analysis: 5
M. Topology: 3

Figure 7: Transformed Problems by Category (Total = 30).

- Calculus: Dawkins (2022); Strang (2023)

D AUTOFORMALIZATION

Initial Autoformalization: For each problem, we employ an LLM-based autoformalizer to translate the natural language problem statement into Lean 4. The prompt configuration utilized for this step is the following: `You are an expert mathematician and Lean 4 formalizer. Your task is to translate the following informal`



A. Abstract Algebra: 2
B. Calculus: 25
C. Combinatorial: 9
D. Complex analysis: 5
E. Fourier Analysis: 3
G. Geometry: 6
H. Linear Algebra: 3
J. Number Theory: 4
K. Probabilities: 16
M. Topology: 8

Figure 8: direct-answer Problems by Category (Total = 81).

```
mathematical statement and proof into standard Lean 4 code.  1.  Ensure
all syntax, imports, and tactics are strictly Lean 4.  Do not output
any Lean3 syntax.  2.  Prioritize existing definitions, classes, and
theorems from Mathlib.  Do not define new terms or structures unless
it is strictly necessary because the concept genuinely does not exist
in the current library.  3.  Before writing the code, explicitly state
which specific Mathlib namespaces or files you are drawing from (e.g.,
Mathlib.ModelTheory.Basic, Mathlib.Combinatorics.SimpleGraph.Basic, or
Mathlib.CategoryTheory.Logic).  4.  Do not formalize the proof.  Leave
the proof blank except for a ``sorry'' statement.  5.  Format your answer
as a valid Lean 4 file ready to be compiled.  The Problem:  [INSERT
THEOREM OR DEFINITION HERE]
```

Syntax Correction: Standalone LLM formalizations remain highly unreliable for generating compilable Lean 4 theorems. To resolve initial syntax errors, the draft code is parsed by the Lean 4 compiler. If compilation fails, the resulting error messages are fed back into the autoformalizer in an iterative loop until the code compiles without syntax errors. The process is monitored by a human expert.

Dual Semantic Verification: To ensure the Lean 4 theorem statement accurately reflects the mathematical intent of the original natural language problem, we conduct semantic checking via a dual mechanism: an independent, secondary LLM evaluator and a human expert. Any semantic discrepancies feedback is routed back to the autoformalizer for correction.

Final Expert Review: Once a statement passes both syntactic and initial semantic filters, an exhaustive final check will be conducted by another human expert before approving the formalization.

D.1 Evaluating LLMs on Lean 4 Junk Theorems

In independent zero-shot conversations, models from multiple providers were presented with these snippets and asked two distinct questions:

1. **Compilation:** Does this self-contained Lean 4 code compile?
2. **Semantics:** What does this code mean, and what theorem is it proving?

A total of 360 calls (180 for compilation, 180 for semantics) were executed across four models: `gpt-5.5`, `gpt-4o`, `deepseek-v4-pro`, and `deepseek-chat`.

D.2 Results and Analysis

Table 4 summarizes the performance of the models across both tasks.

Table 4: Model performance on the Lean 4 Junk Theorems testbed.

Model	Compilation Accuracy (%)	Semantic Understanding (0–2)
`gpt-5.5`	58%	2.00
`deepseek-v4-pro`	4%	1.91
`deepseek-chat`	4%	1.51
`gpt-4o`	18%	0.89

Compilation Prediction: By construction, every snippet in the test corpus successfully compiles in Lean 4. However, the models heavily tended to predict the opposite. The misleading, "nonsense" mathematical reading of the junk theorems consistently overrode the models' ability to trace the actual type-checking logic. Accuracy ranged from a high of 58% for `gpt-5.5` down to just 4% for the DeepSeek models.

Semantic Meaning: The models' explanations of the theorems were hand-graded on a 0–2 scale based on whether the model could effectively "see through" the junk and explain the underlying formal trick (e.g., successfully recognizing the exploitation of junk values like `[].head! = 0` in empty lists).

Here, the model rankings differed drastically from the compilation task. Most notably, `deepseek-v4-pro` demonstrated exceptional semantic understanding (scoring 1.91/2.00) while predicting compilation failures 96% of the time.

The experimental results highlight that understanding the trick of a junk theorem and trusting that it type-checks are fundamentally separate skills for current LLMs. Even when models accurately deduce the formal trick behind a statement, they frequently misjudge the boundary between plausible-looking Lean code and code that actually compiles. This finding motivates our design choice to use a verifier-in-the-loop framework, relying on both LLMs and human experts for semantic alignment while strictly delegating type-checking judgments to the Lean 4 compiler.

