RLMEval: Evaluating Research-Level Neural Theorem Proving

Auguste Poiroux **Antoine Bosselut** **Viktor Kunčak**
School of Computer and Communication Sciences
EPFL, Switzerland
{auguste.poiroux, antoine.bosselut, viktor.kuncak}@epfl.ch

Abstract

Despite impressive results on curated benchmarks, the practical impact of large language models (LLMs) on research-level neural theorem proving and proof autoformalization is still limited. We introduce **RLMEval**, an evaluation suite for these tasks, focusing on research-level mathematics from real-world Lean formalization projects. RLMEval targets the evaluation of neural theorem proving and proof autoformalization on challenging research-level theorems by leveraging real Lean Blueprint formalization projects. Our evaluation of state-of-the-art models on RLMEval, comprising 613 theorems from 6 Lean projects, reveals a significant gap: progress on existing benchmarks does not readily translate to these more realistic settings, with the best model achieving only a 10.3 % pass rate. RLMEval provides a new, challenging benchmark designed to guide and accelerate progress in automated reasoning for formal mathematics.

1 Introduction

Automatically translating mathematical content from natural language into a formal language suitable for proof assistants (Szegedy, 2020; Wang et al., 2020), is crucial for bridging human mathematical reasoning with machine-verifiable proofs. Neural theorem proving (NTP) and proof autoformalization using large language models (LLMs) have demonstrated remarkable progress (Polu and Sutskever, 2020; Jiang et al., 2023) in these tasks. This success is predominantly measured on curated benchmarks such as MiniF2F (Zheng et al., 2022) or ProofNet (Azerbayev et al., 2023a). While valuable, these benchmarks suffer from issues, such as saturation (e.g., MiniF2F reaching 88.9 % success (Ren et al., 2025)), formalization inaccuracies (e.g., ∼30 % in ProofNet (Poiroux et al., 2024)), or a narrow focus on competition-style problems. They do not fully capture the complexities of real-world, research-level mathematics. Consequently, model performance on these benchmarks may not reliably predict their practical utility in assisting with ongoing, complex formalization projects.

To address this gap, this work introduces **RLMEval**[1], a benchmark for evaluating neural theorem proving and proof autoformalization on research-level mathematics within contemporary Lean 4 projects. RLMEval distinguishes itself by focusing on *blueprint theorems*, significant, high-level results from real-world Lean projects. These theorems embody core conceptual advances, unlike the more numerous auxiliary lemmas which often involve smaller, routine deductions and typically constitute over 75% of theorems in Lean projects (see Table 2). By concentrating on these challenging blueprint theorems, RLMEval provides a more realistic and demanding testbed for LLMs.[2] This targeted evaluation aims to steer LLM development towards capabilities that can meaningfully contribute to the advancement of formal mathematics. Our main contributions are the following:

1. RLMEval: A novel evaluation benchmark for research-level neural theorem proving and proof autoformalization. RLMEval is applicable to a wide range of Lean blueprint projects and, to our knowledge, is the first benchmark specifically designed for these advanced tasks at the research level within the Lean ecosystem. Extensible and versioned annually, RLMEval will continuously test model capabilities and limit data contamination.

2. An evaluation of state-of-the-art LLMs using RLMEval. This evaluation reveals a significant disparity in performance compared to established benchmarks, thereby pinpointing critical areas for future research and development.

[1] Apache 2.0 license, similar to the projects it relies on
[2] Examples from RLMEval are in Appendix A.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 10946–10957
November 4-9, 2025

2 Related Work

Research in neural theorem proving (NTP) has seen significant advancements, from early work like GPT-f (Polu and Sutskever, 2020) to recent LLM-based techniques (e.g., Xin et al., 2024a; Ren et al., 2025; Wang et al., 2025). Some methods explore using intermediate, possibly less-rigorous reasoning to guide formal proof generation, while others, like LeanAgent (Kumarappan et al., 2025), train on multiple Lean repositories in a curriculum to accumulate knowledge. Proof autoformalization has also progressed, with early breakthroughs in Isabelle where models like Codex translated informal math problems into formal specifications (Wu et al., 2022). The Draft, Sketch, and Prove (DSP) approach (Jiang et al., 2023) further improved this by using informal proofs to generate formal proof sketches, which are then completed by an automated theorem prover.

