HERMES: Towards Efficient and Verifiable Mathematical Reasoning in LLMs

Azim Ospanov [*1,2], Zijin Feng [†1], Jiacheng Sun[1], Haoli Bai[1], Xin Shen[3], and Farzan Farnia[2]

[1]Huawei Technologies
[2]Department of Computer Science and Engineering, The Chinese University of Hong Kong
[3]Celia Team

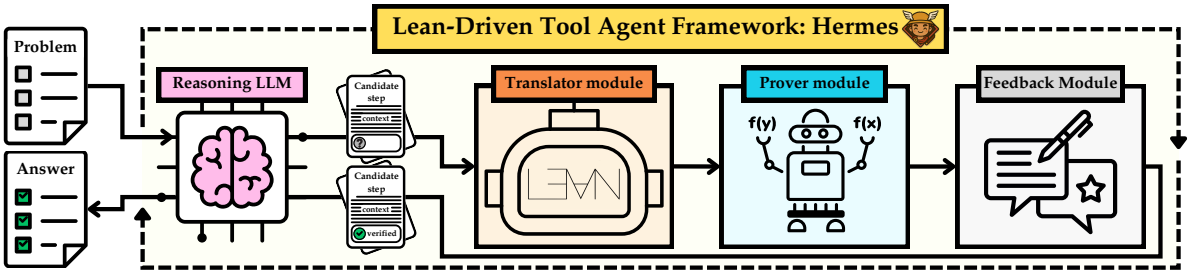


Figure 1: Overview of the *Hermes* framework. *Hermes* is a Lean4-driven, multi-modular reasoning agent integrating LLM reasoning with formal verification for reliable mathematical problem solving. It comprises four modules: an LLM that generates reasoning steps, a translator that formalizes these steps into Lean code, a prover that symbolically verifies their correctness, and a feedback module that returns verification signals for subsequent reasoning. This design enables iterative reasoning with improved correctness and efficiency.

Abstract

Informal mathematics has been central to modern large language model (LLM) reasoning, offering flexibility and enabling efficient construction of arguments. However, purely informal reasoning is prone to logical gaps and subtle errors that are difficult to detect and correct. In contrast, formal theorem proving provides rigorous, verifiable mathematical reasoning, where each inference step is checked by a trusted compiler in systems such as Lean, but lacks the exploratory freedom of informal problem solving. This mismatch leaves current LLM-based math agents without a principled way to combine the strengths of both paradigms. In this work, we introduce *Hermes*, the first tool-assisted agent that explicitly interleaves informal reasoning with formally verified proof steps in Lean. The framework performs intermediate formal checking to prevent reasoning drift and employs a memory module that maintains proof continuity across long, multi-step reasoning chains, enabling both exploration and verification within a single workflow. We evaluate *Hermes* on four challenging mathematical reasoning benchmarks using LLMs of varying parameter scales, from small models to state-of-the-art systems. Across all settings, *Hermes* reliably improves the reasoning accuracy of base models while substantially reducing token usage and computational cost compared to reward-based approaches. On difficult datasets such as AIME'25, *Hermes* achieves up to a 67% accuracy improvement while using 80% fewer total inference FLOPs. The implementation and codebase are publicly available at `https://github.com/aziksh-ospanov/HERMES`.

1 Introduction

In recent years, Large Language Models (LLMs) have achieved remarkable proficiency in mathematical reasoning [1–4], with some systems even demonstrating the potential to solve Olympiad-level problems [5]. A

*First Author, aospanov9@cse.cuhk.edu.hk
†Corresponding Author, fengzijinn@gmail.com

key advancement driving this progress is the Chain-of-Thought (CoT) approach, which enables LLMs to plan step-by-step reasoning, decompose complex problems into sub-goals, generate intermediate reasoning steps, and iteratively assess and correct them. However, long CoTs remain susceptible to logical leaps, subtle errors, and hallucinations, stemming from incomplete domain knowledge, imprecise reasoning, or the accumulation of small mistakes over multiple steps [6–8]. These issues could lead to unstable reasoning and degraded performance, and when uncertain, LLMs may produce overly long or repetitive reasoning traces, increasing token usage without necessarily improving correctness.

To address these limitations, researchers have proposed Process Reward Models (PRMs) and Outcome Reward Models (ORMs) [9,10], which aim to guide LLM reasoning toward correctness by scoring intermediate reasoning steps or final solutions. Specifically, PRMs assign a score at each reasoning step, rewarding local correctness, while an ORMs provide a single scalar reflecting overall solution quality. While these models can improve reasoning performance, they function largely as black-box evaluators that assign numerical scores without explaining why a reasoning trajectory is valid or flawed, offering limited interpretability and no explicit verification of mathematical correctness [11]. Moreover, their training requires substantial human curation [6], and automated supervision methods introduce noise by inferring step correctness labels from final answers [12], leading to misalignment with true stepwise correctness. Ultimately, because both PRMs and ORMs rely on LLMs as evaluative backbone, their reward signals inherit the stochasticity, biases, and instability of LLM-based judgment [13].

In parallel to these, another line of work has focused on formal theorem proving, which relies on proof assistants with trusted kernels such as Lean4 [14], Coq [15], and Isabelle [16]. These systems enforce rigorous formal, machine-checkable reasoning, in contrast to the informal reasoning of traditional mathematics expressed in natural language. Recently, formal language-based systems such as AlphaProof and AlphaGeometry [17–19] have achieved remarkable success on International Mathematical Olympiad (IMO) problems, rivaling top human performers. Their principal strength lies in complete verifiability and strict immunity to hallucinations, as formal verification is embedded directly into the proof search process, ensuring that inference is rigorously justified.

Inspired by the strengths of both paradigms, in this work, we bridge the gap between formal and informal mathematics by combining the verifiability of formal reasoning with the flexibility and expressiveness of LLM-based informal reasoning. We introduce *Hermes*, a multi-modular scalable tool-augmented agent (abbreviated from "**H**ybrid Ag**E**nt for **R**easoning in **M**athematics with N**E**uro-**S**ymbolic Lean4 verification"), designed to integrate formal verification into the LLM reasoning process. *Hermes* leverages modern LLMs' tool-calling capability to verify individual reasoning steps during inference and builds a memory block that ensures continuity of proof claims. For each critical proof step, the agent translates the natural-language statement into Lean goal, verifies translation consistency through back-translation, and invokes a prover module to attempt a proof or counter-proof. The resulting formal signal is then fed back into the LLM to inform its next reasoning step. The overview of our agentic framework is illustrated in Figure 1. We show that incorporating *Hermes* significantly enhances LLM accuracy across mathematical reasoning benchmarks of varying difficulty. It reduces token usage compared to traditional score-based methods and provides interpretable, step-level correctness feedback, offering transparency into how reasoning paths evolve and why certain trajectories lead to valid conclusions while others result in hallucinations. Our contributions are as follows:

- We develop the first tool-based Lean4 reasoning agent that verifies feasible intermediate proof steps during inference, providing LLMs with symbolic-engine–backed correctness signals for mathematical reasoning.
- We introduce a Lean4-powered memory block that accumulates and validates intermediate claims in context, ensuring cross-step consistency and reducing the propagation of errors in long reasoning chains.
- Comprehensive experiments evaluate the effectiveness and efficiency of *Hermes* against eight baseline methods across four benchmarks. Integrating *Hermes* constantly improves performance across all settings, yielding an average accuracy gain of 14% , and when using DeepSeek-V3.1 as the base model, it achieves up to 67% higher accuracy while using 80% less computational budget on AIME'25 benchmark.

2 Related Works

Automatic Theorem Proving. Recent LLM-based provers have achieved remarkable advances in formal reasoning within Lean4. Goedel-Prover-V2 [20] introduces large-scale, verifier-guided training with scaffolded data synthesis to achieve state-of-the-art accuracy on MiniF2F and PutnamBench. Kimina-Prover [21, 22] emphasizes structured reasoning patterns and reinforcement learning to improve sample efficiency. DeepSeek-Prover-V2 [23] focuses on subgoal decomposition, using hierarchical reasoning to bridge informal and formal proofs. Seed-Prover [24] advances lemma-style reasoning and multi-tier inference, achieving competition-level results on IMO and Putnam benchmarks. Collectively, these systems highlight a substantial body of work on automated theorem proving.

Autoformalization. Recent work in autoformalization, translating informal mathematical statements into formal statements, has made rapid advancements. For example, Goedel-Autoformalizer [20] built a dataset of 1.64 million formal statements from natural-language problems in Lean4 and used this to train a high-performing autoformalizer model. Meanwhile, Mathesis-Autoformalizer [25] introduces a reinforcement-learning framework with a novel "LeanScorer" for assessment, and shows accuracy gains on the Gaokao-Formal benchmark. The Kimina-Autoformalizer [21, 22] similarly converts natural-language problems into Lean4 statements via a fine-tuned LLM and expert iteration. Finally, HERALD-Translator [26] provides a large annotated Lean4 corpus (≈580K statements) by back-translating parts of Mathlib and demonstrates high accuracy in miniF2F [27] benchmark. According to previous research efforts, the current generation of autoformalization models are capable of translating informal statements to formal goals with syntactic and semantic correctness.

LLM collaboration with external experts. Another prominent line of research focuses on using external feedback to guide LLMs toward producing correct responses. Specifically, APOLLO [28] proposed a model-agnostic proof-repair framework that enhances sample efficiency and reliability without increasing model size. Ringer [29] introduced a Coq compiler-based repair agent that adapts to changes in the underlying definitions. Moreover, [30] show that combining the strengths of built-in solvers (e.g., Sledgehammer in Isabelle, built-in tactics in Lean) with LLMs leads to performance gains. [20, 22, 31, 32] report that incorporating compiler feedback encourages LLMs to adapt and correct previously generated proofs. This behavior is observed in general-purpose models but can be further amplified by appending compiler feedback during the fine-tuning stage for dedicated theorem proving LLMs.

Enhanced Reasoning Techniques. To further advance LLM reasoning, score-based models have emerged that evaluate the chain of thought produced by reasoning models. The goal is to identify and prioritize the most effective reasoning traces. Two main approaches exist: Outcome Reward Models (ORMs), which assess only the final answer, and Process Reward Models (PRMs), which score reasoning steps sequentially. [9, 10] Furthermore, Safe [33] proposed training a small LSTM reward model based on Lean4 verified reasoning traces. Another line of work focuses on tool-based agents that equip LLMs with more complex functionality and the ability to query and verify their own outputs. Numerous studies have shown that teaching LLMs to use specific tools enhances their performance on given tasks [34, 35], including reasoning. [36] introduced Chain-of-Abstraction to better leverage tools in multi-step reasoning, while [37] examined the importance of using tools in the right scenarios to improve reasoning in LLMs.

Neurosymbolic Solvers and LLMs. Recent work has advanced neuro-symbolic reasoning by integrating LLMs with formal logic and theorem proving. LogicLM [38] and LogicLM++ [39] translate natural-language problems into symbolic form and iteratively refine them for solver-based verification. DSP [40] and DSP+ [41] bridge informal reasoning and formal proofs through a draft–sketch–prove pipeline. Lean-STaR [42] enhances formal reasoning by interleaving natural-language rationales with Lean tactics, while DTV [43] explores deterministic verifier-guided reasoning. Collectively, these methods demonstrate the growing effectiveness of hybrid LLM–symbolic approaches for logical and mathematical reasoning.

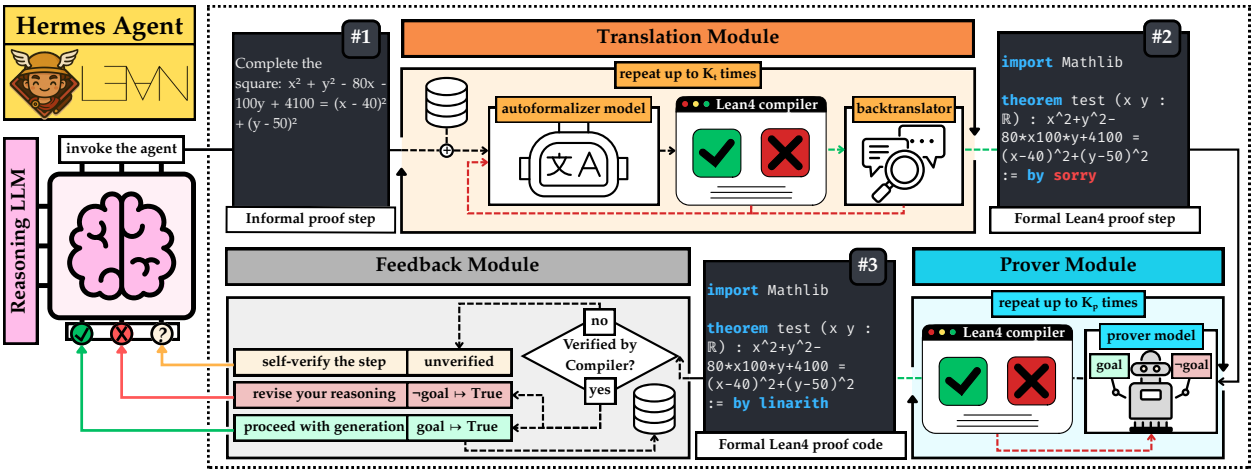


Figure 2: Full *Hermes* framework with illustrative examples.

3 Our Approach

In this section, we describe *Hermes*, our framework for a verifiable and interpretable tool-based agent for mathematical reasoning. *Hermes* is designed to be split into multiple swappable modules that analyze intermediate proof steps and produce proper verification for each mathematical step. Figure 2 illustrates the complete agent pipeline, including intermediate proof transformations after each module. At the core of our agent lies the tool-calling capability of modern LLMs, which allows the model to pause generation and query external agents to verify reasoning steps or retrieve additional context. In our design, the LLM is instructed to verify every critical mathematical step contributing to the final proof.

Our Lean backend of choice is the Lean4 REPL [44], which provides compatibility with Python-based tool integration. We also employ the verification scheduler, which parallelizes Lean code verification for faster inference. The REPL validates proofs generated by the prover module, returning a simple `True/False` signal that indicates the correctness of the intermediate proof step. Additionally, our memory block uses embedding-based retrieval and ranking to extract the most relevant contextual information. The following subsections describe each *Hermes* module in detail.

3.1 Translation Module

The translation Module is responsible for accurately translating each proof step expressed in natural language into its equivalent Lean4 statement. At this stage, we employ a dedicated Lean autoformalizer model that generates a Lean statement corresponding to the informal proof step, inserting a `sorry` placeholder. To ensure the correctness and semantic relevance of the translation, we apply two-step verification. First, the translated statement must successfully compile in Lean. Second, the validated Lean code is backtranslated into informal language, and the LLM evaluates the equivalence between the original and backtranslated proof steps. If the two are deemed equivalent, the translation module forwards the Lean statement to the prover module. The number of attempts to obtain a valid translation is controlled by the parameter K_t, which defines the translation budget. A higher K_t typically yields more accurate translations but may increase the agent's runtime. This hyperparameter is user-controlled. This module is highlighted in orange in Figure 2.

3.2 Prover Module

The prover Module receives a valid Lean statement and attempts to find a corresponding proof within Lean. At this stage, we incorporate a whole-proof generation model together with built-in Lean solvers;

however, any proof-generation method, such as tree-based provers or premise-selection theorem provers, is compatible with our agent. This module attempts two proofs in parallel: the original goal and its negation, ¬goal. The objective is to either prove the original statement or derive a counter-proof (e.g., a counterexample) to demonstrate the invalidity of the step. Similar to the Translation Module, the prover operates under a sampling budget controlled by the parameter K_p. We observe that increasing K_p improves performance but incurs additional inference cost, as the prover must explore more proof attempts before reaching a conclusion. Once the prover finds a proof or counter-proof, it returns this verification signal to the LLM to guide the subsequent reasoning chain. This module is highlighted in blue in Figure 2.

3.3 Memory Block

Memory Block is responsible for collecting all validated proof steps in a structured database. Each proof step is stored alongside its vector representation to enable fast, similarity-based retrieval. Because some problems involve long chains of thought (CoT) with branching, non-overlapping reasoning paths, we implement a top-k retrieval mechanism to select the most relevant memories for the currently evaluated proof step. Furthermore, to maintain proof continuity and prevent the model from pursuing irrelevant or disjoint reasoning paths, the retrieved memories are passed as additional context to the translation module. These are incorporated into the translated Lean statement as preliminary hypotheses, ensuring logical consistency across proof steps. The memory block corresponds to the "database image" in Figure 2.

3.4 Lean4 REPL Feedback

Our agent returns one of three states: (1) proof succeeded, (2) counter-proof found, or (3) verification failed. The first two states drive the next-generation strategy: the LLM either revises its reasoning when hallucination is detected or continues exploring already-verified steps. The third state arises when the autoformalizer fails to produce a valid translation, or the prover cannot prove/disprove the goal. The design is compatible with any autoformalizer–prover pairing, so as research progresses, stronger components can be swapped in seamlessly. Given feedback, the LLM then chooses to (1) continue exploring the current line of reasoning, (2) attempt an alternative approach, or (3) proceed cautiously and self-verify reasoning, since the step did not pass the Lean4 compiler.

4 Experiments

4.1 Experimental Setup

***Hermes* setup.** For *Hermes* evaluation, we set both K_p and K_t to 4, which strikes a good balance between inference speed and downstream Lean verification accuracy. Lean timeout is set to 60 seconds. The memory module selects the top three recorded steps as context. The prover of choice is Goedel-Prover-V2-8B [20] and the autoformalizer model is Goedel-Autoformalizer-8B [20], unless stated otherwise. Chosen embedding model for the memory block is Qwen3-Embedding-0.6B [45]. The Lean version matches the one used during model training - `v4.9.0`.

Baseline methods. We compare our approach against several strong baselines, with Safe [33] serving as the previous state-of-the-art method. We further introduce Safe*, an improved variant that replaces the prover and autoformalizer in Safe with more advanced model Goedel-Prover-V2 [20] and Goedel-Autoformalizer [20] to match our *Hermes* setup. In addition, we include ZS-CoT, a zero-shot chain-of-thought baseline, and Majority@5, which selects the most frequent answer from five sampled reasoning traces. We also include a range of outcome and process reward models. The outcome reward models include Skywork-Reward-Llama-3.1-8Bv0.2 [46] and ArmoRM-Llama3-8Bv0.1 [47], while the process reward models include math-shepherd-mistral-7b-prm [10] and RLHFlow/Llama3.1-8B-PRMDeepseek-Data [48]. All experiments are conducted on three base models: Qwen3-8B [49], OpenAI o3-mini [50], and DeepSeek-V3.1 [51].

Table 1: Accuracy (%) of different reasoning models under four inference strategies: zero-shot CoT (@1), majority vote (@5), reward-model selection (Best-of-5), and *Hermes* (@1). Results are reported on four benchmarks: MATH500 (MATH), AIME'25 (AIME), CollegeMath (CM), and HARDMath2 (HM2).

	Qwen3-8B				**OpenAI o3-mini**				**DeepSeek-V3.1**			
	MATH	AIME	CM	HM2	MATH	AIME	CM	HM2	MATH	AIME	CM	HM2
ZS-CoT@1	84.8	20.0	69.1	4.3	95.8	63.3	75.2	23.2	94.8	46.7	78.0	22.7
Majority@5	87.0	16.7	70.3	4.7	96.8	70.0	76.0	22.7	96.6	40.0	78.5	25.6
Skywork	91.0	**30.0**	72.0	5.7	96.8	83.3	76.1	29.4	96.6	53.3	80.2	28.4
ArmoRM	88.6	**30.0**	72.4	5.2	96.2	76.7	76.1	28.4	95.6	56.7	80.5	26.5
Shepherd	87.8	23.3	70.2	5.7	96.4	80.0	75.7	25.6	96.2	46.7	79.0	28.0
RLHFlow	84.0	20.0	69.4	5.7	95.8	70.0	75.9	28.4	95.4	50.0	78.7	25.6
Lean-based methods												
Safe	89.4	23.3	72.4	5.7	96.0	83.3	75.7	26.1	96.2	46.7	80.9	27.5
Safe*	89.4	23.3	72.5	6.2	96.8	83.3	75.8	25.6	96.6	53.3	81.0	27.5
Hermes	**91.2**	**30.0**	**73.0**	**6.6**	**97.2**	**86.7**	**78.9**	**31.3**	**97.4**	**66.7**	**83.3**	**30.3**

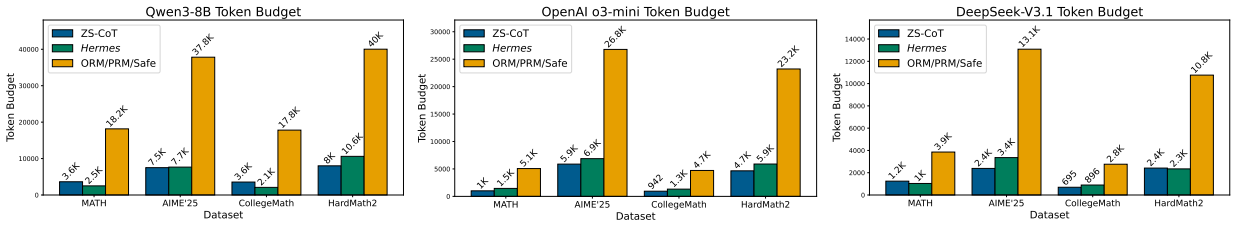
Datasets and evaluation. We conduct experiments on four widely used math benchmarks: MATH-500 [52], AIME'25 [53], CollegeMath [54], and HARDMath2 [55]. For brevity, we refer to them as MATH, AIME, CM, and HM2, respectively, throughout the experiments. Each problem has a 15-minute time limit and an 8,192-token budget (prompt + generation). All *Hermes* accuracy reports are @1, i.e., the reasoning models were given only one attempt per problem. For score-based methods, we evaluate each problem with N generated candidates (CoTs) and report Best-of-N (BoN). BoN accuracy is computed based on the reasoning trace selected by the reward model from the N candidates. Unless stated otherwise, N is set to 5 reasoning traces, consistent with the configuration adopted in Safe [33].

4.2 Performance of *Hermes* against base LLMs and reward-based generation

Table 1 reports the accuracy of eight baseline inference methods and our proposed *Hermes* across three reasoning models and four benchmark datasets, with each method evaluated on every combination of model and dataset. We selected representative reasoning models from three distinct parameter scales: a sub-10B model (Qwen3-8B), a medium-sized model (o3-mini), and a large model (DeepSeek-V3.1), each from a distinct provider. Compared with non–Lean-based methods, our *Hermes* consistently outperforms all baselines across models and benchmarks, except on the AIME benchmark with Qwen3-8B, where Skywork and ArmoRM achieve comparable performance. Specifically, averaged across all reasoning models and benchmarks, *Hermes* surpasses zero-shot CoT, Majority@5, ORM-based Skywork and ArmoRM, and PRM-based Shepherd and RLHFlow methods by 23.4%, 23.9%, 5.7%, 8%, 12.2%, and 14.9%, respectively. Compared with the Lean-based baselines Safe and Safe*, *Hermes* achieves consistently higher accuracy across all benchmarks, exceeding them by an average of 11.2% and 9%, respectively. This demonstrates *Hermes*'s more effective integration of neural reasoning and symbolic verification. We also observe that on more challenging datasets, *Hermes* achieves the largest relative improvements. For instance, on the harder dataset HM2, Deepseek-V3.1+*Hermes* improves performance by 33.5%, whereas on the easier MATH dataset, the gain is only 2.7% compared to ZS-CoT. This suggests that *Hermes* is particularly effective at correcting errors in settings that require more advanced mathematical reasoning.

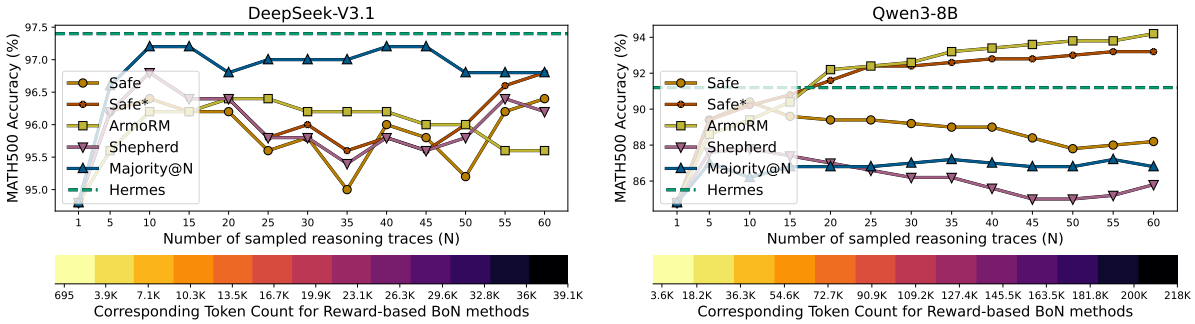
4.3 Token budgets and generation efficiency

As shown in Figure 3, our evaluation indicates that the reasoning token budget of *Hermes* is comparable to that of zero-shot chain-of-thought, while being 4–6 times lower than score-based methods such as Safe, ORMs, and PRMs. These results demonstrate that incorporating intermediate, verifiable feedback not only improves accuracy across a range of benchmarks but also significantly reduces token budgets compared to



(a) Qwen3-8B (b) OpenAI o3-mini (c) DeepSeek-V3.1

Figure 3: Average reasoning token usage per problem on MATH500, AIME'25, CollegeMath, and HardMath2 under Zero-Shot Chain-of-Thought, *Hermes*, and Reward-based Best-of-5 settings.



(a) DeepSeek-V3.1 (b) Qwen3-8B

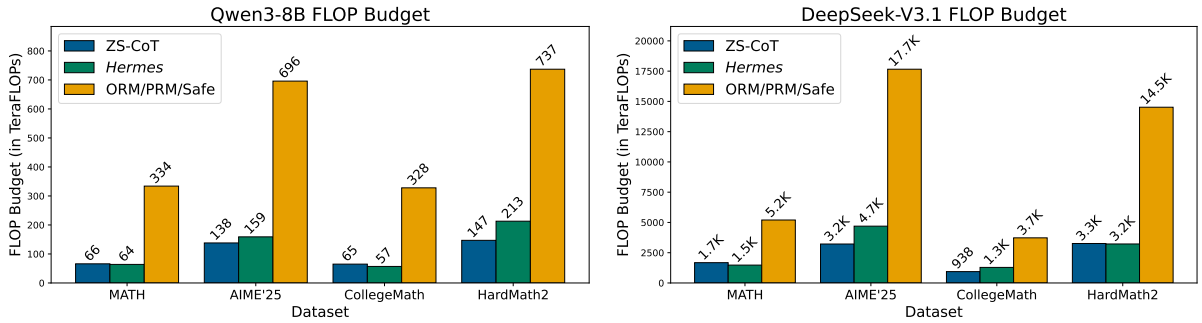
Figure 4: Scaling behavior of BoN across ORM, PRM, Safe and Majority vote. The green dashed line indicates *Hermes* performance at @1. The corresponding token consumption for each BoN is shown below as a one-dimensional heatmap.

reward-model–graded reasoning trace selection.

To further investigate the performance of score-based methods, we conducted an additional experiment in which we increased the number of sampled reasoning traces from 1 (zero-shot chain-of-thought) to 60. Figure 4 reports BoN accuracy across four inference methods: Safe, Safe*, math-shepherd-mistral-7b-prm, and ArmoRM-Llama3-8Bv0.1. *Hermes* is fixed at @1 and is shown as a dashed green line. Using DeepSeek-V3.1 as the base reasoning model, we observe that scaling N does not yield significant gains in accuracy. All reward-based methods underperform compared to *Hermes*, while consuming a linearly increasing number of tokens. This suggests that, rather than repeatedly regenerating reasoning traces, it is more effective to incorporate higher-quality intermediate verification and guidance mechanisms that enhance reasoning quality without incurring additional token costs. On the other hand, the same experiment on Qwen3-8B shows that reward-based models can outperform *Hermes* when given a sufficiently large token budget. We observe that for $n \geq 20$, both Safe* and ArmoRM exceed 92% accuracy; however, they consume roughly 28 times more tokens than *Hermes*. We hypothesize that reward-based BoN is more effective when the quality of reasoning traces exhibits higher variance, as is the case for smaller models such as Qwen3-8B.

4.4 Computational Efficiency Analysis

Following the methodology of [56], we further examine the computational efficiency of *Hermes* in comparison to CoT and reward-based approaches. Because our agent requires communication among the reasoning, translation, and prover models, we estimate the total computational cost in terms of FLOPs for two representative models: Qwen3-8B and DeepSeek-V3.1. OpenAI o3-mini is excluded from this analysis, as



(a) Qwen3-8B

(b) DeepSeek-V3.1

Figure 5: Average TeraFLOPs per problem on MATH500, AIME'25, CollegeMath, and HardMath2 under Zero-Shot Chain-of-Thought, *Hermes*, and Reward-Based Best-of-5 settings.

its architecture and parameter count are not publicly available. We approximate the FLOP budget using the formula $C_{forward} = 2N + 2n_{layer}n_{ctx}d_{attn}$, where N denotes the number of model parameters, n_{layer} the number of layers, n_{ctx} the input context length (set to the maximum of 8,192 tokens in our experiments), and d_{attn} the dimensionality of the attention output. Values for N, n_{layer}, and d_{attn} were derived from each model's configuration files. As shown in Figure 5, even after accounting for the additional computation required by the *Hermes* agent, its total cost remains comparable to zero-shot CoT and substantially lower than that of reward-based models. These results demonstrate that, despite involving external translation and theorem-proving modules, *Hermes* maintains strong computational efficiency and effective resource utilization.

4.5 Ablation studies on different *Hermes* modules

Recalling Section 3, *Hermes* is designed as a multi-modular agent. In this section, we conduct an ablation study to evaluate the contribution of its two key modules: the memory buffer and the prover module. The translation and feedback modules are not ablated, as they serve as essential intermediaries bridging informal and formal mathematical reasoning. When ablating the memory buffer, *Hermes* no longer stores intermediate proof steps, preventing both the translation module and prover module from accessing previously validated information necessary for maintaining coherent reasoning across steps. When ablating the prover module, the agent omits a query to the dedicated prover model and instead relies solely on the built-in Lean solvers, similar to the AutoSolver module described in [28].

Table 2 shows the results on the MATH and AIME benchmarks using DeepSeek-V3.1 as the reasoning model. Overall, removing either module leads to a clear drop in accuracy across both benchmarks. Specifically, the prover model has a stronger impact on overall performance, as removing it reduces accuracy from 97.4% to 93.0% on MATH and from 66.7% to 50.0% on AIME. It is because the prover offers more reliable symbolic verification of complex dependencies and intermediate lemmas, thereby offering more accurate feedback to the reasoning model and preventing the accumulation of logical errors across reasoning steps. Removing the memory buffer yields a similar reduction, from 97.4% to 97.0% on MATH and from 66.7% to 60.0% on AIME, due to the loss of access to previously validated steps, which in turn disrupts reasoning continuity. Moreover, the ablation is more sensitive on the more challenging AIME dataset than on the easier MATH benchmark, with a relative performance drop of 25.0% and 4.5%, respectively, when both modules are removed compared to the full *Hermes* configuration. This sensitivity arises because AIME problems typically require longer reasoning traces, deeper inter-dependencies between intermediate steps, and a larger reasoning budget. As the reasoning trace extends, the process becomes increasingly unstable, leading to error propagation. These results indicate that the advantages of *Hermes*'s modular design become more pronounced as reasoning complexity increases.

Table 2: Ablation study of different sub-modules of *Hermes* . The checkmark (✓) denotes the inclusion of a module, while the cross (✗) indicates its removal.

Memory Buffer	Prover Model	Accuracy (%) MATH	Accuracy (%) AIME
✗	✗	93.0	50.0
✗	✓	97.0	60.0
✓	✗	93.0	50.0
✓	✓	**97.4**	**66.7**

Table 3: Accuracy (%) of *Hermes* on HM2 with a combination of various autoformalizer sampling budget (K_t) and prover sampling budget (K_p).

		Prover (@K_p)			
		@1	@4	@8	@16
Autoformalizer (@K_t)	@1	15.6	16.6	15.2	16.6
	@4	23.7	30.3	30.8	31.8
	@8	23.2	30.8	31.8	33.7

4.6 Sensitivity analysis on different values of K_p and K_t parameters

Table 3 illustrates the sensitivity of *Hermes* to sampling hyperparameters. We vary the autoformalizer sampling budget (K_t) from 1 to 8 and prover sampling budget (K_p) from 1 to 16, which is the number of translation and proof verification attempts, respectively. In this experiment, we use DeepSeek-V3.1 as the reasoning model and conduct evaluations on the HM2 dataset. Overall, increasing either K_t or K_p, or both jointly, leads to improved performance. Specifically, as shown in Table 3, increasing K_t from 1 to 4 yields an average accuracy improvement of 82.2% across tested prover budgets, and further increasing K_t from 4 to 8 results in 2.2% gain. It is because the translator occasionally introduces minor errors that fail Lean verification and backtranslation, and a higher sampling budget increases the likelihood of generating correct formalizations. For K_p, accuracy improves substantially by 22.3% when increasing the budget from 1 to 4, as multiple prover attempts enable exploration of alternative proof trajectories and improve the likelihood of successful verification. Beyond K_p=8, the gain becomes marginal. These results suggest that moderate autoformalizer and prover budgets are sufficient for reliable verification without incurring excessive computational cost. We thus set K_t=4 and K_p=4 as the default values in our experiments. Additional ablation results are provided in Section B of the Appendix.

5 Conclusion

In this work, we introduced *Hermes*, a Lean4-powered, tool-augmented agent for advanced mathematical reasoning. The framework combines the flexibility of informal problem solving with the rigor of formal verification by validating intermediate steps that would otherwise be prone to hallucinations or logical errors. Built on state-of-the-art Lean4 theorem-proving and autoformalization models, *Hermes* incorporates a dedicated memory mechanism that preserves proof continuity across long reasoning chains and guides the reasoning LLM towards a correct solution. Experimental results show that *Hermes* achieves higher accuracy on four challenging mathematical benchmarks while consuming substantially fewer inference resources than reward-based approaches. On average, *Hermes* improves accuracy by 14% while using at least 4× fewer reasoning tokens. This improvement stems from the synergistic interaction between its memory module, back-translator, and Lean4's native solvers. Overall, *Hermes* represents a step toward unifying informal and formal mathematical reasoning within a scalable, verifiable, interpretable, and modular framework.

References

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[2] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc., 2022.

[3] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *ArXiv preprint*, abs/2309.12284, 2023.

[4] Xin Xu, Tong Xiao, Zitong Chao, Zhenya Huang, Can Yang, and Yang Wang. Can llms solve longer math word problems better? *arXiv preprint arXiv:2405.14804*, 2024.

[5] Yichen Huang and Lin F. Yang. Winning gold at imo 2025 with a model-agnostic verification-and-refinement pipeline, 2025.

[6] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2), January 2025.

[7] Haolang Lu, Yilian Liu, Jingxin Xu, Guoshun Nan, Yuanlong Yu, Zhican Chen, and Kun Wang. Auditing meta-cognitive hallucinations in reasoning large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[8] Shayan Ali Akbar, Md Mosharaf Hossain, Tess Wood, Si-Chi Chin, Erica M Salinas, Victor Alvarez, and Erwin Cornejo. HalluMeasure: Fine-grained hallucination measurement using chain-of-thought reasoning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15020–15037, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[9] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

[10] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9426–9439. Association for Computational Linguistics, 2024.

[11] Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. R-PRM: reasoning-driven process reward modeling. *CoRR*, abs/2503.21295, 2025.

[12] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516, Vienna, Austria, July 2025. Association for Computational Linguistics.

[13] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.

[14] 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*, page 625–635, Berlin, Heidelberg, 2021. Springer-Verlag.

[15] Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

[16] Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283. Springer Science & Business Media, 2002.

[17] Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin Lauter, Swarat Chaudhuri, and Dawn Song. Formal mathematical reasoning: A new frontier in AI. In *Proceedings of the International Conference on Machine Learning*, 2025.

[18] Google DeepMind. AI achieves silver-medal standard solving international mathematical olympiad problems. `https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/`, 2024. Accessed: 2025-05-08.

[19] Trieu Trinh, Yuhuai Wu, Quoc Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 2024.

[20] 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. *CoRR*, abs/2508.03613, 2025.

[21] 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, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *ArXiv*, 2025.

[22] Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Ying Zhang, Zhu Zekai, Lu Jian Qiao, Hugues de Saxcéand, Thibaut Barroyer, Ebony Zhang, Bolton Bailey, Frederick Pu, Jonas Bayer, Marina Vinyes, Zhengying Liu, Li Jia, and the AI-MO team. Kimina-prover: Applying test-time rl search on large formal reasoning models. `https://huggingface.co/blog/AI-MO/kimina-prover`, July 2025. Published July 10, 2025.

[23] Huajian Xin, Z.Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Haowei Zhang, 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. In *The Thirteenth International Conference on Learning Representations*, 2025.

[24] Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, Cheng Ren, Jiawei Shen, Wenlei Shi, Tong Sun, He Sun, Jiahui Wang, Siran Wang, Zhihong Wang, Chenrui Wei, Shufa Wei, Yonghui Wu, Yuchen Wu, Yihang Xia, Huajian Xin, Fan Yang, Huaiyuan Ying, Hongyi Yuan, Zheng Yuan, Tianyang Zhan, Chi Zhang, Yue Zhang, Ge Zhang, Tianyun Zhao, Jianqiu Zhao, Yichi Zhou, and Thomas Hanwen Zhu. Seed-prover: Deep and broad reasoning for automated theorem proving. *CoRR*, abs/2507.23726, 2025.

[25] Xuejun Yu, Jianyuan Zhong, Zijin Feng, Pengyi Zhai, Roozbeh Yousefzadeh, Wei Chong Ng, Haoxiong Liu, Ziyi Shou, Jing Xiong, Yudong Zhou, Claudia Ong, Austen Jeremy Sugiarto, Yaoxi Zhang, Wai Ming Tai, Huan Cao, Dongcai Lu, Jiacheng Sun, Qiang Xu, Shen Xin, and Zhenguo Li. Mathesis: Towards formal theorem proving from natural languages. *CoRR*, abs/2506.07047, 2025.

[26] Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. Herald: A natural language annotated lean 4 dataset. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[27] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: a cross-system benchmark for formal Olympiad-level mathematics, 2022.

[28] Azim Ospanov, Farzan Farnia, and Roozbeh Yousefzadeh. Apollo: Automated llm and lean collaboration for advanced formal reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[29] Talia Ringer. *Proof Repair*. PhD thesis, University of Washington, 2021. Ph.D. dissertation.

[30] Albert Qiaochu Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygóźdź, Piotr Miłoś, Yuhuai Wu, and Mateja Jamnik. Thor: Wielding hammers to integrate language models and automated theorem provers. In *Advances in Neural Information Processing Systems*, 2022.

[31] Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-proof generation and repair with large language models, 2023.

[32] Roozbeh Yousefzadeh, Xuenan Cao, and Azim Ospanov. A Lean dataset for International Math Olympiad: Small steps towards writing math proofs for hard problems. *Transactions on Machine Learning Research*, 2025.

[33] Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. Safe: Enhancing mathematical reasoning in large language models via retrospective step-aware formal verification. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12171–12186, Vienna, Austria, July 2025. Association for Computational Linguistics.

[34] Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-rong Wen. Tool learning with large language models: a survey. *Frontiers of Computer Science*, 19(8), January 2025.

[35] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023.

[36] Silin Gao, Jane Dwivedi-Yu, Ping Yu, Xiaoqing Ellen Tan, Ramakanth Pasunuru, Olga Golovneva, Koustuv Sinha, Asli Celikyilmaz, Antoine Bosselut, and Tianlu Wang. Efficient tool use with chain-of-abstraction reasoning, 2025.

[37] Kangyun Ning, Yisong Su, Xueqiang Lv, Yuanzhe Zhang, Jian Liu, Kang Liu, and Jinan Xu. Wtu-eval: A whether-or-not tool usage evaluation benchmark for large language models, 2024.

[38] Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-LM: empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the 2023 Conference on Empirical Methods in Natural Language Processing (Findings of EMNLP)*, Singapore, Dec 2023.

[39] Shashank Kirtania, Priyanshu Gupta, and Arjun Radhakrishna. LOGIC-LM++: Multi-step refinement for symbolic formulations. In Bhavana Dalvi Mishra, Greg Durrett, Peter Jansen, Ben Lipkin, Danilo Neves Ribeiro, Lionel Wong, Xi Ye, and Wenting Zhao, editors, *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024)*, pages 56–63, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[40] Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timoth'ee Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, Sketch, and Prove: Guiding formal theorem provers with informal proofs. In *International Conference on Learning Representations*, 2023.

[41] Chenrui Cao, Liangcheng Song, Zenan Li, Xinyi Le, Xian Zhang, Hui Xue, and Fan Yang. Reviving dsp for advanced theorem proving in the era of reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[42] Haohan Lin, Zhiqing Sun, Yiming Yang, and Sean Welleck. 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, 2025.

[43] Jin Peng Zhou, Charles E Staats, Wenda Li, Christian Szegedy, Kilian Q Weinberger, and Yuhuai Wu. Don't trust: Verify–grounding llm quantitative reasoning with autoformalization. In *The Twelfth International Conference on Learning Representations*, 2023.

[44] Lean FRO. A read-eval-print-loop for Lean 4. `https://github.com/leanprover-community/repl`, 2023.

[45] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

[46] Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *CoRR*, abs/2410.18451, 2024.

[47] Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 10582–10592. Association for Computational Linguistics, 2024.

[48] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.

[49] Qwen Team. Qwen3 technical report, 2025.

[50] OpenAI. Openai o3-mini. `https://openai.com/index/openai-o3-mini/`, January 2025.

[51] DeepSeek-AI et al. Deepseek-v3 technical report, 2025.

[52] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

[53] American Mathematics Competitions and Art of Problem Solving. Aime problems and solutions — 2025. `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`, 2025.

[54] Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. Mathscale: Scaling instruction tuning for mathematical reasoning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[55] James V. Roggeveen, Erik Y. Wang, Will Flintoft, Peter Donets, Lucy S. Nathwani, Nickholas Gutierrez, David Ettel, Anton Marius Graf, Siddharth Dandavate, Arjun Nageswaran, Raglan Ward, Ava Williamson, Anne Mykland, Kacper K. Migacz, Yijun Wang, Egemen Bostan, Duy Thuc Nguyen, Zhe He, Marc L. Descoteaux, Felix Yeung, Shida Liu, Jorge García Ponce, Luke Zhu, Yuyang Chen, Ekaterina S. Ivshina, Miguel Fernandez, Minjae Kim, Kennan Gumbs, Matthew Scott Tan, Russell Yang, Mai Hoang, David Brown, Isabella A. Silveira, Lavon Sykes, Ahmed Roman, William Fredenberg, Yiming Chen, Lucas Martin, Yixing Tang, Kelly Werker Smith, Hongyu Liao, Logan G. Wilson, Alexander Dazhen Cai, Andrea Elizabeth Biju, and Michael P. Brenner. Hardmath2: A benchmark for applied mathematics built by students as part of a graduate class. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[56] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Appendices

A Prompt and instruction templates

```
# Hermes Tool Instruction
Formally validates a **single** reasoning step using a formal Lean4 verifier. Invoke
    this function when facing a step that potentially involves a hallucination. Make
    sure to verify **every** critical mathematical proof step.

Args:
    proof_step (str): A proof step that includes both the goal to be proven and the
    relevant context (e.g., variables, assumptions, and previously proven statements).
    Always explicitly specify the relevant context, such as domains, data types, and any
    other necessary details. Make sure to state the proof step in English.

Returns:
    str: A status string indicating the verification result:
        - **CORRECT**: Step verified by Lean 4.
        - **INCORRECT**: Step rejected by Lean 4 (e.g., the prover proved a
    contradiction or the opposite statement).
        - **VERIFICATION FAILURE**: Step could not be verified by Lean 4 (e.g., the
    prover was unable to prove the statement or find contradictory arguments)
        - **NO VERIFICATION**: Step skipped (e.g., non-formalizable, non-mathematical,
    or missing required definitions).

Notes:
    - Treat **CORRECT** steps as reliable within the given formalization.
    - Treat **INCORRECT** steps as requiring revision; a suggested correction is
    returned with this label.
    - Use **VERIFICATION FAILURE** to indicate inconclusive or ill-formed steps that
    Lean 4 was unable to prove or disprove.
    - Use **NO VERIFICATION** for instructional text or incomplete definitions.
```

```
# Answer verification prompt
Given a question, an answer to that question, and the ground truth for that question's
    answer, you need to check if the given answer matches the ground truth.
* If the answer is complete and correct, simply reply with True.
* If the given answer does not match the ground truth or is incomplete, reply with False.
* Do **NOT** respond with any other characters.

Question:
<question>

Answer:
<answer>

Ground Truth:
<ground_truth>

Does the answer match the ground truth? (True or False):
```

````
# NL->FL translation prompt for Goedel-Formalizer-V2-8B
Please autoformalize the following natural language problem statement in Lean 4.
Use the following theorem name: test
The natural language statement is:
<question>
Think before you provide the lean statement.
````

````
# Theorem proving prompt for Goedel-Prover-V2-8B
Complete the following Lean 4 code:

```lean4
<header>

<body>```

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

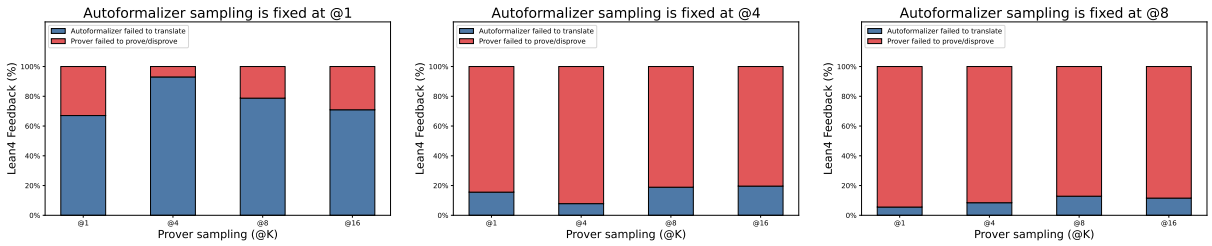
````
# NL->FL translation prompt for Kimina-Autoformalizer-7B
Please autoformalize the following problem in Lean 4 with a header. Use the following
    theorem names: my_favorite_theorem.

<question>
````

````
# Theorem proving prompt for Kimina-Prover-RL-1.7B
Think about and solve the following problem step by step in Lean 4.
# Formal statement:
```lean4
<header>

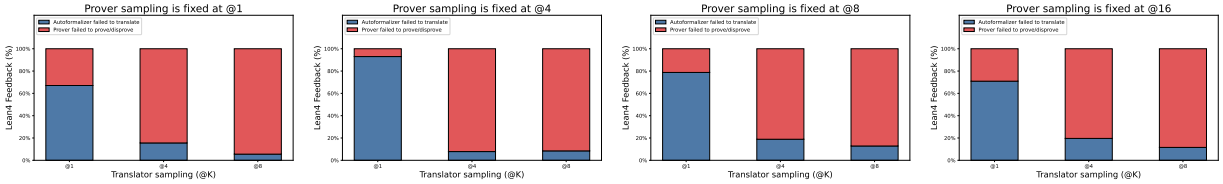
<body>
```
````

The prompt instructions and templates are shown above. For the *Hermes* prompt, we follow the LangChain community guidelines. The tool is named `verify_one_mathematical_step` to encourage the model to verify only a single step rather than an entire proof. The tool description is flexible and can be modified; in particular, adjusting the description can change how frequently the tool is invoked. The answer-verification prompt is adapted from [33]. For the autoformalizer and prover prompts, we use the officially released versions provided by the original developers.
````



(a) Translator @1 (b) Translator @4 (c) Translator @8

Figure 6: Distribution of *Hermes* failures with a fixed translation budget (K_t).



(a) Prover @1 (b) Prover @4 (c) Prover @8 (d) Prover @16

Figure 7: Distribution of *Hermes* failures with a fixed prover budget (K_p).

B Additional ablation on different combinations of autoformalizers and provers

Following the ablation studies presented in Table 3 of the main text, we provide additional insights into the selection of K_t and K_p. Figures 6 and 7 illustrate the proportion of translator and prover errors, respectively, each with either the translation or prover budget held fixed. We observe that at @1, translation failures are overwhelmingly frequent, exceeding 60%. As the translation budget K_t increases, such failures become rare, contributing to 10% or less of all Lean verification errors. Conversely, when $K_p = 1$, the prover becomes highly unreliable, with unresolved proof steps accounting for up to 95% of all verification failures. Increasing K_p to 16 yields diminishing returns, as theorem proving is inherently more complex than autoformalization, requiring a substantially larger budget for high-quality proof generation. However, given that *Hermes* was evaluated under moderate computational constraints, our focus remained on efficient inference and agent interaction. Our results align with prior work, where whole-proof generation models typically employ very large sampling budgets, ranging from 1,024 to 25,600 samples. We hypothesize that in large-scale settings, increasing K_p beyond 16 may further reduce prover failure rates.

[MATH500] Precalculus - 768

Find all solutions to $\sin\left(\tan^{-1}(x) + \cot^{-1}\left(\frac{1}{x}\right)\right) = \frac{1}{3}$. Enter all the solutions, separated by commas.

...
Case 1: \(x > 0\)
If \(x > 0\), then \(y = \frac{1}{x} > 0\). Therefore, the identity holds:
\[\cot^{-1}\left(\frac{1}{x} \right) = \tan^{-1}(x) \]

...

Thus, both solutions are valid.

Case 2: \(x < 0\)
If \(x < 0\), then \(y = \frac{1}{x} < 0\). The identity \(\cot^{-1}(y) = \tan^{-1}\left(\frac{1}{y} \right)\) doesn't hold directly.

I recall that for \(y < 0\), the following identity is true:
\[\cot^{-1}(y) = \pi + \tan^{-1}\left(\frac{1}{y} \right) \]

But, I'm not sure if this is accurate. Let me verify.

Looking at the range of \(\cot^{-1}(y)\):
- The range of \(\cot^{-1}(y)\) is \((0, \pi)\).
- For \(y > 0\), \(\cot^{-1}(y) = \tan^{-1}(1/y)\).
- For \(y < 0\), \(\cot^{-1}(y) = \pi + \tan^{-1}(1/y)\).

But, \(\tan^{-1}(1/y)\) for \(y < 0\) will give a value between \(-\pi/2\) and \(0\). Adding \(\pi\) to it gives a value between \(\pi/2\) and \(\pi\), which is within the range of \(\cot^{-1}(y)\).

Thus, for \(x < 0\):
\[\cot^{-1}\left(\frac{1}{x} \right) = \pi + \tan^{-1}(x) \]

Now, the expression inside the sine becomes:
\[\tan^{-1}(x) + \cot^{-1}\left(\frac{1}{x} \right) = \tan^{-1}(x) + \pi + \tan^{-1}(x) = \pi + 2\tan^{-1}(x) \]

Thus, the equation is:
\[\sin(\pi + 2 \tan^{-1}(x)) = \frac{1}{3} \]

I know that \(\sin(\pi + \theta) = -\sin(\theta)\). Therefore:
\[\sin(\pi + 2 \tan^{-1}(x)) = -\sin(2 \tan^{-1}(x)) = \frac{1}{3} \]

But from the previous case, \(\sin(2 \tan^{-1}(x)) = \frac{2x}{1 + x^2}\). Therefore:
\[-\frac{2x}{1 + x^2} = \frac{1}{3} \]

Simplify:
\[\frac{-2x}{1 + x^2} = \frac{1}{3} \]
\[-6x = 1 + x^2 \]
\[x^2 + 6x + 1 = 0 \]

The solutions are:
\[x = \frac{-6 \pm \sqrt{36 - 4}}{2} = \frac{-6 \pm \sqrt{32}}{2} = \frac{-6 \pm 4\sqrt{2}}{2} = -3 \pm 2\sqrt{2} \]

Now, since \(x < 0\) in this case, we need to check both solutions:
1. \(x = -3 + 2\sqrt{2} \approx -3 + 2.828 = -0.172 < 0\)
2. \(x = -3 - 2\sqrt{2} \approx -3 - 2.828 = -5.828 < 0\)

Both solutions are negative, so both are valid.
...
Summary of Solutions
From both cases, we have four potential solutions:
1. \(x = 3 + 2\sqrt{2}\)
2. \(x = 3 - 2\sqrt{2}\)
3. \(x = -3 + 2\sqrt{2}\)
4. \(x = -3 - 2\sqrt{2}\)

All of them satisfy the original equation, as verified.

Final Answer
The solutions are \(\boxed{3 + 2\sqrt{2}}\), \(\boxed{3 - 2\sqrt{2}}\), \(\boxed{-3 + 2\sqrt{2}}\), and \(\boxed{-3 - 2\sqrt{2}}\).

...

First, I notice that $\tan^{-1}(x)$ and $\cot^{-1}\left(\frac{1}{x}\right)$ are related. Let me verify if there's a relationship between these two inverse trigonometric functions.

Lean agent verified proof steps:
1. *For $x > 0$, $\theta = \tan^{-1}(x)$ implies $\tan(\theta) = x$, so $\cot(\theta) = 1/x$, which means $\theta = \cot^{-1}(1/x)$. Therefore, $\tan^{-1}(x) = \cot^{-1}(1/x)$ for $x > 0$.*
2. *For $x = 0$, $\tan^{-1}(0) = 0$ and $\cot^{-1}(1/0)$ is undefined since division by zero is not allowed. Therefore, $x = 0$ is not in the domain.*

Let me continue the analysis:

From the verification, I can see that for $x \neq 0$, $\tan^{-1}(x) = \cot^{-1}(1/x)$. This means the expression simplifies to:

$\sin(2\tan^{-1}(x)) = 1/3$

Let me continue the mathematical reasoning:

Let $\theta = \tan^{-1}(x)$, so $x = \tan(\theta)$. Then we have:
$\sin(2\theta) = 1/3$

Using the identity $\sin(2\theta) = 2\tan(\theta)/(1 + \tan^2(\theta)) = 2x/(1 + x^2)$

So the equation becomes:
$2x/(1 + x^2) = 1/3$

Solving this:
$6x = 1 + x^2$
$x^2 - 6x + 1 = 0$

Using the quadratic formula:
$x = [6 \pm \sqrt{(36 - 4)}]/2 = [6 \pm \sqrt{32}]/2 = [6 \pm 4\sqrt{2}]/2 = 3 \pm 2\sqrt{2}$

Now I need to check if both solutions are valid. Let me verify if they satisfy the original equation:

Lean verified proof steps:
1. *For $x = 3 + 2\sqrt{2}$, $\tan^{-1}(x)$ is defined since $x > 0$, and $\cot^{-1}(1/x)$ is also defined since $1/x > 0$. The sum $\tan^{-1}(x) + \cot^{-1}(1/x) = 2\tan^{-1}(x)$ is valid, and $\sin(2\tan^{-1}(x)) = 2x/(1+x^2) = 1/3$, so this is a valid solution.*
2. *For $x = 3 - 2\sqrt{2}$, since $3 - 2\sqrt{2} \approx 3 - 2.828 = 0.172 > 0$, $\tan^{-1}(x)$ is defined and $\cot^{-1}(1/x)$ is also defined. The sum $\tan^{-1}(x) + \cot^{-1}(1/x) = 2\tan^{-1}(x)$ is valid, and $\sin(2\tan^{-1}(x)) = 2x/(1+x^2) = 1/3$, so this is also a valid solution.*

Therefore, both solutions are valid.

Final Answer: $3 + 2\sqrt{2}, 3 - 2\sqrt{2}$

Figure 8: Illustration of *Hermes*'s effect on chain-of-thought generation: the model attempts to solve the precalculus-768 problem from the MATH500 dataset, and *Hermes* detects and corrects a hallucinated reasoning step.

C Examples of *Hermes* corrected problems

Figure 8 shows a case where *Hermes* helps the LLM avoid pursuing an incorrect reasoning path, in this example, attempting to prove that $\tan^{-1}(x) = \cot^{-1}(1/x)$ for $x < 0$. The Lean-validated steps guide the model away from invalid proof directions. We observe that LLMs often generate answers with unwarranted confidence and without verifying intermediate steps. Our agent mitigates this behavior by enforcing step-by-step validation and stricter reasoning discipline.

D Distribution of successfully proved problems by math topic under different inference strategies

To better understand the impact of our agent on the underlying reasoning models, we analyze the distribution of solved problems across different mathematical topics from MATH500 (Tables 4, 5, 6) and HardMath2 (Tables 7, 8, 9). Our results indicate that most mathematical domains benefit from the inclusion of Lean-based verification, although a few topics experience slight performance drops. Geometry, for instance, shows lower gains, which we attribute to the limited availability of geometry-related theorems in Mathlib, the primary mathematical library for Lean. Furthermore, we observe that certain reward models exhibit domain-specific strengths—for example, Skywork consistently performs better in geometry-related tasks. Overall, we expect that as Lean and Mathlib continue to evolve, these performance gaps will diminish, and the verification capabilities of our agent will further improve with future updates to theorem provers and supporting libraries.

Table 4: Distribution of correct answers by topic on MATH500 for the DeepSeek-V3.1 base reasoning model.

	Algebra	Counting & Probability	Geometry	Intermediate Algebra	Number Theory	Pre-algebra	Pre-calculus
ZS-CoT	123	33	35	91	62	78	52
Hermes	**124**	**36**	36	**93**	62	**80**	**56**
Skywork	123	35	**39**	92	62	77	55
ArmoRM	123	35	**39**	89	62	76	54
Shepherd	123	**36**	38	92	62	77	53
RLHFlow	123	34	35	91	62	79	53

Table 5: Distribution of correct answers by topic on MATH500 for the o3-mini base reasoning model.

	Algebra	Counting & Probability	Geometry	Intermediate Algebra	Number Theory	Pre-algebra	Pre-calculus
ZS-CoT	124	**38**	35	89	62	77	54
Hermes	124	37	**37**	93	61	**79**	55
Skywork	123	37	36	**94**	62	77	55
ArmoRM	124	37	35	93	61	76	55
Shepherd	124	**38**	35	91	62	78	54
RLHFlow	123	37	36	91	61	78	53

Table 6: Distribution of correct answers by topic on MATH500 for the Qwen3-8B base reasoning model.

	Algebra	Counting & Probability	Geometry	Intermediate Algebra	Number Theory	Pre-algebra	Pre-calculus
ZS-CoT	120	28	27	70	60	75	44
Hermes	**123**	**33**	31	**84**	61	**77**	47
Skywork	120	32	**32**	82	**62**	**77**	**50**
ArmoRM	118	31	31	80	**62**	75	46
Shepherd	121	31	29	72	61	**77**	48
RLHFlow	118	28	23	72	58	73	48

Table 7: Distribution of correct answers by topic on HARDMath2 for the DeepSeek-V3.1 base reasoning model.

	asympytotic series	boundary layers	integrals	nonlinear ode	nonlinear pde	other	wkb
ZS-CoT	0	22	9	2	8	3	4
Hermes	0	**28**	**10**	3	14	3	6
Skywork	0	23	8	3	16	3	7
ArmoRM	0	21	**10**	3	12	3	7
Shepherd	0	21	9	2	**17**	2	8
RLHFlow	0	20	8	1	**17**	1	7

Table 8: Distribution of correct answers by topic on HARDMath2 for the o3-mini base reasoning model.

	asympytotic series	boundary layers	integrals	nonlinear ode	nonlinear pde	other	wkb
ZS-CoT	0	14	10	3	13	1	8
Hermes	1	**25**	10	3	16	1	**10**
Skywork	1	22	10	1	**19**	1	8
ArmoRM	1	21	11	3	16	1	7
Shepherd	0	12	**12**	3	17	1	9
RLHFlow	**2**	21	10	2	16	1	8

Table 9: Distribution of correct answers by topic on HARDMath2 for the Qwen3-8B base reasoning model.

	asympytotic series	boundary layers	integrals	nonlinear ode	nonlinear pde	other	wkb
ZS-CoT	0	0	6	3	0	0	0
Hermes	0	**2**	**8**	2	0	0	2
Skywork	0	0	7	3	0	0	2
ArmoRM	0	0	6	3	0	0	2
Shepherd	0	0	7	3	0	0	2
RLHFlow	0	0	7	3	0	0	2