D.3 Limitations of Current Mathlib Coverage

Consider the following candidate problem from our dataset, taken from Petersen (2006):

> Let (M, g) be a complete Riemannian manifold of dimension $n \geq 2$. Let vol denote the canonical Riemannian volume measure, and let $B(p, R)$ denote the open geodesic ball of radius R centered at $p \in M$. Let $v(n, k, R)$ denote the volume of a geodesic ball of radius R in the n-dimensional simply connected space form of constant sectional curvature k. Assume the Ricci curvature of M satisfies the uniform lower bound $\mathrm{Ric} \geq (n-1)k$. If there exists a point $p \in M$ and a radius $R > 0$ (with $R \leq \pi/\sqrt{k}$ if $k > 0$) such that $\mathrm{vol}(B(p, R)) = v(n, k, R)$, then the metric g has constant sectional curvature k restricted to the ball $B(p, R)$.

Even stating this theorem in Lean would require formal definitions of sectional curvature, Ricci curvature, geodesic balls, Riemannian volume, and model space forms. While Mathlib contains some foundations for smooth manifolds and Riemannian manifolds, many of the more specialized components of Riemannian geometry needed here are not yet available. Moreover, a standard proof of this result relies on the Bishop–Gromov inequality, a central theorem in comparison geometry that is also not formalized in Mathlib at the time of writing.

E COMPUTATIONAL BUDGETS

The computational budget for proof-step generation based models is usually computed with $K = N \times S \times T$ where N is the number of independent attempts; S is the number of tactics generated each time a node is expanded; and T is the number of the expansion rounds (iterations) per run. InternLM2-Math-Plus and InternLM2.5-Step-Prover both use the same search-style cap ($K = 1 \times 32 \times 100$)—one run per problem, 32 sampled continuations per expansion step, 100 expansion iterations, with early stop on first success. Gödel-Prover SFT and Gödel-Prover DPO each use pass@32. DeepSeek-Prover-V1.5 Base, SFT, and RL each use pass@128 parallel whole-proof sampling; DeepSeek-Prover-V1.5-RL with RMaxTS uses tree search with a default 128 LLM rollouts per problem plus upstream data repeat=16 for scheduling.

The computational budget for whole-proof generation models is represented as pass@K by measuring the number of problems for which at least one valid proof is found among the top K generated attempts. DeepSeek-R1-Distill each use pass@32 on Lean 4. MA-LoT's two backbones share the same paper preset: 16 root proofs, 8 correction samples per proof, search depth 2 (documented as 16 + 8×2). Gödel-Prover-V2-32B uses pass@32 on the initial round plus self-correction: 2 correction rounds with 2 samples each after the initial 32 whole-proof draws. DeepSeek-V3.2 uses pass@32 by default, DeepSeek-R1 uses pass@16. GPT-5.4 uses pass@32 and Claude-sonnet-5 uses pass@16. If applicable, the maximum token allowed per attempt is set to be 8192.

Aristotle, Codex and OpenGauss are agents rather than whole-proof generators, so each gets a single attempt per problem (pass@1), and its budget is limited by time and agent steps instead of a number of sampled proofs. Codex runs GPT-5.4 through the Codex CLI with one call per problem in a read-only sandbox and a 600 s timeout. OpenGauss runs GPT-5.4 with one agent query per problem; it can only read and edit files and is capped at 20 tool-calling turns and 300 s. Aristotle (Harmonic) is a hosted agent and we submit each problem once and wait up to 1800 s, and its internal compute budget is not visible to us.

F EVALUATION PROCESS

F.1 DIRECT-ANSWER QUESTIONS

For answer generation, all models receive the same prompt, which instructs them to solve the problem and provide exactly one final answer enclosed in `boxed...`. Each problem is attempted up to five times using stochastic decoding with $T = 0.7$ and $p = 0.95$, with a maximum of 4096 generated tokens per attempt. Answer extraction and grading follow a two-stage procedure. We first extract the expression enclosed in `boxed...` and check its equivalence to the reference answer using a procedure similar to that of Hendrycks et al. Hendrycks et al. (2021). If this stage fails but a boxed answer is successfully extracted, an LLM judge is used to determine whether the predicted answer is mathematically equivalent to the reference answer.