MINIF2F (Zheng et al., 2022) offers Olympiad-level problems across multiple proof assistants. PROOFNET (Azerbayev et al., 2023a) provides Lean theorems with informal statements/proofs, targeting both autoformalization and theorem proving. PUTNAMBENCH (Tsoukalas et al., 2024) focuses on challenging Putnam competition problems. These benchmarks are focused on competition-style problems or undergraduate-level mathematics. MINICTX (Hu et al., 2024) evaluates provers on real Lean projects but is tied to a specific Lean version and includes all theorems, many of which are auxiliary technical lemmas.

RLMEval differentiates itself by: (1) focusing on *research-level mathematics* drawn from real-world Lean blueprint projects (Massot, 2025); (2) specifically targeting *blueprint theorems*, which represent significant conceptual steps, unlike benchmarks that include numerous simpler lemmas (see Table 2); (3) introducing *proof autoformalization* (informal proof to formal proof) alongside neural theorem proving as core tasks; and (4) ensuring broad compatibility with Lean versions, allowing evaluation on a wider array of ongoing projects.

3 Methodology

RLMEval provides a comprehensive suite for evaluating neural theorem proving and proof autoformalization on research-level Lean mathematics. We release it along with a dedicated Python interface, LeanInteract, for robust communication with the Lean proof assistant (Moura and Ullrich, 2021), offering programmatic control over Lean through its REPL interface (Morrison, 2023). A key feature crucial for RLMEval is its multi-version support, achieved through manual backporting of the latest REPL features and bug fixes to all 41 Lean versions between v4.7.0-rc1 and v4.19.0 (a significant undertaking ensuring broad applicability). This approach prioritizes broad compatibility and ease of use for benchmarking across the evolving Lean ecosystem, setting it apart from existing interaction tools like LeanDojo (Yang et al., 2023) or Pantograph (Aniva et al., 2024), which are tied to specific Lean versions or require compute-intensive project tracing. This property is a key cornerstone of RLMEval for its extensibility, laying the foundations for long-term maintenance and future updates. Because our benchmark is based on real-world projects, data contamination concerns apply. In order to continuously mitigate this risks, we plan to release new versions of RLMEval with more recent Lean projects to continuously reduce data contamination risks when evaluating current and future models.

Benchmark Design. Our methodology for curating RLMEval is inspired by RLM25 (Poiroux et al., 2024), a research-level benchmark for autoformalization of theorem statements. RLMEval leverages existing Lean blueprint projects (Massot, 2025), which curate high-quality alignments between natural language mathematics and their formal counterparts. These projects, authored by domain experts, ensure the precision and real-world relevance of the benchmark data.

A core design principle of RLMEval is its focus on *blueprint theorems*, i.e. formal theorems that are linked in the informal blueprint of the projects. The blueprint theorems represent the main, high-level steps of mathematical development, akin to theorems found in research papers. This contrasts with previous works (Kumarappan et al., 2025; Hu et al., 2024) that include a large proportion of simpler, auxiliary lemmas. Table 2 illustrates the difference in proof lengths between the main theorems and auxiliary lemmas in RLMEval: 16.6 vs 6.6 lines on average. Auxiliary lemmas regularly represent more than 75 % of the total theorems in a Lean project. By focusing on blueprint theorems, RLMEval targets more complex, research-level reasoning tasks, providing a more challenging and realistic benchmark. RLMEval comprises 613 theorems from the 6 Lean projects detailed in Table 1. RLMEval supports the following tasks:

Table 1: Lean Blueprint projects used to build RLMEval. We obtained agreement from the primary authors of these projects to evaluate our models on them. Difficulty is a subjective assessment based on our experience with the projects and the models' performance.

Project	Domain	Difficulty	#Thms	Lean	First Commit
Carleson	Analysis	hard	110	v4.14.0-rc2	20 Oct 2023
FLT	Number Theory	medium	52	v4.14.0-rc2	19 Nov 2023
PFR	Combinatorics	hard	144	v4.14.0-rc3	13 Nov 2023
PNT	Analytic Number Theory	medium	99	v4.14.0-rc2	9 Jan 2024
FLT3	Number Theory	easy	84	v4.7.0-rc2	22 Mar 2024
TLB	Information & Probability Theory	medium	124	v4.13.0-rc3	22 Feb 2024

Table 2: Proof length statistics (in number of lines, comments are trimmed) for theorems within project blueprints (main theorems) versus auxiliary lemmas across the RLMEval projects. The '% Auxiliary lemmas' column indicates the proportion of all theorems that are auxiliary lemmas.

Project	% Auxiliary lemmas	Main theorems Proof Length	Auxiliary lemmas Proof Length
PFR	75.3 %	23.2	9.0
FLT	72.2 %	12.8	4.0
FLT3	15.9 %	8.8	4.9
Carleson	85.9 %	27.0	7.8
PNT	78.3 %	16.7	8.3
TLB	83.0 %	11.2	5.7
Avg	68.3 %	16.6	6.6

- **Neural Theorem Proving (NTP)**: Given a formal Lean statement, generate a complete and verifiable Lean proof.
- **Proof Autoformalization**: Given an informal (natural language) proof and its corresponding formal statement, generate a complete, verifiable Lean proof.

To assess models under varying conditions, RLMEval uses two evaluation modes:

- **Easy mode**: Models access all definitions and lemmas from the source project, including non-blueprint auxiliary lemmas, i.e. technical lemmas that are not included in the informal blueprint.
- **Normal mode**: Models access only blueprint theorems from the source project; they do not have access to these non-blueprint auxiliary lemmas and thus may need to prove equivalent intermediate results themselves. This simulates a more realistic task, closer to what mathematicians formalizing research results are faced with.

4 Experimental Setup and Results

We evaluate several LLMs on the RLMEval tasks. Our primary metric is `pass@k`, which measures the percentage of problems solved with at least one successful proof among k generated attempts. For our main results, we use $k = 128$ samples per problem, a budget significantly larger than the 8 samples used in miniCTX (Hu et al., 2024), allowing for a more comprehensive assessment. For each theorem, models receive the full in-file context up to the point where the proof is to be generated, including all preceding definitions, lemmas, and imports within that specific file. In normal mode, lemmas not present in the informal blueprint are excluded from the context, while in easy mode, all lemmas from the project are available.

Our baseline model is **Llemma 7B** (Azerbayev et al., 2023b), a model pretrained on mathematics, but not for proof-search specifically. We also evaluate leading models specifically tuned for Lean theorem proving: **DeepSeek-Prover-V1.5-RL** (Xin et al., 2024b), **DeepSeek-Prover-V2-7B** (Ren et al., 2025), **Goedel-Prover-SFT** (Lin et al., 2025), and **KiminaProver-Preview-7B** (Wang et al., 2025). Our evaluation on RLMEval examines their ability to generalize to research-level mathematics.

Figure 1 presents the main pass@128 rates for both neural theorem proving and proof autoformalization tasks, comparing "easy" and "normal" modes. The overall performance on RLMEval is markedly lower than on benchmarks like MiniF2F. For instance, DeepSeek-Prover-V2-7B, the best-performing model, achieves only 10.3 % pass@128 on proof autoformalization (normal mode), in stark contrast to its reported 75 %+ on MiniF2F with

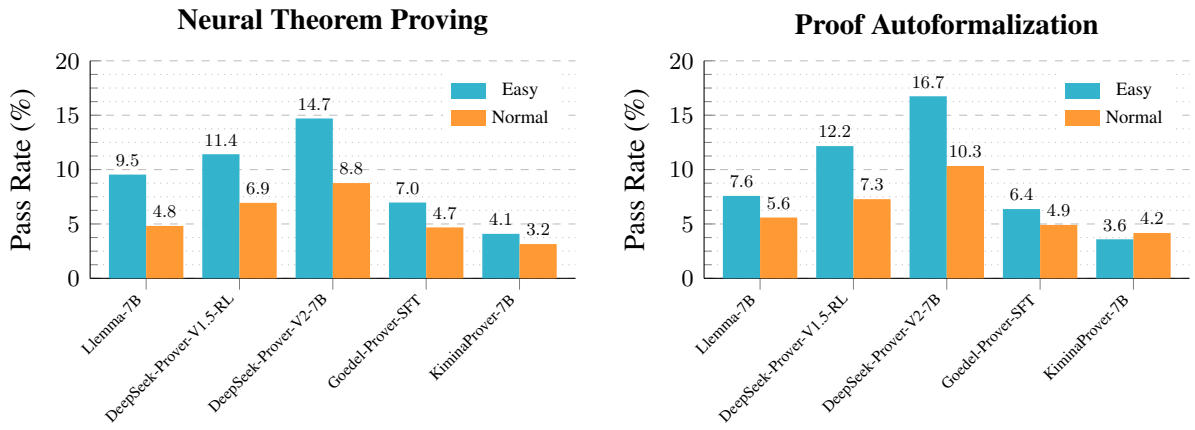

Figure 1: Pass rate on RLMEval using pass@128 for neural theorem proving (left) and proof autoformalization (right), in Easy and Normal modes.

a smaller pass@32 budget. This disparity underscores that **current models struggle significantly with the complexity of research-level mathematics in real-world projects**.

Models consistently perform better in the "easy" mode (with access to project-specific auxiliary lemmas). DeepSeek-Prover-V2-7B improves from 8.8 % (normal) to 14.7 % (easy) in neural theorem proving (NTP), and from 10.3 % to 16.7 % in proof autoformalization. This performance gap highlights the challenges models face when working without direct access to project-specific lemma support.

Providing an informal proof offers a modest benefit (proof autoformalization versus neural theorem proving (NTP)). For DeepSeek-Prover-V2-7B (normal mode), this translates to an improvement of ∼1.5 percentage points (from 8.8 % to 10.3 %). This suggests that current models can leverage informal proofs to some extent, but the benefit remains limited for complex, research-level problems.

Performance varies substantially across RLMEval projects (see Table 5 for details). For example, FLT3 yields the highest success rates (up to 32.1 % for DeepSeek-Prover-V2-7B), while Carleson presents a greater challenge (max 2.73 %). This variation indicates that mathematical domain and formalization style significantly impact model effectiveness.

Figure 2 illustrates how performance scales with an increasing number of samples per theorem for various models. As in traditional benchmarks, we observe that pass rate improves with additional samples. While pass rates continue to increase

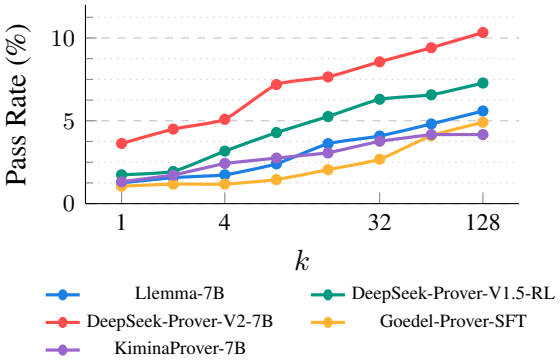

Figure 2: Scaling trends for the proof autoformalization task in normal mode on RLMEval for various models and pass@k values.

with more samples for most models, the gains diminish noticeably, suggesting that simply scaling up sampling may not overcome the fundamental challenges posed by research-level mathematics. This contrasts with results on benchmarks like miniF2F, where aggressive sampling (e.g., pass@8192) produced substantial improvements (Ren et al., 2025), further highlighting the greater difficulty of RLMEval.

Proof length. Proof length serves as a natural proxy for theorem complexity and human-perceived difficulty. Table 2 reveals a clear difficulty ordering based on average proof lengths: FLT3 < TLB < FLT < PNT < PFR < Carleson. This ordering strongly correlates with model performance trends shown in Table 5 and Table 6, where pass@k scores consistently degrade from FLT3 to Carleson across all evaluated models. This correlation validates proof length as a meaningful indicator of difficulty for both humans and auto-

mated theorem provers.

Analysis of successful model-generated proofs reveals significant insights about current capabilities. Table 3 shows that models successful proofs are short compared to their human-written counterparts, averaging only 2.5–6.0 lines compared to the 16.6-line human average. Critically, our inspection reveals that virtually all LLM-generated proofs are longer than the corresponding human proofs, indicating that current models primarily succeed on theorems that admit concise proof strategies, a subset representing the easier problems within RLMEval.

DeepSeek-Prover-V2-7B exhibits notably different behavior, generating proofs averaging 6.0 lines compared to 2.5–2.8 lines for other models, and producing the longest individual proof at 111 lines. Manual inspection reveals this model tends to generate verbose proofs with repetitive or redundant steps, suggesting room for improvement in proof conciseness and efficiency.

Model	Average	Max
Llemma-7B	2.8	20
DeepSeek-Prover-V1.5-RL	2.6	20
DeepSeek-Prover-V2-7B	6.0	111
Goedel-Prover-SFT	2.5	14
KiminaProver-7B	2.6	14
Human-Written	16.6	248

Table 3: Proof length in lines of code, comments trimmed, for successful proofs generated by models on RLMEval. The length of the official human-written proofs is reported for comparison. Data aggregated from the proof autoformalization, normal mode experiments.

5 Conclusion

We introduced **RLMEval**, a benchmark for neural theorem proving and proof autoformalization on research-level mathematics. Focusing on real-world mathematical formalization challenges, RLMEval establishes a realistic standard for evaluating the practical utility of neural theorem provers in mathematical settings. Our evaluation on 613 research-level mathematics theorems reveals a stark performance drop compared to traditional benchmarks like MiniF2F. The best model achieved only $10.3\,\%$ on proof autoformalization (normal mode), highlighting that current models are still far from reliably handling the complexities of ongoing formalization efforts. This performance gap is particularly pronounced for projects like Carleson with complex mathematical structures, while being somewhat narrower for more concrete domains like algebraic number theory (FLT3). The disparity between "easy" and "normal" modes across all models demonstrates the critical role of auxiliary lemmas in successful theorem proving. Access to these intermediate results improved performance by up to 6%, suggesting that enhanced techniques for lemma discovery or generation could be valuable directions for future research.

Limitations

Potential data contamination. Llemma 7B (Azerbayev et al., 2023b) has been released before the first commit of the projects used in RLMEval, making data leakage impossible. However, other models have been released more recently, and data contamination is unclear given the lack of information about the content of their pre-training set. Note, however, that data contamination would likely result in overestimating the current state of the art, which we already found to be low compared to more artificial benchmarks. Furthermore, thanks to the use of the LeanInteract tool (Poiroux et al., 2025), RLMEval is extensible to new Lean blueprint projects. As such, we plan to release new versions of RLMEval with more recent Lean projects to continuously reduce data contamination risks when evaluating current and future models.

Limited Scaling. We evaluate using pass@128 which, while being substantially higher than the pass@8 from miniCTX (Hu et al., 2024), falls short to the pass@8192 on MiniF2F or pass@1024 on ProofNet used by DeepSeek in Ren et al. (2025). Additionally, we use the 7B version of DeepSeek-Prover-V2 and not the 671B one. Our results therefore very likely underestimate the best achievable performance on RLMEval as of today. Nevertheless, the 7B model already achieves $75.6\,\%$ in a low compute budget (pass@32) on MiniF2F while the 671B version achieves $88.9\,\%$ on the latest sampling budget (pass@8192). These results indicate that even the 671B model would likely not achieve high scores on RLMEval. Additionally, running such a large model with pass@8192 requires a large amount of computation and time which, as of today, are a bit unrealistic for practical usage.

Context. Our current evaluation provides models with the full in-file context preceding the target theorem. While common, this setup may disadvantage

neural theorem proving (NTP) models trained on self-contained theorems, making them less adept at leveraging long, complex in-file contexts. Ideally, models should receive a more sophisticated context, incorporating broader project and research information. Future work should investigate optimal context retrieval strategies for these research-level tasks, moving beyond simple in-file truncation. Techniques could include retrieval-augmented generation or methods to summarize or filter premises from the entire project or its dependencies. Developing and evaluating such mechanisms is a key area for future work.

Informal Proofs. One interesting aspect of our evaluation is how the availability of informal proofs affects performance. While we observed a modest improvement when providing informal proofs over just providing formal statements, the benefit is somewhat limited. In particular, we found that some informal proofs in RLMEval can be cryptic without broader project context, using shorthand references to other theorems or principles. The PFR sample in Appendix A is such an example. These terse proofs illustrate the gap between idealized benchmark settings, using self-contained informal proofs, and real-world mathematics as depicted by RLMEval. Future work should explore how to better leverage informal proofs, possibly by providing additional context.

Acknowledgments

We thank the Lean community for their support and feedback, in particular the authors of the Lean blueprint projects included in RLMEval. We also gratefully acknowledge the support of the IC school of computer and communication sciences, the Swiss National Science Foundation (No. 215390), Innosuisse (PFFS-21-29), the EPFL Center for Imaging, Sony Group Corporation, and a Meta LLM Evaluation Research Grant.

References

Leni Aniva, Chuyue Sun, Brando Miranda, Clark Barrett, and Sanmi Koyejo. 2024. Pantograph: A Machine-to-Machine Interaction Interface for Advanced Theorem Proving, High Level Reasoning, and Data Extraction in Lean 4. *arXiv preprint*. ArXiv:2410.16429.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023a. ProofNet: Autoformalizing and Formally Proving Undergraduate-Level Mathematics. *arXiv preprint*. ArXiv:2302.12433 [cs].

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023b. Llemma: An Open Language Model For Mathematics. *arXiv preprint*. ArXiv:2310.10631 [cs].

Jiewen Hu, Thomas Zhu, and Sean Welleck. 2024. miniCTX: Neural Theorem Proving with (Long-)Contexts. *arXiv preprint*. ArXiv:2408.03350 [cs].

Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. 2023. Draft, Sketch, and Prove: Guiding Formal Theorem Provers with Informal Proofs. *arXiv preprint*. ArXiv:2210.12283 [cs].

Adarsh Kumarappan, Mo Tiwari, Peiyang Song, Robert Joseph George, Chaowei Xiao, and Anima Anandkumar. 2025. LeanAgent: Lifelong Learning for Formal Theorem Proving. *arXiv preprint*. ArXiv:2410.06209 [cs].

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. 2025. Goedel-Prover: A Frontier Model for Open-Source Automated Theorem Proving. *arXiv preprint*. ArXiv:2502.07640 [cs].

Patrick Massot. 2025. PatrickMassot/leanblueprint. Original-date: 2021-01-22T10:50:48Z.

Kim Morrison. 2023. leanprover-community/repl. Original-date: 2023-03-30T23:12:19Z.

Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, pages 625–635, Cham. Springer International Publishing.

Auguste Poiroux, Viktor Kuncak, and Antoine Bosselut. 2025. LeanInteract: A Python Interface for Lean 4. Original-date: 2025-02-05T21:35:17Z.

Auguste Poiroux, Gail Weiss, Viktor Kunčak, and Antoine Bosselut. 2024. Improving Autoformalization using Type Checking. *arXiv preprint*. ArXiv:2406.07222 [cs].

Stanislas Polu and Ilya Sutskever. 2020. Generative Language Modeling for Automated Theorem Proving. *arXiv preprint*. ArXiv:2009.03393 [cs, stat].

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. 2025. DeepSeek-Prover-V2: Advancing Formal Mathematical Reasoning via Reinforcement Learning for Subgoal Decomposition. *arXiv preprint*. ArXiv:2504.21801 [cs].

Christian Szegedy. 2020. A Promising Path Towards Autoformalization and General Artificial Intelligence. volume 12236, pages 3–20, Cham. Springer International Publishing. Book Title: Intelligent Computer Mathematics Series Title: Lecture Notes in Computer Science.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. 2024. Putnam-Bench: Evaluating Neural Theorem-Provers on the Putnam Mathematical Competition. *arXiv preprint*. ArXiv:2407.11214.

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 Saxc'e, 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.

Qingxiang Wang, Chad Brown, Cezary Kaliszyk, and Josef Urban. 2020. Exploration of Neural Machine Translation in Autoformalization of Mathematics in Mizar. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 85–98. ArXiv:1912.02636 [cs].

Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with Large Language Models. *arXiv preprint*. ArXiv:2205.12615 [cs].

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024a. DeepSeek-Prover: Advancing Theorem Proving in LLMs through Large-Scale Synthetic Data. *arXiv preprint*. ArXiv:2405.14333 [cs].

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. 2024b. DeepSeek-Prover-V1.5: Harnessing Proof Assistant Feedback for Reinforcement Learning and Monte-Carlo Tree Search. *arXiv preprint*. ArXiv:2408.08152 [cs].

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models. *arXiv preprint*. ArXiv:2306.15626 [cs, stat].

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. MiniF2F: a cross-system benchmark for formal Olympiad-level mathematics. *arXiv preprint*. ArXiv:2109.00110 [cs] version: 2.

A Appendix

A.1 Model Configurations

The evaluation of models for both Neural Theorem Proving (NTP) and Proof Autoformalization tasks was conducted using sampling-based decoding to compute pass@128. Specific hyperparameters for each model are detailed in Table 4. All models were accessed via a local vLLM server on H100 80GB GPUs.

A.2 Detailed results

Table 4: Hyperparameters for model evaluations. We use officially recommended hyperparameters for all models. All configurations used 128 samples per RLM25 entry.

Model	**Temp.**	**Top P**	**Max Gen. Tokens**	**Max Total Tokens**
Llemma 7B	0.7	0.95	1024	4096
DeepSeek-Prover-V1.5-RL	1.0	0.95	1024	4096
DeepSeek-Prover-V2-7B	1.0	0.95	1024	4096
Goedel-Prover-SFT	1.0	0.95	1024	4096
KiminaProver-7B	0.6	0.95	4096	16384

Table 5: Detailed pass@k rates (%) for Proof Autoformalization on RLMEval projects. