

TAOANALYSISBENCH: Do Automated Theorem Prover LLMs Generalize Beyond MathLib?

Alexander K Taylor^{1*}, Junyi Zhang^{1*}, Ethan Ji¹, Vigyan Sahai²,
Haikang Deng¹, Yuanzhou Chen¹, Yifan Yuan¹, Di Wu¹, Jia-Chen Gu¹
Kai-Wei Chang^{1†}, Nanyun Peng^{1†}, Amit Sahai^{1†}, Wei Wang^{1†}

¹University of California, Los Angeles ²University of California, Santa Barbara
ataylor2@g.ucla.edu

Abstract

Automated theorem proving (ATP) benchmarks largely consist of problems formalized in MathLib, so current ATP training and evaluation are heavily biased toward MathLib’s definitional framework. However, frontier mathematics is often exploratory and prototype-heavy, relying on bespoke constructions that deviate from standard libraries. In this work, we evaluate the robustness of current ATP systems when applied to a novel definitional framework. We introduce TAOANALYSISBENCH, an undergraduate-level benchmark derived from the exercises and definitional frameworks given in Terence Tao’s *Analysis I*, which formalizes analysis by constructing core mathematical concepts from scratch, without relying on standard Mathlib definitions, as well as by mixing from-scratch and MathLib constructions. For fair evaluation, we build an agentic pipeline that automatically extracts a compilable, self-contained local environment for each problem. To isolate the effect of definitional frameworks, we additionally translate every problem into a mathematically equivalent Mathlib formulation, yielding paired TAOANALYSISBENCH-MATHLIB statements for direct comparison. While state-of-the-art ATP models perform capably within the MathLib framework, performance drops by an average of 2.18× on the definitionally equivalent Tao formulation. This indicates that the main bottleneck is limited generalization across definitional frameworks rather than task difficulty. TAOANALYSISBENCH thus highlights a gap between benchmark performance and applicability, and provides a concrete foundation for developing and testing provers better aligned with research mathematics.

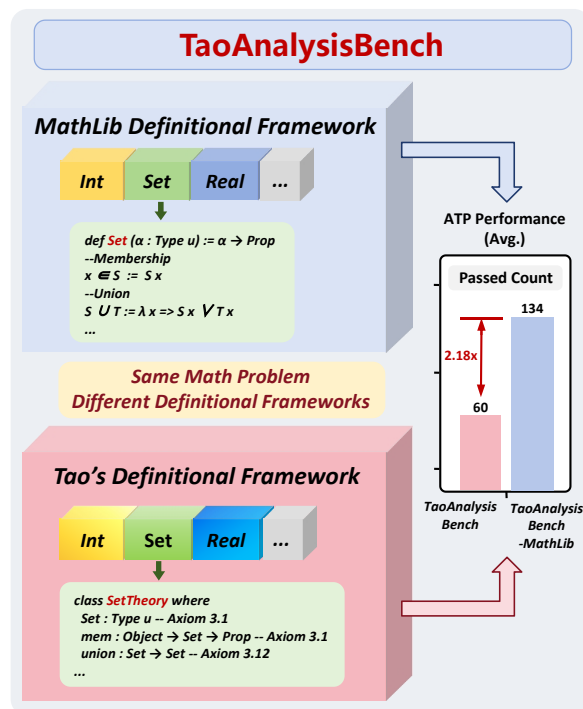


Figure 1: Tao’s *Analysis I* constructs several mathematical structures in LEAN from scratch, often using formulations that diverge significantly from their equivalents in MathLib. TAOANALYSISBENCH provides mathematically equivalent MathLib counterparts, and reveals a substantial degradation in model performance when navigating Taos specific definitional framework.

1 Introduction

Applying large language reasoning models (LLMs) to formally verifiable programming languages like Lean 4 has emerged as a critical frontier for advancing the mathematical reasoning abilities of AI. Accordingly, several recent works have focused on training specialized automated theorem proving (ATP) LLMs to reason in Lean and have achieved high solve rates on formalized benchmarks, such as MiniF2F (Lin et al., 2025c;

*Equal contribution.

†Senior authors served as equal advisors and are listed alphabetically.

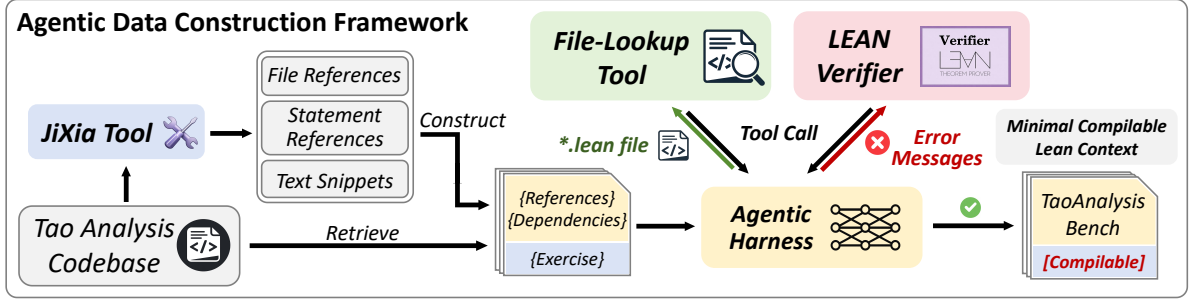


Figure 2: Agentic framework for automatically extracting a compilable, self-contained local environment from a formalized textbook. The construction of references and dependencies for each exercise is performed using the JiXia tool. An agentic harness, equipped with a file-lookup tool and a Lean verifier, then iteratively builds a compilable, self-contained local environment.

Ren et al., 2025; Wang et al., 2025; Zheng et al., 2022).

Current Lean datasets for training and evaluating ATP models (Wang et al., 2025; Xin et al., 2024; Ren et al., 2025; Lin et al., 2025b,c; Xin et al., 2025) are built almost entirely within the definitional framework of MathLib (Zheng et al., 2022; Tsoukalas et al., 2024; Azerbayev et al., 2023; AI-MO, 2025; Gao et al., 2025), the largest and de facto standard mathematical library for Lean 4 (Community, 2025). However, exploratory mathematical research often ventures beyond the boundaries of MathLib and requires problem-specific definitions and constructions to prototype concepts and guide proofs (Tao, 2023; Degenne et al., 2025; Harbor-Collins and Hulse, 2025). This introduces a distribution shift between the definitional framework used in public SOTA ATP model training and the frameworks to which they are applied. As such, current benchmarks offer limited insight into whether ATP models can generalize to novel definitional frameworks.

We introduce TAOANALYSISBENCH, the first benchmark designed to measure ATP robustness beyond MathLib by targeting generalization to a novel definitional framework. TAOANALYSISBENCH is grounded in 600 exercises from Terence Tao’s Lean formalization of Analysis I (Tao, 2025a). This formalization develops core concepts of analysis from first principles rather than relying on standard MathLib definitions, and in some cases combines from-scratch constructions with MathLib components.

We develop a series of agentic pipelines to package each problem with a self-contained, compilable local environment extracted from the formalized textbook. We also construct a paired control: for each Tao-style problem, we construct a mathe-

matically equivalent MathLib formulation, which we title TAOANALYSISBENCH-MATHLIB. This paired representation isolates the effect of definitional frameworks from the difficulty of each problem, enabling a direct evaluation of impact on model performance.

We evaluate several state-of-the-art (SOTA) ATP models and two frontier models on this paired evaluation. We find that performance on TAOANALYSISBENCH drops by an average $2.18\times$ for SOTA models and $1.94\times$ for frontier models on the same problems in TAOANALYSISBENCH-MATHLIB. The observed gap implies that training in the MathLib definitional framework does not reliably transfer to unfamiliar definitional formalizations, even when the underlying mathematics is equivalent. We believe TAOANALYSISBENCH exposes a key hurdle towards applying public SOTA ATP models to research mathematics, and provides a concrete testbed for aligning future prover models with real-world mathematical workflows.

Our main contributions are as follows: (1) We propose TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB, which together constitute the first Lean benchmark targeting generalization beyond MathLib via a new definitional framework using undergraduate-level Analysis. (2) We point out the limitations of small specialized ATP models: they do not generalize well to bespoke definitions and are substantially less capable than agentic frameworks. (3) We introduce a series of pipelines that automatically extract self-contained, compilable local environments for problems from a large, formalized project and provide equivalent formalizations in MathLib syntax, which enables scalable, high-fidelity generation of mathematically grounded training data for future models.

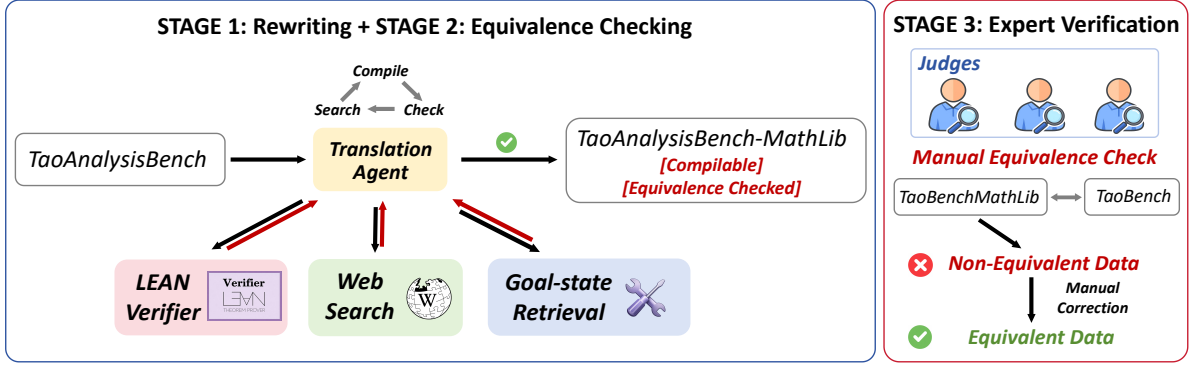


Figure 3: Translation pipeline for converting exercises from Tao’s definitional framework to MathLib’s. Rewriting and equivalence checking ensure that TAOANALYSISBENCH-MATHLIB is compilable and mathematically equivalent, followed by expert verification for additional assurance.