F.2 MULTIPLE-CHOICE REASONING QUESTIONS

For answer generation, the system prompt instructs the model to answer only the multiple-choice question, using the theorem-proving problem as context, and to return a single option letter enclosed in parentheses (e.g., `(a)`). The theorem-proving problem and the corresponding MC question are concatenated into a single prompt. Each attempt is limited to 2048 generated tokens. Predictions are compared against the reference answers and evaluated using exact-match accuracy.

G ADDITIONAL RESULTS

In this section, we present additional results on model performance on MathAdv. Table 6 reports domain-level accuracy with and without interaction, while Tables 7–22 provide domain-level Lean 4 and multiple-choice results for individual models.

	DeepSeek-V3.2	GPT-5.4
Non-Interactive	5.00%	9.06%
Interactive	7.50%	13.75%
Δ	+2.50 pp	+4.69 pp

Table 5: Overall Lean 4 accuracy with and without interaction.

Table 6: Lean 4 accuracy by domain with and without interaction.

Domain	N	DS Non-Int.	DS Int.	DS Δ	GPT Non-Int.	GPT Int.	GPT Δ
Abstract Algebra	30	10.0%	10.0%	+0.0%	16.7%	23.3%	+6.7%
Calculus	26	3.9%	3.9%	+0.0%	7.7%	7.7%	+0.0%
Combinatorial	23	4.3%	4.3%	+0.0%	8.7%	8.7%	+0.0%
Complex Analysis	15	0.0%	0.0%	+0.0%	0.0%	0.0%	+0.0%
Fourier Analysis	31	9.7%	9.7%	+0.0%	9.7%	12.9%	+3.2%
Functional Analysis	25	4.0%	4.0%	+0.0%	4.0%	4.0%	+0.0%
Geometry	24	0.0%	8.3%	+8.3%	0.0%	8.3%	+8.3%
Linear Algebra	20	10.0%	20.0%	+10.0%	35.0%	35.0%	+0.0%
Logic	23	0.0%	8.7%	+8.7%	4.3%	17.4%	+13.0%
Number Theory	10	20.0%	30.0%	+10.0%	30.0%	40.0%	+10.0%
Probabilities	54	3.7%	3.7%	+0.0%	3.7%	9.3%	+5.6%
Real Analysis	31	3.2%	6.5%	+3.2%	9.7%	19.4%	+9.7%
Topology	8	0.0%	0.0%	+0.0%	0.0%	0.0%	+0.0%

Table 7: Accuracy by domain for DeepSeek V1.5 Base.

Domain	Lean 4 Acc. (%)	MC Acc. (%)
Abstract Algebra	0.0	58.3
Calculus	0.0	48.1
Combinatorial	0.0	65.2
Complex Analysis	0.0	43.8
Fourier Analysis	3.2	54.8
Functional Analysis	0.0	48.0
Geometry	0.0	69.2
Linear Algebra	10.0	45.0
Logic	0.0	44.4
Number Theory	0.0	50.0
Probabilities	0.0	51.9
Real Analysis	3.2	70.0
Topology	0.0	57.1
Overall	1.2	54.3

Table 8: Accuracy by domain for DeepSeek V1.5 SFT.

Domain	Lean 4 Acc. (%)	MC Acc. (%)
Abstract Algebra	16.7	16.7
Calculus	0.0	37.0
Combinatorial	4.3	43.5
Complex Analysis	0.0	37.5
Fourier Analysis	9.7	29.0
Functional Analysis	4.0	32.0
Geometry	8.3	38.5
Linear Algebra	20.0	30.0
Logic	13.0	29.6
Number Theory	30.0	50.0
Probabilities	9.3	32.7
Real Analysis	6.5	35.0
Topology	0.0	21.4
Overall	9.1	32.8

Table 9: Accuracy by domain for DeepSeek V1.5 RL.

Domain	Lean 4 Acc. (%)	MC Acc. (%)
Abstract Algebra	16.7	41.7
Calculus	7.7	18.5
Combinatorial	4.3	52.2
Complex Analysis	0.0	25.0
Fourier Analysis	9.7	22.6
Functional Analysis	4.0	44.0
Geometry	8.3	50.0
Linear Algebra	30.0	50.0
Logic	17.4	33.3
Number Theory	50.0	30.0
Probabilities	11.1	23.1
Real Analysis	3.2	35.0
Topology	0.0	35.7
Overall	11.2	34.1

Table 10: Accuracy by domain for RMaxTS.

Domain	Lean 4 Acc. (%)	MC Acc. (%)
Abstract Algebra	20.0	41.7
Calculus	15.4	29.6
Combinatorial	4.3	30.4
Complex Analysis	0.0	37.5
Fourier Analysis	12.9	48.4
Functional Analysis	4.0	44.0
Geometry	12.5	30.8
Linear Algebra	50.0	40.0
Logic	26.1	33.3
Number Theory	60.0	80.0
Probabilities	13.0	34.6
Real Analysis	16.1	40.0
Topology	0.0	39.3
Overall	16.6	38.5

Table 11: Accuracy by domain for InternLM2-Math.

Domain	Lean 4 Acc. (%)	MC Acc. (%)
Abstract Algebra	23.3	75.0
Calculus	19.2	70.4
Combinatorial	4.3	73.9
Complex Analysis	0.0	68.8
Fourier Analysis	9.7	54.8
Functional Analysis	4.0	60.0
Geometry	8.3	88.5
Linear Algebra	30.0	60.0
Logic	13.0	55.6
Number Theory	40.0	40.0
Probabilities	9.3	42.3
Real Analysis	16.1	90.0
Topology	0.0	64.3
Overall	13.1	63.1

Table 12: Accuracy by domain for InternLM2.5 Step Prover.

Domain	Solved	Total	Lean 4 Acc. (%)
Abstract Algebra	5	30	16.7
Calculus	4	26	15.4
Combinatorial	0	23	0.0
Complex Analysis	0	15	0.0
Fourier Analysis	3	31	9.7
Functional Analysis	1	25	4.0
Geometry	2	24	8.3
Linear Algebra	4	20	20.0
Logic	2	23	8.7
Number Theory	3	10	30.0
Probabilities	6	54	11.1
Real Analysis	3	31	9.7
Topology	0	28	0.0
Overall	33	320	10.3

Table 13: Accuracy by domain for Goedel SFT.

Domain	Lean 4 Acc. (%)	MC Acc. (%)
Abstract Algebra	16.7	16.7
Calculus	11.5	22.2
Combinatorial	4.3	26.1
Complex Analysis	0.0	18.8
Fourier Analysis	9.7	12.9
Functional Analysis	4.0	28.0
Geometry	4.2	26.9
Linear Algebra	30.0	5.0
Logic	13.0	18.5
Number Theory	40.0	10.0
Probabilities	11.1	25.0
Real Analysis	3.2	15.0
Topology	0.0	39.3
Overall	10.6	21.8

Table 14: Accuracy by domain for Goedel DPO.

Domain	Lean 4 Acc. (%)	MC Acc. (%)
Abstract Algebra	10.0	16.7
Calculus	15.4	22.2
Combinatorial	4.3	26.1
Complex Analysis	0.0	12.5
Fourier Analysis	9.7	16.1
Functional Analysis	8.0	28.0
Geometry	8.3	30.8
Linear Algebra	30.0	15.0
Logic	13.0	14.8
Number Theory	40.0	30.0
Probabilities	11.1	11.5
Real Analysis	6.5	15.0
Topology	0.0	17.9
Overall	11.2	18.9

Table 15: Goedel v2 union outcomes aggregated by domain.

Domain	Solved	Total	Lean 4 Acc. (%)
Abstract Algebra	7	30	23.3
Calculus	5	26	19.2
Combinatorial	2	23	8.7
Complex Analysis	1	15	6.7
Fourier Analysis	4	31	12.9
Functional Analysis	1	25	4.0
Geometry	3	24	12.5
Linear Algebra	13	20	65.0
Logic	5	23	21.7
Number Theory	7	10	70.0
Probabilities	11	54	20.4
Real Analysis	11	31	35.5
Topology	0	8	0.0
Overall	70	320	21.9

Table 16: Accuracy by domain for MA-LoT + DeepSeek V1.5.

Domain	Solved	Total	Lean 4 Acc. (%)
Abstract Algebra	4	30	13.3
Calculus	3	26	11.5
Combinatorial	1	23	4.3
Complex Analysis	3	15	20.0
Fourier Analysis	3	31	9.7
Functional Analysis	1	25	4.0
Geometry	8	24	33.3
Linear Algebra	4	20	20.0
Logic	4	23	17.4
Number Theory	5	10	50.0
Probabilities	5	54	9.3
Real Analysis	2	31	6.5
Topology	0	28	0.0
Overall	43	320	13.4

Table 17: Accuracy by domain for MA-LoT + Goedel.

Domain	Solved	Total	Lean 4 Acc. (%)
Abstract Algebra	3	30	10.0
Calculus	4	26	15.4
Combinatorial	1	23	4.3
Complex Analysis	2	15	13.3
Fourier Analysis	4	31	12.9
Functional Analysis	1	25	4.0
Geometry	8	24	33.3
Linear Algebra	7	20	35.0
Logic	3	23	13.0
Number Theory	5	10	50.0
Probabilities	6	54	11.1
Real Analysis	2	31	6.5
Topology	0	28	0.0
Overall	46	320	14.4

Table 18: Accuracy by domain for GPT-5.4.

Domain	Pass@16 (%)	Pass@32 (%)	MC Acc. (%)
Abstract Algebra	16.7	16.7	83.3
Calculus	7.7	7.7	77.8
Combinatorial	8.7	8.7	87.0
Complex Analysis	0.0	0.0	87.5
Fourier Analysis	9.7	9.7	71.0
Functional Analysis	4.0	4.0	84.0
Geometry	0.0	4.2	100.0
Linear Algebra	35.0	35.0	70.0
Logic	4.3	13.0	92.6
Number Theory	30.0	30.0	70.0
Probabilities	3.7	7.4	73.1
Real Analysis	9.7	9.7	90.0
Topology	0.0	0.0	85.7
Overall	9.1	10.6	82.0

Table 19: Accuracy by domain for DeepSeek-R1.

Domain	Lean 4 Pass@16 (%)	MC Acc. (%)
Abstract Algebra	10.0	75.0
Calculus	7.7	55.6
Combinatorial	4.3	65.2
Complex Analysis	26.7	62.5
Fourier Analysis	12.9	45.2
Functional Analysis	4.0	64.0
Geometry	33.3	73.1
Linear Algebra	35.0	20.0
Logic	26.1	55.6
Number Theory	40.0	50.0
Probabilities	11.1	40.4
Real Analysis	0.0	75.0
Topology	0.0	21.4
Overall	14.4	51.7

Table 20: Accuracy by domain for DeepSeek-V3.2.

Domain	Pass@16 (%)	Pass@32 (%)	MC Acc. (%)
Abstract Algebra	10.0	10.0	66.7
Calculus	3.9	3.9	66.7
Combinatorial	4.3	4.3	87.0
Complex Analysis	0.0	0.0	75.0
Fourier Analysis	9.7	9.7	58.1
Functional Analysis	4.0	4.0	80.0
Geometry	0.0	0.0	88.5
Linear Algebra	10.0	10.0	80.0
Logic	0.0	0.0	70.4
Number Theory	20.0	20.0	70.0
Probabilities	3.7	5.6	59.6
Real Analysis	3.2	3.2	95.0
Topology	0.0	0.0	85.7
Overall	5.0	5.3	74.1

Table 21: Lean 4 accuracy by domain for OpenGauss.

Domain	Lean 4 Acc. (%)
Abstract Algebra	10.0
Calculus	3.8
Combinatorics	4.3
Complex Analysis	0.0
Fourier Analysis	9.7
Functional Analysis	4.0
Geometry	8.3
Linear Algebra	25.0
Logic	13.0
Number Theory	20.0
Probability	5.6
Real Analysis	6.5
Topology	0.0
Overall	8.1

Table 22: Lean 4 and multiple-choice accuracy by domain for Codex (GPT-5.4).

Domain	Lean 4 Acc. (%)	MC Acc. (%)
Abstract Algebra	23.3	83.3
Calculus	7.7	81.5
Combinatorics	13.0	87.0
Complex Analysis	6.7	87.5
Fourier Analysis	9.7	71.0
Functional Analysis	8.0	88.0
Geometry	4.2	100.0
Linear Algebra	20.0	65.0
Logic	8.7	88.9
Number Theory	40.0	70.0
Probability	14.8	76.9
Real Analysis	25.8	90.0
Topology	0.0	100.0
Overall	14.1	83.9

H REPRESENTATIVE ERROR EXAMPLES

This appendix provides representative examples of the most common error categories observed during theorem proving. For each category, we include the original problem, a snippet of the generated Lean proof, the verifier feedback, and a brief analysis of the failure.

Table 23: Representative error categories.

Category	Typical Cause
Incorrect theorem application	Goal state mismatch
Incorrect tactic selection	Appropriate strategy, wrong tactic
Hallucinated Mathlib declarations	Non-existent theorem/library names
Parser and syntax errors	Invalid Lean syntax or proof context
Repeated-tactic loops	No progress despite continued generation

H.1 INCORRECT THEOREM APPLICATION

Problem. *Q1: Let x be an element of a cyclic group of order 15. If exactly two of x^3, x^5, and x^9 are equal, prove that order of $x^{13} = 3$.*

Model: GPT-5.4

Generated proof snippet:

```
have h15 : x ^ 15 = 1 := by
  rw [← hG]
  exact pow_card_eq_one x
have hx3 : x ^ 3 = 1 := by
  rcases h with h | h | h
  ·rcases h with ⟨h35, _⟩
    have := congrArg (fun y => y * (x ^ (-3 : ℤ))) h35
    simp [zpow_natCast, mul_assoc] at this
    exact this
...
have horder_dvd : orderOf x ∣3 := by
  exact orderOf_dvd_of_pow_eq_one hx3
have horder_ne_one : orderOf x ≠1 := by
  intro ho
  apply hx_ne_one
  exact orderOf_eq_one_iff.mp ho
have horder_eq : orderOf x = 3 := by
  omega
simpa [horder_eq]
```

Verifier feedback:

```
line 18, column 12-29:
function expected at
  pow_card_eq_one
term has type
  ?m.10070 ^ Fintype.card ?m.10067 = 1

line 15, column 28 to line 18, column 29:
unsolved goals
...
h15 : x ^ 15 = 1
⊢ orderOf (x ^ 13) = 3

line 19, column 4-8:
unexpected token 'have'; expected command
```

Analysis. The model correctly identifies the high-level strategy of proving $x^3 = 1$ and rewriting $x^{13} = x$. However, in the branch $x^3 = x^5$, it incorrectly derives $x^2 = 1$ rather than the required $x^3 = 1$. The resulting proof state no longer matches the intended argument, leaving the final goal unsolved.

H.2 Incorrect Tactic Selection

Problem. *Q1: Let x be an element of a cyclic group of order 15. If exactly two of x^3, x^5, and x^9 are equal, prove that order of $x^{13} = 3$.*

Model. GPT-5.4

```
have horder_dvd : orderOf x ∣3 := by
  exact orderOf_dvd_of_pow_eq_one hx3

have horder_ne_one : orderOf x ≠1 := by
  intro ho
  apply hx_ne_one
  exact orderOf_eq_one_iff.mp ho

have horder_eq : orderOf x = 3 := by
  omega
```

Analysis. The model selects a tactic that is inappropriate for the current goal: the tactic `omega` in the final step. The remaining goal involves divisibility rather than linear arithmetic. Although the overall proof direction is reasonable, the chosen tactic cannot establish the required divisibility property.

H.3 Repeated-Tactic Loops

Problem. *Q1: Let x be an element of a cyclic group of order 15. If exactly two of x^3, x^5, and x^9 are equal, prove that order of $x^{13} = 3$.*

Model. DeepSeek-Prover-V1.5-RL

Generated proof snippet.

```
have h₃ : x ^ 5 = x ^ 9 := by
  apply Eq.symm
  apply Eq.symm
  apply Eq.symm
  apply h₂
have h₄ : orderOf (x ^ 13) = 3 := by
  apply Eq.symm
  apply Eq.symm
  apply Eq.symm
  apply h₃
```

Analysis. The model repeatedly applies the same tactic without making progress toward the goal. Such loops often consume the token budget while leaving the proof state essentially unchanged.

I Representative Success Pattern

I.1 Exact-lemma proofs

Problem. *Q2: A Dedekind algebra is a triple $(N, s : N \to N, o \in N)$ s.t. $o \notin$ range(s), s is injective, and $N = clo_s(o) := \cap\{X : o \in X$ and $(\forall x \in X)s(x) \in X\}$ (i.e. closure of all sets that are closed under s). Prove that for any Dedekind algebra (N, s, o) and any formula $\phi(x)$ (possibly with parameters in N), one has that if $\varphi(o)$ and $(\forall n \in N)(\varphi(n) \to \varphi(s(n)))$, then $(\forall n \in N)\varphi(n)$*

Model. DeepSeek-R1

Generated proof snippet.

```
intro n
have hclosure_set : sClosed A.s A.o {x | phi x} := by
  exact ⟨h0, λx hx => hstep x hx⟩
exact (Set.mem_sInter.mp (A.carrier_is_closure n)) {x | phi x}
    hclosure_set
```

Analysis. The model discharges its goal with `Set.mem_sInter.mp` applied to the closure axiom.

Problem. *Q3: Prove that for any Von Neumann ordinals α, β, it holds that $\alpha \in \beta \vee \beta \in \alpha \vee \alpha = \beta$*

Model. DeepSeek-R1

Generated proof snippet.

```
intro h
rcases h with ⟨f, hsurj⟩
let D : Set ℕ:= {n | n ∉f n}
have hD : ∃m, f m = D := hsurj D
rcases hD with ⟨m, hm⟩
have hmem : m ∈D ↔m ∉f m := by
  simp [D]
have hmem' : m ∈D ↔m ∉D := by
  rw [hm] at hmem
  exact hmem
by_cases h : m ∈D
·exact (hmem'.mp h) h
·exact h (hmem'.mpr h)
```

Analysis. The model reduces ordinal trichotomy to a single call of `lt_trichotomy` followed by a case split.

I.2 HAVE-HEAVY PROOFS

Problem. *Q4: Prove that $\mathcal{P}(\mathbb{N})$, the power set of $\mathbb{N}$, is not enumerable*

Model. DeepSeek-R1

Generated proof snippet.

```
intro h
rcases h with ⟨U, hUcomp, hU⟩
have hNone : Nat.Partrec (fun x : ℕ=> Part.none) := Nat.Partrec.none
rcases hEnum.2 (fun x : ℕ=> Part.none) hNone with ⟨i, hi⟩
have htot : ∀x, phi i x = Part.some (U i x) := by
  intro x; exact hU i x
have hnone : ∀x, phi i x = Part.none := by
  intro x; simpa [hi]
have hcontra : Part.some (U i 0) = Part.none := by
  calc
    Part.some (U i 0) = phi i 0 := by symm; exact htot 0
    _ = Part.none := hnone 0
exact (Part.some_ne_none (U i 0)) hcontra
```

Problem. *Q5: Assume the set of partial recursive unary functions is enumerable; that is, there is an enumeration $\{\varphi_i(x)\}_{i\in\omega}$ of partial recursive unary functions. Prove that there is no universal computable function; that is, the universal function $U : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, U(k, x) := \phi_k(x)$ is not both total and computable*

Model. DeepSeek-R1

Generated proof snippet.

```
intro n
have hclosure_set : sClosed A.s A.o {x | phi x} := by
  exact ⟨h0, λx hx => hstep x hx⟩
exact (Set.mem_sInter.mp (A.carrier_is_closure n)) {x | phi x}
    hclosure_set
```

I.3 HAVE-HEAVY PROOFS

Problem. *Q6: Let $\mathcal{L} = \{c, f, A\}$ be a first order language with constant symbol c, a one-place (i.e. unary) function symbol f, and a two-place predicate (i.e. binary relation) symbol A. Let the structure $\mathfrak{M}$ be defined by:*

- $|\mathfrak{M}| = \{1, 2, 3\}$
- $c^{\mathfrak{M}} = 3$
- $f^{\mathfrak{M}}(1) = 2, f^{\mathfrak{M}}(2) = 3, f^{\mathfrak{M}}(3) = 3$
- $A^{\mathfrak{M}} = \{(1, 2), (2, 3), (3, 3)\}$

Let $s(v) = 1$ for all variables v. For arbitrary variable v, a v-variant *of s is an assignment $s[m/v]$ that disagrees with s at most by assigning a different value m to v. Show that $\mathfrak{M}, s \models \exists x\Big(A(f(z), c) \to \forall y\big(A(y, x) \vee A(f(y), x)\big)\Big)$*

Model. DeepSeek-R1

Generated proof snippet.

```
refine ⟨M.three, λh => ?_⟩
intro y
cases y
·right; exact True.intro
·left; exact True.intro
·left; exact True.intro
```

J MULTI-TURN PROMPT RESULTS EXAMPLE

J.1 MORE HAVE PROOFS

Problem. *Q7: Suppose 120 students take a final exam and the mean of their scores is 90. You have no other information about the students and the exam, that is, you should not assume that the highest possible score is 100. You may, however, assume that exam scores are nonnegative. Show that the best possible upper bound on the number of students who scored at least 180 is 60.*