2 Benchmark Construction

Research in mathematics often requires introducing novel definitions and constructs that are not yet in formalized Lean libraries. Characterizing the efficacy of public ATP models in unfamiliar Lean environments requires formalized definitional frameworks distinct from MathLib, but in a mathematically familiar setting. We specify the second criterion to control for mathematical difficulty: for sufficiently difficult mathematics, a model may know a result exists but not how to prove it. In practice, much formalization is done post hoc, and hence we expect models to have access to the latter in natural language.

Tao’s Lean formalization of *Analysis I* defines many standard mathematical objects using custom inductive types or structures for pedagogical clarity, adopts naming schemes that differ from MathLib conventions, and organizes definitions within section- and subsection-level namespaces. Additionally, some concepts are defined as fields of larger structures rather than as standalone definitions. As a result, even mathematically elementary statements may become out-of-distribution to models trained primarily on MathLib-style Lean code. See Appendix A.7 for a detailed breakdown of the overlap between the MathLib and Analysis I definitional frameworks.

To compare model capabilities on the two definitional frameworks, we construct twin versions of the problem set used in the benchmark: TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB. TAOANALYSISBENCH consists of formalized textbook problems and provides all relevant definitions and references necessary to correctly state the problem in a fully compilable, in-context environment. This setting is designed to

assess how well models function on non-MathLib formalizations when contextual information is provided. TAOANALYSISBENCH-MATHLIB consists of MathLib-syntax formulations of the same exercises, enabling an in-distribution evaluation that isolates mathematical difficulty from representational mismatch.

2.1 TAOANALYSISBENCH

2.1.1 Minimal Compilable Environment

Constructing a benchmark that preserves the exact textbook formalizations while producing a self-contained compilable Lean snippet presents significant technical challenges for current LLMs. We construct an agentic pipeline as well as custom tools to automate the construction of the TAOANALYSISBENCH problem set. As shown in Figure 2, our data aggregation pipeline consists of three stages: retrieval of constructions required for compilation for each problem, construction of the compilable environment, and compilation verification.

The first stage is to retrieve constructions referenced in the formalization of the exercise. Individual textbook exercises depend on a nontrivial subset of previously defined formalized theorems, lemmas, definitions, and notation, often spread across multiple subsections. We approach this by first retrieving only the definitions strictly required to faithfully compile each exercise. We use the JiXia static Lean analysis tool (frenzymath, 2024) to recursively retrieve all references of a benchmark problem.¹ These references are provided in the context-construction prompt for each problem, along with the file of origin to ground the context

¹We extend JiXia to support retrieval of structure declarations in addition to constants and theorems.

Model	TAOANALYSISBENCH		TAOANALYSISBENCH-MATHLIB		
	Pass@128	(%)	Pass@128	(%)	Ratio
Kimina-Prover-8B	30/600	5.00	40/600	6.70	1.33×
Goedel-Prover-V2-8B	31/600	5.20	86/600	14.30	2.77×
DeepSeek-Prover-V2-7B	72/600	12.00	146/600	24.30	2.03×
Goedel-Prover-V2-32B	<u>41/600</u>	<u>6.83</u>	<u>88/600</u>	<u>14.67</u>	2.15×
	Pass@32	(%)	Pass@32	(%)	Ratio
Goedel-Prover-V2-8B (w/ Self-Correction)	<u>81/600</u>	<u>13.50</u>	<u>198/600</u>	<u>33.33</u>	2.47×
Goedel-Prover-V2-32B (w/ Self-Correction)	108/600	18.00	249/600	41.50	2.31×

Table 1: Model performance on TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB. Experiments here use FP8. Results from experiments using FP16 can be found in Table 10.

construction to a fixed set of definitions. This mitigates model tendencies to hallucinate or reformulate statements.

The second stage is constructing the compilable environment. We introduce a file-lookup tool into the agent that allows the model to retrieve source files, enabling access to the local structure required to faithfully reconstruct compilable Lean environments for each problem.

The third stage is compilation verification. We introduce a Lean-Compilation tool that provides the model with a local Lean environment in which to test the compilation of the generated context for each benchmark problem. The tool returns either a success message or the stack trace in the event of failed compilation attempts to improve the agent’s ability to debug failures.

2.1.2 Analysis I-Tagged and Enhanced Compilable Environment

To allow for a fair comparison with the MathLib definitional framework, we enhance the minimal compilable environment provided in the previous stage. We provide an annotation of the *Analysis I* lean companion with tagging for search tactics. Incorporating this tagging into the verification environment allows the model to use tagged statements tagged to advance the proof state. We also re-introduce statements from the parent module of each problem to provide additional context. To this end, we construct an agentic pipeline, as shown in Figure 5, that consists of two stages: textbook tagging and environment enrichment.

The first stage is annotating textbook modules with tactic tags. We nest an agentic pipeline within

the larger pipeline that itself consists of two agents: an annotator agent and a verifier agent. The annotator agent is provided a build-validate-strip tool that iteratively builds the project and strips breaking tag attributes to ensure that the module compiles cleanly. The verifier agent provides a review of the annotations for each module and provides feedback to the annotation agent.

The second stage is enriching the minimal compilable environment. To allow access to the tagged version of the textbook constructed in the first stage, we use downstream agents to add annotated textbook module imports and statements from the exercise’s parent module to the minimal compilable environments. The resulting context consists of imports of all relevant modules, de-duplicated statements from the parent module, and the minimal compilable environment, and the exercise. Similar to prior pipelines, we provide the enrichment agent with a lean verification tool to verify environment compilation. Lastly, a reviewer agent provides a pass to verify that statements have been copied faithfully and do not introduce trivial closures to the new environment.

The result of this pipeline is a tagged version of the *Analysis I* lean companion and the final version of TAOANALYSISBENCH environments that together allow search tactics to function in generated proofs in the novel definitional framework.

2.2 TAOANALYSISBENCH-MATHLIB

To isolate the effect of the mathematical difficulty of the problems, we translate from TAOANALYSISBENCH statements, written in Tao’s definitional framework with in-context local definitions,

Model	TAOANALYSISBENCH		TAOANALYSISBENCH-MATHLIB		
	Pass@4	(%)	Pass@4	(%)	Ratio
GPT-5.5 (OpenAI, 2026)	123/600	20.5	216/600	36.0	1.76×
Claude-4.7-Opus (Anthropic, 2026)	85/600	14.2	183/600	30.5	2.15×
	Pass@1	(%)	Pass@1	(%)	Ratio
Agentic prover (GPT-5.5)	526/600	87.7	575/600	95.8	1.09×

Table 2: Model Performance on TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB for frontier foundation models.

Model	Minimal-Compilable		Enhanced		Ratio
	Pass@128	(%)	Pass@128	(%)	
Kimina-Prover-8B	5/600	5.00	30/600	5.0	6.25×
Goedel-Prover-V2-8B	28/600	4.70	31/600	5.20	1.11×
DeepSeek-Prover-V2-7B	57/600	9.50	72/600	12.0	1.26×
Goedel-Prover-V2-32B	<u>38/600</u>	<u>6.30</u>	<u>41/600</u>	<u>6.83</u>	1.16×

Table 3: Minimal-Compilable Environment v. Enhanced Environment.

into statements that use standard MathLib definitions, which we title TAOANALYSISBENCH-MATHLIB. We build an agentic pipeline to perform this reformulation at scale shown in Figure 3, followed by expert manual verification of the resulting MathLib statements to ensure mathematical equivalence with their Tao counterparts. Our pipeline consists of two stages: reformulation of the TAOANALYSISBENCH exercise environment to the MathLib definitional framework and equivalence verification.

The first stage is the reformulation of the TAOANALYSISBENCH exercises. To enable the model to ground its translation in MathLib’s canonical definitions and usage conventions, we enable web-search to allow the model to retrieve authoritative documentation and examples for core MathLib constructs. We also explicitly restrict the model from introducing new local definitions or notation and require adherence to a fixed MathLib header, ensuring consistency across translated problems. The agent then verifies that the resulting reformulation compiles using the Lean compiler tool.

The second stage is equivalence verification. We provide the agent with a JiXia-based goal-state extraction tool, which extracts the target proof states for both the Tao and MathLib formulations and grounds the comparison of their respective proof obligations. The agent assesses whether the two formulations are mathematically equivalent, given their statements and target states.

The final output of this pipeline is TAOANALYSISBENCH-MATHLIB, a MathLib-based reformulation of TAOANALYSISBENCH that preserves the mathematical intent of the original textbook exercises while remaining fully compatible with MathLibs definitional framework.

2.2.1 Expert Manual Verification

To verify the quality of both versions of the benchmark, we had expert human annotators provide determinations on whether the TAOANALYSISBENCH faithfully retrieved the context for each exercise, and whether each MathLib formalization was mathematically equivalent to its Tao counterpart. Each annotator had completed an equivalent *Analysis I* course and had experience in formal proving in Lean. Issues identified by annotators in context retrieval helped us construct a final agentic context retrieval pipeline that produces correct context for 150 problems in TAOANALYSISBENCH. Human annotators identified 33 often subtle issues with the MathLib translations and provided corrected versions of the problems. These grades were then used as in-context examples to a strong reasoning LLM, which provided the grades for the rest of the 450 examples. The full protocol for this is provided in Appendix A.5.

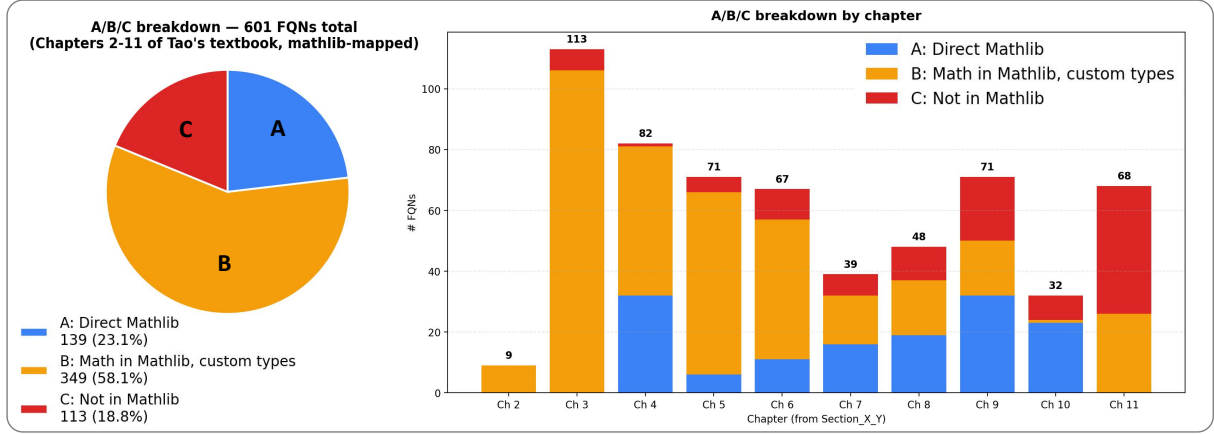


Figure 4: MathLib Coverage Distribution for TAOANALYSISBENCH. Left: overall distribution by MathLib coverage category. Right: chapter-wise coverage breakdown.

Model	A	B	C	Overall
	Pass@128(%)	Pass@128(%)	Pass@128(%)	Pass@128(%)
TAOANALYSISBENCH				
Kimina-Prover-8B	3/139 (2.2%)	2/349 (0.6%)	0/112 (0.0%)	5/600 (0.8%)
Goedel-Prover-V2-8B	24/139 (17.3%)	4/349 (1.1%)	0/112 (0.0%)	28/600 (4.7%)
DeepSeek-Prover-V2-7B	49/139 (35.3%)	8/349 (2.3%)	0/112 (0.0%)	57/600 (9.5%)
Goedel-Prover-V2-32B	32/139 (23.0%)	10/349 (2.9%)	0/112 (0.0%)	42/600 (7.0%)
GD-8B (w/ Self-Correction)	58/139 (41.7%)	21/349 (6.0%)	2/112 (1.8%)	81/600 (13.5%)
GD-32B (w/ Self-Correction)	71/139 (51.1%)	29/349 (8.3%)	8/112 (7.1%)	108/600 (18.0%)
TAOANALYSISBENCH-MATHLIB				
Kimina-Prover-8B	7/139 (5.0%)	32/349 (9.2%)	1/112 (0.9%)	40/600 (6.7%)
Goedel-Prover-V2-8B	26/139 (18.7%)	58/349 (16.6%)	2/112 (1.8%)	86/600 (14.3%)
DeepSeek-Prover-V2-7B	50/139 (36.0%)	92/349 (26.4%)	4/112 (3.6%)	146/600 (24.3%)
Goedel-Prover-V2-32B	28/139 (20.1%)	58/349 (16.6%)	2/112 (1.8%)	88/600 (14.7%)
GD-8B (w/ Self-Correction)	67/139 (48.2%)	120/349 (34.4%)	13/112 (11.6%)	200/600 (33.3%)
GD-32B (w/ Self-Correction)	81/139 (58.3%)	144/349 (41.3%)	24/112 (21.4%)	249/600 (41.5%)

Table 4: Model Performance by MathLib Coverage Category. GD denotes Goedel-Prover-V2.

3 Experiments

3.1 Experimental Setup

We evaluate four public SOTA ATP models: DeepSeek-Prover-v2-7B (Ren et al., 2025), Goedel-Prover-v2-8B (Lin et al., 2025c), Goedel-Prover-v2-32B (Lin et al., 2025c), and Kimina-Prover-8B (Wang et al., 2025). We use default inference and decoding settings with a temperature of 1.0 and reasoning chains of up to 8192 tokens. We use Lean 4.26.0 for all experiments mentioned (including Lean compiler tools). We report pass@128 accuracy for a single end-to-end (E2E) generation as the primary metric for each model. We also include the pass@32 accuracy for the self-correction setting for both Goedel-Prover-v2 models, which allows the model to access feed-

back from a REPL-based lean compiler (See Appendix A.12 for details).

We also evaluate two frontier models: GPT-5.5 (OpenAI, 2026) and Claude-4.7-Opus (Anthropic, 2026). We report pass@4 accuracy for a single E2E generation as the primary metric and provide each model with its maximum reasoning budget. We also report an agentic setting for GPT-5.5 evaluated at pass@1 that provides the model with a REPL-based lean compiler tool.

3.2 Main Results

Table 1 reports pass@128 results on TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB. Although the two versions of the benchmarks consist of mathematically equivalent problem sets, all evaluated provers perform sub-

stantially worse on the Tao formulations than on the MathLib formulations. This consistent degradation highlights the difficulty current provers face under a shift in definitional frameworks.

The performance gap persists across ATP model families, parameter scales, and inference configurations, indicating a shared limitation rather than model-specific deficiencies. On TAOANALYSISBENCH, overall E2E performance remains low: the strongest result is achieved by DeepSeek-Prover-V2-7B at 12.00%, while the largest model, Goedel-Prover-V2-32B, reaches 6.83%. In contrast, on TAOANALYSISBENCH-MATHLIB, models under the E2E setting perform $2.18\times$ better. This uniform gap suggests that current public ATPs consistently generalize poorly outside the MathLib definitional framework on which they are primarily trained and evaluated.

Our evaluation of frontier foundation models in Table 2, while not directly comparable, provides context for the performance of SOTA ATP models. Although these models are not explicitly trained for formal theorem proving, they are effective at using in-context information to solve unfamiliar problems. This comparison allows us to assess whether the Tao vs. MathLib performance gap reflects issues specific to ATP specialization or a broader difficulty posed by the Tao definitional framework. As shown in Table 2, both GPT-5.5 and Claude-4.7-Opus exhibit significant performance improvements on TAOANALYSISBENCH-MATHLIB as compared to TAOANALYSISBENCH.

While E2E model performance tends to be very low, we find that models evaluated under settings that allow iterative refinement with verification offers significant performance improvements under similar inference budgets and lower pass rates. Goedel-Prover-8B and Goedel-Prover-32B with self-correction improves $2.57\times$ and $2.63\times$ over end-to-end generation on TAOANALYSISBENCH and $2.31\times 2.57\times$ and $2.82\times$ over end-to-end generation on TAOANALYSISBENCH-MATHLIB, respectively, with half the token budget. The agentic setting with compilation tool for GPT-5.5 improves $4.27\times$ over end-to-end generation on TAOANALYSISBENCH and $2.66\times$ over end-to-end generation on TAOANALYSISBENCH-MATHLIB with a quarter of the budget. Under the self-correction setting, performance on TAOANALYSISBENCH-MATHLIB was still 19.83 and 23.50 percentage points higher for Goedel-Prover-V2-8B and Goedel-Prover-V2-32B with

self-correction, respectively. Under the agentic setting with the compilation tool for GPT-5.5, performance was 8.10 percentage points higher. TAOANALYSISBENCH therefore exposes a clear gap between in-distribution benchmark success and robustness to definitional variation.

4 Analysis

4.1 Environmental Enrichment Ablation

To provide a fair comparison between TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB, we incorporate a two-stage process for aggregating a compilable environment for each exercise. We first collect the minimal compilable environment that consists of all statements needed to compile the exercise. We then enrich this environment with access to search-tactic-tagged textbook modules and additional statements from the exercise’s parent module. Table 3 shows the results of this ablation. We observe a modest improvement in the performance of each model, roughly 2 percentage points on average. However, Kimina-Prover-8B shows a $6.25\times$ improvement over the minimal context, which indicates heavy reliance on solver tactics to close proofs.

4.2 Coverage Analysis

To characterize each exercises relation to existing Mathlib content, we classify it into three categories: A: in Mathlib; B: in Mathlib but custom-built; and C: not in Mathlib, following the taxonomy of the Mathlib coverage analysis of Taos Analysis I exercises (Kirov, 2025). The category definitions are summarized in Table 5. We perform this classification using an LLM-based agent with GPT-5-Pro. For each exercise, the agent identifies the statement and any custom types, then uses rg (ripgrep) to search a local MathLib clone for candidate lemmas, returning matching lines. Based on these results, it outputs the coverage category, the matching MathLib lemma and file path when applicable, and a one-sentence justification. The aggregated outcomes are reported in Figure 4.

Table 4 reports performance by MathLib coverage category for four open-source provers on TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB. The Tao–MathLib gap is strongly mediated by coverage category. Models perform best on Category A problems, where statements are directly supported by MathLib types and named lemmas, but degrade sharply on Categories B and

C. The gap is especially pronounced in Category B: although the underlying results exist in MathLib, the Tao formulation uses custom types or definitions, reducing pass rates on TAOANALYSISBENCH to 0.6%–2.9%, compared with 9.2%–26.4% on TAOANALYSISBENCH-MATHLIB. Category C remains the hardest setting, with near-zero performance on Tao formulations and only marginal gains after MathLib translation. These results support our central claim that current provers are highly sensitive to definitional framing. They perform best when a problem can be connected directly to existing MathLib knowledge, but struggle sharply when equivalent mathematics is presented through custom textbook formalizations. Results for proprietary and agentic models are provided in Appendix A.6.

5 Related Works

5.1 Formal Mathematical Benchmarks

Early benchmarks for ATP models involved large-scale extraction of theorems and complementary proofs from MathLib (Yang et al., 2023). However, the standard for benchmarking formal reasoning capabilities has become high-school-level competition math exam problems, such as the International Math Olympiad (Trinh et al., 2024; Luong and Lockhart, 2025; OpenAI, 2025; Yu et al., 2025; Hubert et al., 2025). Competition-style benchmarks such as MINIF2F (Zheng et al., 2022), AIME (AI-MO, 2025), and Putnam-Bench (Tsoukalas et al., 2024) aggregate competition problems from past years and are widely used as standard evaluation targets for end-to-end proving performance. Other datasets focus on translating from informal language to formal languages: ProofNet (Azerbayev et al., 2023), the HERALD dataset (Gao et al., 2025), and Lean Workbook (Ying et al., 2024) provide natural-language statements (and often proofs) with mathematically equivalent formalizations, enabling evaluation of both auto-formalization and proof search. More recent efforts expand topical coverage beyond contest mathematics (Liu et al., 2025; Jiang et al., 2025; Ren et al., 2025).

5.2 Automated Theorem Proving

Automated theorem proving in Lean has evolved rapidly, with earlier approaches leveraging inference-time strategies (Yang et al., 2023; Lin et al., 2025a; Xin et al., 2025) and more recent

approaches using LLMs for end-to-end generation (Ren et al., 2025; Lin et al., 2025c; Wang et al., 2025). Inference-time strategies such as best-first or expert-iteration search established strong performance on standard benchmarks (Xin et al., 2025). Subsequent approaches that incorporated synthetic data generation and reinforcement learning further improved the state-of-the-art (Dong and Ma, 2025). The DeepSeek-Prover-V2 (Ren et al., 2025), Kimina-Prover (Wang et al., 2025), and Goedel-Prover-V2 model (Lin et al., 2025c) families adopt iterative self-improvement loops that bootstrap from successful proofs and sub-goal decompositions. However, these pipelines rely largely on MathLib formalizations, raising open questions about robustness and generalization under shifts in underlying definitions and formal representations.

6 Conclusion

Our work identifies a concrete limitation of current public LLM-based ATP models. We observe that strong performance on Mathlib-formulated problems does not reliably transfer to definitionally equivalent formalizations outside the MathLib framework. Across model families and scales, accuracy degrades substantially when core mathematical concepts are reintroduced through alternative definitions. Our analysis of the impact of context features and manual evaluation of the results indicate that this gap is due more to sensitivity to novel formulations in TAOANALYSISBENCH than to increased mathematical difficulty over TAOANALYSISBENCH-MATHLIB. This poses an acute problem because much of exploratory mathematics takes place in novel definitional frameworks that are distinct from MathLib. By isolating the effects of changing definitional frameworks, TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB show that existing ATP benchmarks conflate general mathematical ability with familiarity with a specific definitional framework. We believe that TAOANALYSISBENCH, TAOANALYSISBENCH-MATHLIB, and our agentic pipelines will provide useful tools for future research into public ATP models that can be useful to research mathematics.

Limitations

While TAOANALYSISBENCH intentionally focuses on undergraduate-level analysis drawn from Taos *Analysis I* in order to isolate the effect of unfamiliar definitional frameworks while controlling for mathematical difficulty, this design also limits its scope. The benchmark is restricted to a single source and does not fully capture the complexity of research-level formalization, where custom frameworks may involve deeper mathematics, larger dependency structures, and more exploratory proof development. We therefore leave the evaluation of ATP models on research-level custom formal frameworks to future work.

Ethical Considerations

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

7 Acknowledgments

The work was partially supported by DARPA HR001126CE054, NSF 2106859, 2312501, 2617441, 2609582, Amazon, NEC, and OpenAI.

References

- AI-MO. 2025. Ai-mo aime dataset. <https://huggingface.co/datasets/AI-MO/aime-validation-aime>.
- Anthropic. 2026. Introducing claude opus 4.7. <https://www.anthropic.com/news/claude-opus-4-7>. Accessed: 2026-05-06.
- Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023. *Proofnet: Autoformalizing and formally proving undergraduate-level mathematics*. *Preprint*, arXiv:2302.12433.
- The Lean Prover Community. 2025. mathlib4: The math library of Lean 4. <https://github.com/leanprover-community/mathlib4>.
- R my Degenne, David Ledvinka, Etienne Marion, and Peter Pfaffelhuber. 2025. *Formalization of brownian motion in lean*. *Preprint*, arXiv:2511.20118. ArXiv preprint, accessed 2026-01-24.
- Kefan Dong and Tengyu Ma. 2025. *STP: Self-play LLM theorem provers with iterative conjecturing and proving*. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 14114–14136. PMLR.
- frenzymath. 2024. jixia: A static analysis tool for lean 4. <https://github.com/frenzymath/jixia>.
- Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. 2025. *Herald: A natural language annotated lean 4 dataset*. In *International Conference on Learning Representations (ICLR)*.
- Georgia Harbor-Collins and Mohit Hulse. 2025. Formalizing class field theory. <https://leanprover-community.github.io/blog/posts/cmi-class-field-theory-workshop/>. Lean prover community blog post, published 2025-08-22, accessed 2026-01-24.
- Thomas Hubert, Rishi Mehta, Laurent Sartran, and 1 others. 2025. *Olympiad-level formal mathematical reasoning with reinforcement learning*. *Nature*. Published 12 Nov 2025.
- Jiedong Jiang, Wanyi He, Yuefeng Wang, Guoxiong Gao, Yongle Hu, Jingting Wang, Nailong Guan, Peihao Wu, Chunbo Dai, Liang Xiao, and Bin Dong. 2025. *Fate: A formal benchmark series for frontier algebra of multiple difficulty levels*. *Preprint*, arXiv:2511.02872.
- Radoslav Kirov. 2025. Mathlib coverage analysis of tao’s analysis i exercises. https://github.com/rkirov/analysis/blob/mathlib-coverage-analysis/mathlib_coverage_analysis.md. Accessed: 2026-05-07.
- Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. 2025a. *Lean-star: Learning to interleave thinking and proving*. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, Singapore, April 24–28, 2025. OpenReview.net.
- Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. 2025b. *Goedel-prover: A frontier model for open-source automated theorem proving*. In *Second Conference on Language Modeling (COLM 2025)*.
- 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. 2025c. *Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction*. *Preprint*, arXiv:2508.03613.
- Junqi Liu, Xiaohan Lin, Jonas Bayer, Yael Dillies, Weijie Jiang, Xiaodan Liang, Roman Soletskyi, Haiming Wang, Yunzhou Xie, Beibei Xiong, Zhengfeng Yang, Jujian Zhang, Lihong Zhi, Jia Li, and Zhengying Liu. 2025. *Combibench: Benchmarking llm capability for combinatorial mathematics*. *Preprint*, arXiv:2505.03171.

- Thang Luong and Edward Lockhart. 2025. Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad. Online; accessed 2025-10-03.
- OpenAI. 2025. Openai status update [tweet]. <https://x.com/OpenAI/status/1946594928945148246>. Online; accessed 2025-10-03.
- OpenAI. 2026. Using gpt-5.5. <https://developers.openai.com/api/docs/guides/latest-model>. Accessed: 2026-05-06.
- Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanxia Zhao, Liye Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *Preprint*, arXiv:2504.21801.
- Terence Tao. 2023. Formalizing the proof of pfr in lean4 using blueprint: a short tour. <https://terrytao.wordpress.com/2023/11/18/formalizing-the-proof-of-pfr-in-lean4-using-blueprint-a-short-tour/>. Blog post, accessed 2026-01-24.
- Terence Tao. 2025a. teorth/analysis: A lean companion to analysis i. <https://github.com/teorth/analysis>. GitHub repository; Apache-2.0 license; Online; accessed 2025-10-03.
- Terence Tao. 2025b. [teorth/analysis](https://github.com/teorth/analysis) (github repository).
- Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, Thang Luong, and 1 others. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625(7941):476–482. Published January 17, 2024; correction published February 23, 2024.
- George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. 2024. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. In *Advances in Neural Information Processing Systems*, volume 37, pages 11545–11569. Neural Information Processing Systems Foundation.
- Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, and 21 others. 2025. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *Preprint*, arXiv:2504.11354.
- Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *Preprint*, arXiv:2405.14333.
- Ran Xin, Zeyu Zheng, Yanchen Nie, Kun Yuan, and Xia Xiao. 2025. Scaling up multi-turn off-policy RL and multi-agent tree search for LLM step-provers. *CoRR*, abs/2509.06493.
- Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J. Prenger, and Animashree Anandkumar. 2023. Leandjo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 21573–21612. Curran Associates, Inc. NeurIPS 2023 Datasets and Benchmarks Track.
- Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Lean workbook: A large-scale lean problem set formalized from natural language math problems. In *Advances in Neural Information Processing Systems*, volume 37, pages 105848–105863.
- Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, Yandong Wen, Ge Zhang, and Weiyang Liu. 2025. Formal-math: Benchmarking formal mathematical reasoning of large language models. *arXiv preprint arXiv:2505.02735*.
- Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations (ICLR)*. Poster.

Category	Description
A	In MathLib. Uses MathLib types and has a named MathLib lemma; the proof can usually be closed by the lemma directly or with minor glue.
B	In MathLib, but custom-built. The mathematical result exists in MathLib, but the exercise uses custom types or definitions, so the prover must reconstruct the proof through the textbook framework.
C	Not in MathLib. Counterexamples, decidability puzzles, bridge lemmas, computation exercises, or results specific to the textbook’s custom framework.

Table 5: MathLib coverage categories.

A Appendix

A.1 AI Usage

We acknowledge the use of AI assistants in writing the code used in the experiments, as well as the use of LLM-as-a-judge to enforce data quality (as mentioned in A.5). However, AI assistants were not used for ideation nor the writing of the paper.

A.2 Licensing and Intended Use

TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB are intended for the evaluation of LLM-based ATPs under a definitional framework shift. The lean formalized companion to Tao’s *Analysis I* was released under Apache-2.0, and our use is consistent with its licensing.

A.3 Computational Budget

Our open-source prover evaluations were run across 12 H100 nodes and 4 H200 nodes, requiring approximately 15k GPU hours. Frontier models were run through APIs, thus their computational budget is not available.

A.4 Benchmark Construction Agents

In the below sections, we provide descriptions of the development of each agent used in benchmark construction.

A.4.1 Minimal Compilable Environment

We find that naïvely including entire chapters or sections of the textbook leads to excessive context length, compilation failures, and unfaithful reconstructions of the original problem and its intent. We observe that without explicit targets,

modern frontier LLMs struggle to identify *all* information critical for compilation over large textbook contexts containing extensive irrelevant information. Furthermore, querying frontier LLMs with the objective of compilation may succeed in compilation while silently changing the intended meaning. For instance, the model may introduce a local alias that collapses one or more definitions and, in so doing, trivializes the goal, e.g., `abbrev Sequence.IsCauchy := True`. Our agentic pipeline is built with these issues in mind. As mentioned in the main text, we pre-process the data using the JiXia tool to identify the minimal set of statements required to compile each problem in the problem set.

However, retrieval based on the JiXia table alone is insufficient. Lean environments constructed after single query attempts frequently fail compilation checks due to missing information or simple syntax errors. Thus to aid in the construction of a compilable environment, we introduce the file-lookup tool into the agent that allows the model to retrieve source files. This enables access to the local structure required to faithfully reconstruct compilable Lean environments for each problem.

Lastly, we introduce the Lean-Compilation tool to allow the model to verify the compilation of the minimal compilable environment.

A.4.2 Analysis I-Tagged and Enhanced Compilable Environment

To annotate textbook modules with tactic tags, we nest an agentic pipeline within the larger pipeline that consists of two agents: an annotator agent and a verifier agent. The annotator agent is provided a build-validate-strip tool that iteratively builds the project and strips breaking tag attributes to ensure that the module compiles cleanly. However, compilation of the module belies issues with search time. To compensate for over-tagging that bogs down verifications with unnecessarily deep searches, the verifier agent provides a review of the annotations for each module and provides feedback to the annotation agent.

To allow the minimal environment to access the tagged versions of the textbook, we use an agent to enrich the minimally compilable environment. This enrichment consists of both adding textbook module imports and statements from the exercise’s parent module to the minimal compilable environments in TAOANALYSISBENCH. A naïve im-

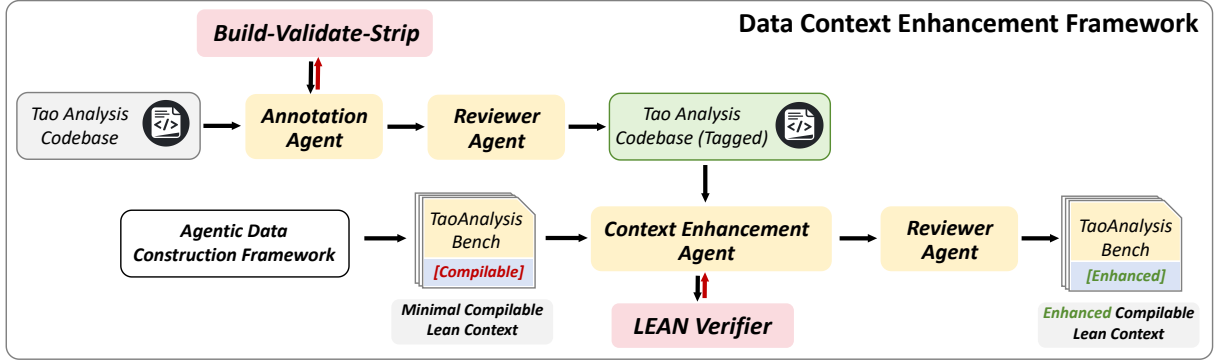


Figure 5: Agentic framework for constructing a search-tactic-tagged version of *Analysis I* and, downstream, an enhancement to the minimal environment provided in TAOANALYSISBENCH to allow models to fully leverage automated search tactics.

port of the exercise’s parent module would allow the model to close the goal with the version of the exercise present in the textbook’s parent module. Thus, to allow search tactics to access statements from the parent module, the enrichment agent folds statements from the parent module (using the tagged version of the textbook) into the minimal compilable environment of each exercise in TAOANALYSISBENCH.

A.4.3 TAOANALYSISBENCH-Mathlib

These paired problems in TAOANALYSISBENCH-MATHLIB enable an in-distribution evaluation of existing public provers under the MathLib definitional framework. Constructing a MathLib counterpart for each exercise in TAOANALYSISBENCH introduces a set of challenges similar, yet distinct from the construction of TAOANALYSISBENCH. In particular, such a reformulation must simultaneously satisfy two constraints: (i) the resulting statement must be syntactically valid and compilable, and (ii) it must remain mathematically faithful to the original textbook formulation while using MathLib formalizations. As shown in Figure 3, we build an agentic pipeline to perform this reformulation at scale, followed by expert manual verification of the resulting MathLib statements to ensure mathematical equivalence with their Tao counterparts. Our pipeline consists of two stages: reformulation of the TAOANALYSISBENCH exercise environment to the MathLib definitional framework and equivalence verification.

The first stage is the reformulation of the TAOANALYSISBENCH exercises. Naïve, end-to-end translation of textbook formalizations into MathLib formulations using frontier language models proves unreliable: only roughly 60% of outputs were syntactically valid Lean code, and a substan-

tially smaller fraction were mathematically equivalent to the original statements. Inspection of failure cases revealed that most errors stemmed from mismatches between Tao’s local definitions and their MathLib counterparts. In particular, the model frequently selected incorrect definitions or employed non-canonical usage patterns, leading to both compilation failures and subtle semantic deviations.

To enable the model to ground its translation in MathLib’s canonical definitions and usage conventions, we enable web-search to allow the model to retrieve authoritative documentation and examples for core MathLib constructs. We also explicitly restrict the model from introducing new local definitions or notation and require adherence to a fixed MathLib header, ensuring consistency across translated problems. The agent then verifies that the resulting reformulation compiles using the Lean compiler tool.

Compilation alone, however, does not guarantee semantic faithfulness. Thus, we include the second stage: equivalence verification. We provide the agent with a JiXia-based goal-state extraction tool, which extracts the target proof states for both the Tao and MathLib formulations and grounds the comparison of their respective proof obligations. The agent assesses whether the two formulations are mathematically equivalent given their statements and target states.

A.5 Expert Manual Annotation of TAOANALYSISBENCH-MATHLIB

We verified the fidelity of the MathLib equivalent problems translated from the textbook versions via both manual expert reviews and LLM-as-evaluator passes. The reviewer qualifications were completion of an equivalent Analysis I course and ex-

perience in Lean programming. Reviewers were sourced from those participating in the project, and the contribution was voluntary and unpaid. We provided both the Tao and MathLib versions of the data to the annotators and asked the annotators to gauge mathematical equivalence according to the grades outline in Table 6. Annotators were asked to provide a manual translation of the Tao example in MathLib syntax if they identified MathLib translations that were graded as a "C" or a "D". The interface for human annotation is shown in Figure 6. We also applied an additional annotation pass by GPT-5-Pro: we provide the model with the annotation guidelines and in-context examples of corrections made by human annotators and query the model to grade the translation and provide a re-translation if the translation grade is a "C" or a "D". One annotator manually reviewed the grades given by the LLM and proposed corrections to produce the final version of the benchmark.

Judgment Labels Our labeling scheme is motivated by an important practical consideration: some definitions in Tao’s *Analysis I* differ from their MathLib counterparts. In the Lean formalization of *Analysis I*, Tao generally favors faithful translation over more idiomatic Lean formulations when the two diverge (Tao, 2025b). Consequently, certain textbook exercises do not admit a strictly equivalent statement in MathLib. We therefore instructed judges to evaluate the core intent of each exercise and the corresponding proof strategy across the two versions. If these are essentially preserved, we regard the pair as acceptable even when the MathLib statement differs slightly from the Tao statement. Based on these considerations, we define four judgment labels, summarized in Table 6.

Judgment Process For each exercise pair, judges first identified the core intent of the two versions and considered whether their proof strategies align. If the intent and proof strategy were essentially the same, the instance was labeled A or B; otherwise, it was labeled C or D. For label B instances, judges were required to explicitly describe the differences between the two versions. For labels C and D, judges were required to (i) describe the differences and (ii) provide a corrected MathLib version of the exercise.

Our benchmark comprises ten chapters (Chapters 2-11) and sixty-five subsections that follow a standard progression from foundational material

Label	Description
A	Fully Equivalent. The two versions express the same mathematical statement. They are logically equivalent under standard mathematical interpretations.
B	Nearly Equivalent. The exercise’s intent and proof strategy remain essentially the same. However, due to definitional differences between Tao’s textbook and Mathlib, the Mathlib formulation is slightly different from the corresponding Tao statement.
C	Minor Differences. The exercise’s intent and proof strategy are not preserved across the two versions, but the discrepancy is minor and requires only small adjustments.
D	Major Differences. The exercise’s intent and proof strategy are not preserved across the two versions, and there is a substantial mismatch in meaning.

Table 6: Definitions of four judgment labels.

(natural numbers, set theory, and number systems) to core real analysis topics, including limits, series, continuity, differentiation, and the Riemann integral. Figure 7 summarizes the chapter-level composition.

Verification outcomes The initial human annotation covered 150 exercise pairs. The resulting translation-equivalence grades were 74.8% A, 19.0% B, 1.0% C, and 5.2% D. GPT-5.5-Pro-xhigh was used to review these examples, including the re-translations proposed by the human annotators. This review identified 30 issues, 12 of which concerned subtle errors in the human re-translations, and proposed corrections for all 30 issues. The human annotators reviewed and accepted each proposed correction.

The calibrated LLM evaluator was then applied to all 600 exercise pairs. The initial evaluation assigned 308 pairs to A, 276 to B, 11 to C, and 5 to D. The 16 C/D pairs were corrected and manually re-verified. This yielded a final benchmark containing 313 A pairs and 287 B pairs. Thus, 52.2% of the final pairs are fully equivalent and 47.8% are nearly equivalent under the criteria in Table 6.

A.6 Additional Coverage Analysis

Table 8 shows that the same coverage-dependent pattern also holds for frontier proprietary models. GPT-5.5 and Claude-4.7-Opus achieve their strongest performance on Category A problems, where the statements are closest to standard Math-

Stage	A	B	C	D
Initial Pass ($n = 600$)	308	276	11	5
Final Pass ($n = 600$)	313	287	0	0

Table 7: Translation-equivalence grades at each verification stage. Grade A denotes full equivalence, while grade B preserves the exercise’s mathematical intent and proof strategy with minor definitional differences.

Lib formulations, and their performance drops substantially on Categories B and C. In particular, for Category B, both models perform much better on TAOANALYSISBENCH-MATHLIB than on TAOANALYSISBENCH: GPT-5.5@4 improves from 11.7% to 33.2%, while Claude-4.7-Opus@4 improves from 6.6% to 29.8%. This again suggests that many Category B problems are mathematically accessible to the models once expressed through familiar MathLib definitions, but become difficult when the same content is presented through Taos custom formalization. Category C remains the most challenging setting for both models, although GPT-5.5 obtains nontrivial gains under MathLib translation, reaching 19.6% at pass@4.

Table 9 further shows that an agentic prover built on GPT-5.5 substantially narrows the gap across all categories, achieving 87.7% overall on TAOANALYSISBENCH and 95.8% on TAOANALYSISBENCH-MATHLIB. Nevertheless, even this agentic setting retains a visible drop under the Tao framework, especially in Category C, where performance decreases from 83.9% to 69.6%. These results indicate that stronger general-purpose reasoning and agentic tool use can mitigate, but not fully eliminate, the difficulty of reasoning under unfamiliar definitional frameworks.

A.7 Data Distribution

A.8 Model Performance by Chapter

Figure 8 reports chapter-level pass rates and shows that the TaoMathLib gap is highly uneven across topics. The largest MathLib advantages occur in chapters whose core objects are represented through substantially different local definitions, including The Natural Numbers (Ch. 2), Integers and Rationals (Ch. 4), The Real Numbers (Ch. 5), and The Riemann integral (Ch. 11). In these chapters, models often fail almost entirely on the

Tao formulation while retaining nontrivial performance on the MathLib translation, indicating that the bottleneck is primarily definitional rather than mathematical. In contrast, chapters such as Limits of Sequences (Ch. 6), Series (Ch. 7), Infinite Sets (Ch. 8), and Continuous Functions on $\mathbb{R}$ (Ch. 9) exhibit smaller gaps and in some cases near-parity.

Notably, Differentiation of Functions (Ch. 10) even shows a slight reversal for several models, with better performance on TAOANALYSISBENCH than on TAOANALYSISBENCH-MATHLIB. Overall, these results suggest that Tao formulations are not uniformly harder; instead, the difficulty depends on how strongly each chapter’s local formalization departs from MathLib conventions.

A.9 Context Length

We analyze how model performance is impacted by the number of local definitions provided in each context. Problems in TAOANALYSISBENCH contain the minimal local environment, including bespoke definitions, notation, and supporting lemmas, required to ensure that the statement compiles. In contrast, TAOANALYSISBENCH-MATHLIB is evaluated in the standard import Mathlib environment and does not include additional local definitions in the prompt. Figure 9 divides the benchmark into four buckets according to the number of locally introduced definitions required in the corresponding TAOANALYSISBENCH context: $n = 0$, $1 \leq n < 5$, $5 \leq n < 10$, and $n \geq 10$.

A clear divergence emerges between the Tao and MathLib formulations as the number of local definitions increases. On TAOANALYSISBENCH, all models achieve their highest pass rates when no local definitions are required, with performance ranging from 2% to 29% at $n = 0$. However, performance drops sharply once local definitions are introduced. For $1 \leq n < 5$ and $5 \leq n < 10$, all models remain at or below 4%, and for $n \geq 10$, performance is nearly zero across the board. This indicates that the presence of unfamiliar local definitions substantially reduces the effectiveness of current ATP models, even when those definitions are provided in context.

By contrast, performance on TAOANALYSISBENCH-MATHLIB is more stable across the same buckets. Although there is some degradation from $n = 0$ to $1 \leq n < 5$, the MathLib formulation remains consistently

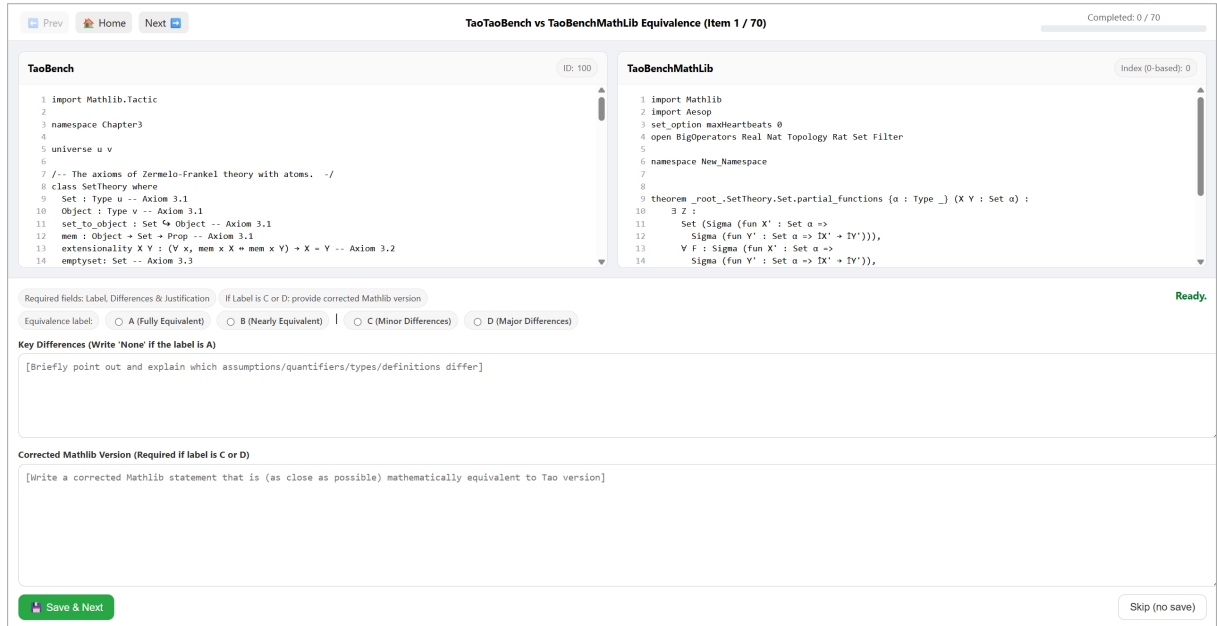


Figure 6: Human Annotation Interface

Model	A	B	C	Overall
	Pass(%)	Pass(%)	Pass(%)	Pass(%)
TAOANALYSISBENCH				
GPT-5.5@1	54/139 (38.8%)	23/349 (6.6%)	4/112 (3.6%)	81/600 (13.5%)
GPT-5.5@4	74/139 (53.2%)	41/349 (11.7%)	8/112 (7.1%)	123/600 (20.5%)
Claude-4.7-Opus@1	24/139 (17.3%)	8/349 (2.3%)	0/112 (0.0%)	32/600 (5.3%)
Claude-4.7-Opus@4	61/139 (43.9%)	23/349 (6.6%)	1/112 (0.9%)	85/600 (14.2%)
TAOANALYSISBENCH-MATHLIB				
GPT-5.5@1	58/139 (41.7%)	74/349 (21.2%)	12/112 (10.7%)	144/600 (24.0%)
GPT-5.5@4	78/139 (56.1%)	116/349 (33.2%)	22/112 (19.6%)	216/600 (36.0%)
Claude-4.7-Opus@1	43/139 (30.9%)	64/349 (18.3%)	10/112 (8.9%)	117/600 (87.7%)
Claude-4.7-Opus@4	65/139 (46.8%)	104/349 (29.8%)	14/112 (12.5%)	183/600 (30.5%)

Table 8: Model Performance by Mathlib Coverage (Proprietary Model)

Model	A	B	C	Overall
	Pass@1(%)	Pass@1(%)	Pass@1(%)	Pass@1(%)
TAOANALYSISBENCH				
Agentic prover (GPT-5.5)	135/139 (97.1%)	313/349 (89.7%)	78/112 (69.6%)	526/600 (87.7%)
TAOANALYSISBENCH-MATHLIB				
Agentic prover (GPT-5.5)	139/139 (100%)	342/349 (98.0%)	94/112 (83.9%)	575/600 (95.8%)

Table 9: Model Performance by Mathlib Coverage (Agentic prover)

more solvable than the Tao formulation once local definitions are present. In particular, for $n \geq 10$, TAOANALYSISBENCH-MATHLIB pass rates remain between 8% and 25%, while the corresponding TAOANALYSISBENCH pass rates

are at or near 0%. This suggests that the underlying problems remain accessible when expressed using familiar MathLib definitions, and that the difficulty in TAOANALYSISBENCH is primarily induced by the shift in definitional framework

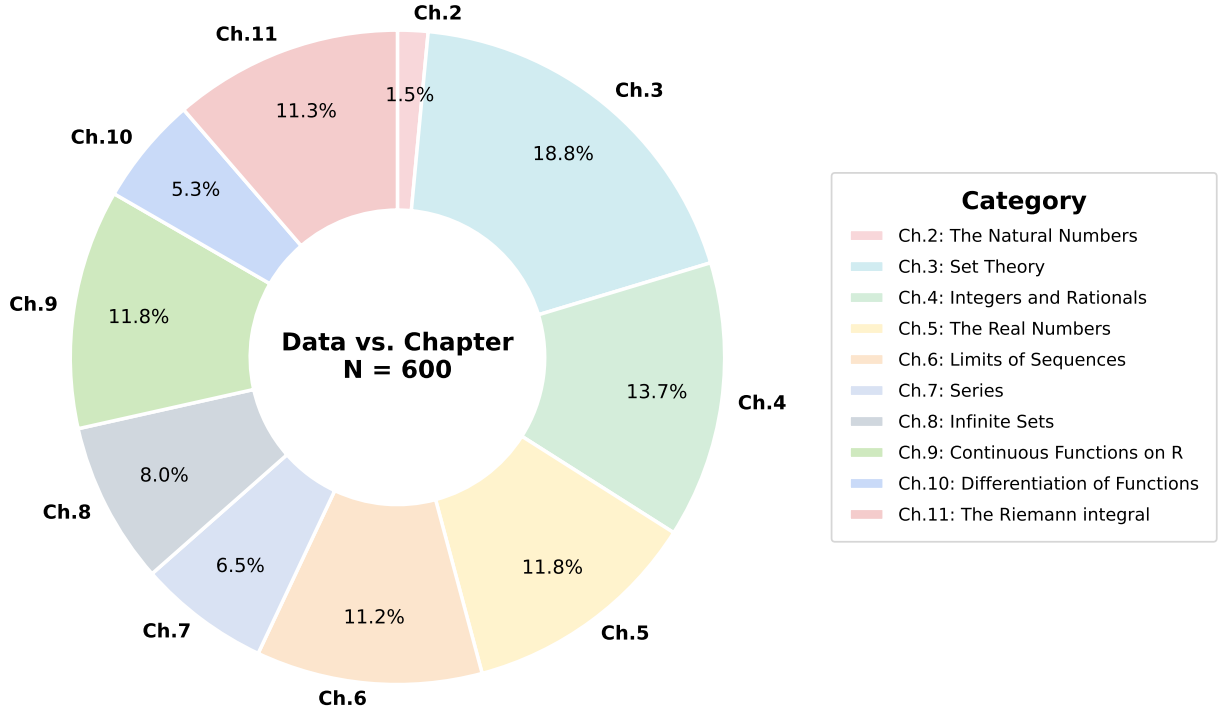


Figure 7: Data Distribution by Chapter.

Model	TAOANALYSISBENCH		TAOANALYSISBENCH-MATHLIB		
	Pass@128	(%)	Pass@128	(%)	Ratio
Kimina-Prover-8B	11/600	1.90	38/600	6.30	3.45×
Goedel-Prover-V2-8B	32/600	5.40	93/600	15.50	2.91×
DeepSeek-Prover-V2-7B	71/600	11.83	149/600	24.80	2.10×
Goedel-Prover-V2-32B	<u>34/600</u>	<u>5.60</u>	<u>90/600</u>	<u>15.00</u>	2.65×

Table 10: Model performance on TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB using FP16.

Model	Minimal-Compilable		Enhanced		Ratio
	Pass@128	(%)	Pass@128	(%)	
Kimina-Prover-8B	11/600	1.90	10/600	1.60	2.91×
Goedel-Prover-V2-8B	32/600	5.40	41/600	6.90	2.91×
DeepSeek-Prover-V2-7B	71/600	11.83	-	-	2.91×
Goedel-Prover-V2-32B	<u>34/600</u>	<u>5.60</u>	<u>40/600</u>	<u>6.60</u>	2.91×

Table 11: Minimal-Compilable Environment v. Enhanced Environment using FP16

rather than by mathematical complexity alone.

The performance delta further supports this interpretation. At $n = 0$, the gap between TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB is small for most models, indicating that Tao problems without additional local definitions are not inherently harder than their MathLib counterparts. As the number of in-context definitions increases, however, the gap grows steadily

for most models. For $n \geq 10$, the MathLib advantage reaches 25 percentage points for DeepSeek-Prover-V2-7B, 18 points for Goedel-Prover-V2-32B, 16 points for Goedel-Prover-V2-8B, and 8 points for Kimina-Prover-8B. These results show that current provers struggle to integrate and reason over unfamiliar definitions introduced in context, even when the relevant information is explicitly available.

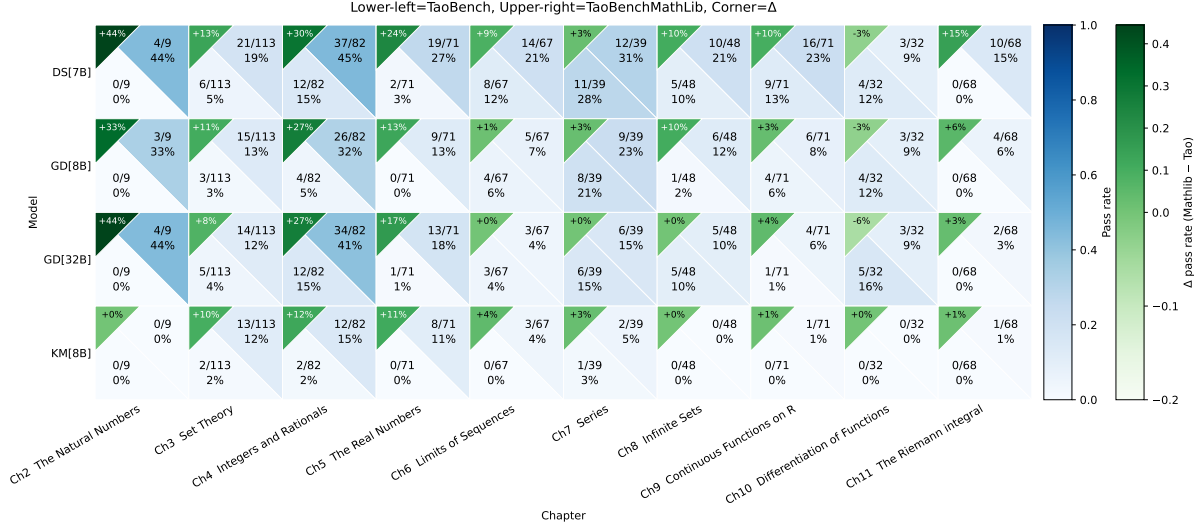


Figure 8: Model Performance by Chapter. For each cell, accuracies on TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB are shown in the upper-right and lower-left corners, respectively. The upper-left corner shows the difference (TAOANALYSISBENCH-MATHLIB - TAOANALYSISBENCH). DS[7B]* indicates DeepSeek-Prover-V2-7B (w/o COT).

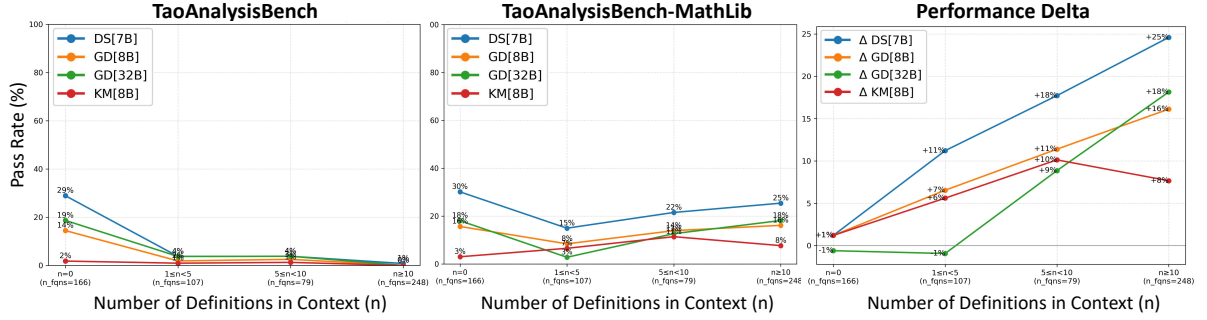


Figure 9: Model Performance vs. Definitions in Context. Performance on TAOANALYSISBENCH and TAOANALYSISBENCH-MATHLIB as a function of the number of in-context definitions. The performance delta is defined as performance (TAOANALYSISBENCH) - performance (TAOANALYSISBENCH-MATHLIB). DS, GD, and KM indicate DeepSeek-Prover-V2, Goedel-Prover-V2, and Kimina-Prover, respectively.

Formal theorem provers are routinely expected to operate in novel definitional environments, where new abstractions must be introduced and used locally. Our results show that current models do not reliably benefit from such in-context definitional information when it departs from the MathLib distribution. This suggests that fine-tuning improves performance within a fixed formal ecosystem, but does not necessarily confer robustness to unseen definitional frameworks, exposing a fundamental limitation of current ATP training paradigms.

A.10 MathLib Translation Pipeline

During the rewriting stage, we forbid the MathLib version from introducing any new local definitions or notation, and restrict it to using the following fixed header:

```
import Mathlib
import Aesop
set_option maxHeartbeats 0
open BigOperators Real Nat Topology Rat
↪ Set Filter
```

A.11 Case Studies

We present two representative instances taken from model performance on complementary problem pairs: one where the prover succeeds on the MathLib formulation but not on the Tao formulation, and one showing the reverse pattern.

Nat.backwards_induction: As shown in Figure 10, this pair highlights how prover performance can depend on access to MathLib’s arithmetic and order-theoretic infrastructure. Empirically, for Goedel-Prover-V2-32B, the TAOANAL-

TaoAnalysisBench

```
import Mathlib.Tactic
namespace Chapter2

inductive Nat where
| zero : Nat
| succ : Nat → Nat
deriving Repr, DecidableEq

postfix:100"++" => Chapter2.Nat.succ

namespace Nat

abbrev recurse (f : Nat → Nat → Nat) (c : Nat) : Nat →
Nat := fun n =>
  match n with
  | Nat.zero => c
  | Nat.succ n => f n (recurse f c n)

abbrev add (n m : Nat) : Nat :=
  Nat.recurse (fun _ sum => sum++) m n

instance instAdd : Add Nat where
  add := add

instance instLE : LE Nat where
  le n m := ∃ a : Nat, m = n + a

end Nat

theorem Nat.backwards_induction {n:Nat} {P: Nat → Prop}
(hind: ∀ m, P (m++) → P m) (hn: P n) :
  ∀ m, m ≤ n → P m := by
  sorry

end Chapter2
```

TaoAnalysisBench-MathLib

```
import Mathlib
import Aesop
set_option maxHeartbeats 0
open BigOperators Real Nat Topology Rat Set Filter
namespace New_Namespace

theorem Nat.backwards_induction {n : ℕ} {P : ℕ → Prop}
(hind : ∀ m, P (m + 1) → P m) (hn : P n) :
  ∀ m, m ≤ n → P m := by
  sorry

end New_Namespace
```

Figure 10: Case Study: Nat.backwards_induction

YSISBENCH version admits 0/128 verified proofs, while the TAOANALYSISBENCH-MATHLIB version admits 64/128 verified proofs.

In the TAOANALYSISBENCH prompt, the prover must reason over a bespoke inductive Nat with addition defined by primitive recursion and order defined existentially by $m \leq n, \exists a, n = m + a$. A backwards-induction argument, therefore, requires reconstructing supporting algebraic and order-theoretic lemmas locally. In the TAOANALYSISBENCH-MATHLIB prompt, these obligations can be met using standard lemmas and automation. The instance-level performance split (0/128 vs. 64/128 for Goedel-32B) corresponds with brittleness to definitional relocation rather than increased mathematical difficulty.

Convergesto.squeeze: As shown in Figure 11, this pair highlights a representational shift that is common throughout analysis formalizations: an explicit ε - δ definition of convergence versus a filter-based Tendsto statement. Empirically, for DeepSeek-Prover-V2-7B, the TAOANALYSISBENCH version admits 2/128 verified proofs, while the TAOANALYSISBENCH-MATHLIB version admits 0/128 verified proofs.

In the TAOANALYSISBENCH prompt,

Convergesto is defined directly via CloseNear: for each $\varepsilon > 0$, one produces a $\delta > 0$ so that $|f(x) - L| < \varepsilon$ holds on a punctured δ -neighborhood intersected with E . A successful proof proceeds by extracting radii δ_f, δ_h from the hypotheses for f and h , choosing $\delta = \min(\delta_f, \delta_h)$, and then deriving the bound for g by combining the pointwise inequalities $f \leq g \leq h$ with the two-sided inequalities obtained from rewriting abs_sub_lt_iff . In contrast, the TAOANALYSISBENCH-MATHLIB version expresses the same squeeze argument as a filter limit $\text{Tendsto } g \text{ (nhdsWithin } x_0 \text{ } E) \text{ (}\mathcal{N}L\text{)}$. Closing the goal in that setting requires translating pointwise bounds on E into Eventually statements in nhdsWithin and then applying an appropriate squeeze principle at the level of filters. The observed asymmetry suggests that, for this model, proof search is more reliable when convergence is presented in an explicit, locally unfolded form than when it is mediated through MathLib’s filter abstraction, even when the underlying mathematics is unchanged.

TaoAnalysisBench

```
import Mathlib.Tactic
import Mathlib.Data.Real.Sign
namespace Real

abbrev CloseFn (ε : ℝ) (X : Set ℝ) (f : ℝ → ℝ) (L : ℝ) : Prop :=
  ∀ x ∈ X, |f x - L| < ε

abbrev CloseNear (ε : ℝ) (X : Set ℝ) (f : ℝ → ℝ) (L : ℝ) (x₀ : ℝ) : Prop :=
  ∃ δ > 0, CloseFn ε (X ∩ Set.Ioo (x₀ - δ) (x₀ + δ)) f L

abbrev adherent' (ε : ℝ) (x : ℝ) (X : Set ℝ) : Prop :=
  ∃ y ∈ X, |x - y| ≤ ε

end Real

namespace Chapter9

abbrev AdherentPt (x : ℝ) (X : Set ℝ) :=
  ∀ ε > (0 : ℝ), ε.adherent' x X

abbrev Convergesto (X : Set ℝ) (f : ℝ → ℝ) (L : ℝ) (x₀ : ℝ) : Prop :=
  ∀ ε > (0 : ℝ), ε.CloseNear X f L x₀

theorem Convergesto.squeeze {E : Set ℝ} {f g h : ℝ → ℝ} {L : ℝ} {x₀ : ℝ} (had : AdherentPt x₀ E)
(hfg : ∀ x ∈ E, f x ≤ g x) (hgh : ∀ x ∈ E, g x ≤ h x)
(hf : Tendsto f (nhdsWithin x₀ E) (ℕ L)) (hh : Convergesto E h L x₀) :
Convergesto E g L x₀ := by
  sorry

end Chapter9
```

TaoAnalysisBench-MathLib

```
import Mathlib
import Aesop
set_option maxHeartbeats 0
open BigOperators Real Nat Topology Rat Set Filter
namespace New_Namespace

theorem Convergesto.squeeze {E : Set ℝ} {f g h : ℝ → ℝ} {L x₀ : ℝ}
(had : x₀ ∈ closure E)
(hfg : ∀ x ∈ E, f x ≤ g x)
(hgh : ∀ x ∈ E, g x ≤ h x)
(hf : Tendsto f (nhdsWithin x₀ E) (ℕ L))
(hh : Tendsto h (nhdsWithin x₀ E) (ℕ L)) :
Tendsto g (nhdsWithin x₀ E) (ℕ L) := by
  sorry

end New_Namespace
```

Figure 11: Case Study: Convergesto.squeeze

A.12 Goedel-Prover-v2 Self-Correction Mode

Here we provide the details of the self-correction mode for Goedel-Prover-v2. The model is given an initial budget of 8192 tokens followed by verification; the model is allowed up to two additional correction rounds with a budget of 4096 tokens if the initial round fails, each followed by a verification (verification feedback is provided back to the model after each failing verification) (Lin et al., 2025c).