Model. GPT-5.4

Successful proof snippet under interaction.

```
theorem problem_16
  (scores : Fin 120 →ℝ)
  (h_nonneg : ∀i, 0 ≤scores i)
  (h_mean : (∑ i, scores i) / 120 = 90) :
  (Finset.univ.filter (fun i => 180 ≤scores i)).card ≤60 := by
  let S : Finset (Fin 120) := Finset.univ.filter (fun i => 180 ≤scores i)
  have hsum : ∑i, scores i = 10800 := by
    have h := h_mean
    norm_num at h
    linarith
  have hS_le_sum : (S.card : ℝ) * 180 ≤∑i in S, scores i := by
    calc
      (S.card : ℝ) * 180 = ∑i in S, 180 := by rw [Finset.sum_const,
          nsmul_eq_mul]
      _ ≤∑i in S, scores i := by
        refine Finset.sum_le_sum ?_
        intro i hi
        exact Finset.mem_filter.mp hi |>.2
  have hsub_le_total : ∑i in S, scores i ≤∑i, scores i := by
    exact Finset.sum_le_univ_sum_of_nonneg (s := S) (f := scores) (by
        intro i; exact h_nonneg i)
  have hcard_le : (S.card : ℝ) * 180 ≤10800 := by
    linarith
```

```
  have hcard_real : (S.card : ℝ) ≤60 := by
    linarith
  exact_mod_cast hcard_real
```

Failed proof snippet without interaction.

```
theorem problem_16
  (scores : Fin 120 →ℝ)
  (h_nonneg : ∀i, 0 ≤scores i)
  (h_mean : (∑ i, scores i) / 120 = 90) :
  (Finset.univ.filter (fun i => 180 ≤scores i)).card ≤60 := by
  let S : Finset (Fin 120) := Finset.univ.filter (fun i => 180 ≤scores i)
    have hsum : ∑i, scores i = 10800 := by
      have h := h_mean
      norm_num at h ⊢
      linarith
    have h_le_sumS : 180 * S.card ≤∑i in S, scores i := by
      calc
        180 * S.card = ∑i in S, 180 := by rw [Finset.sum_const,
            nsmul_eq_mul]
        _ ≤∑i in S, scores i := by
          refine Finset.sum_le_sum ?_
          intro i hi
          exact le_of_lt?
```

Analysis. The model introduces intermediate `have` statements that are never used in the final proof, a pattern observed less frequently in the non-interactive setting.

J.2 Manual longer proofs

Problem. *Q8: Let* $\{X_i\}_{i\geq 1}$ *be i.i.d. and* $\mathbb{E}X_1 > 0$*. Given* $a > 0$*, show that* $\mathbb{E}\tau < \infty$ *for* $\tau = \inf\{k \geq 1 : S_k > a\}$*, where* $S_k = \sum_{i=1}^{k} X_i$*.*

Model. GPT-5.4

Successful proof snippet under interaction.

```
theorem bona_1(n : ℕ) :
   Fintype.card { f : Fin n →Fin n // Function.Bijective f } = Nat.
       factorial n := by
  classical
  let e :
     { f : Fin n →Fin n // Function.Bijective f } ≃Equiv.Perm (Fin n) :=
    { toFun := fun f =>
       Equiv.ofBijective f.1 f.2
      invFun := fun σ=>
       ⟨σ, σ.bijective⟩
      left_inv := by
       intro f
       apply Subtype.ext
       funext x
       rfl
      right_inv := by
       intro σ
       ext x
       rfl }
  calc
    Fintype.card { f : Fin n →Fin n // Function.Bijective f }
       = Fintype.card (Equiv.Perm (Fin n)) := Fintype.card_congr e
    _ = Nat.factorial (Fintype.card (Fin n)) := Fintype.card_perm
    _ = Nat.factorial n := by simp
```

Failed proof snippet without interaction.

```
theorem bona_1(n : ℕ) :
```

```
    Fintype.card { f : Fin n →Fin n // Function.Bijective f } = Nat.
        factorial n := by
  simpa using Fintype.card_perm n
```

Analysis. The proof under interactive mode transitions from `simpa using Fintype.card_perm n` to an explicit `Equiv.Perm` construction through a `calc` chain.

Problem. *Q9: Prove that for any Von Neumann ordinals α, β, it holds that $\alpha \in \beta \vee$*

Model. GPT-5.4

Successful proof snippet under interaction.

```
theorem open_logic_9 (α β: Ordinal) :
  α< β∨β< α∨α= β:= by
  by_cases h : α= β
  ·exact Or.inr <| Or.inr h
  ·have hne : α≠β:= h
    rcases lt_or_gt_of_ne hne with hlt | hgt
    ·exact Or.inl hlt
    ·exact Or.inr <| Or.inl hgt
```

Failed proof snippet without interaction.

```
theorem open_logic_9 (α β: Ordinal) :
  α< β∨β< α∨α= β:= by
  simpa [lt_or_gt_of_ne, eq_comm] using lt_trichotomy αβ
```

Analysis. Under the interactive mode, the model replaces a `simpa`-with-`lt_trichotomy` strategy with an explicit `by_cases` argument using `lt_or_gt_of_ne`.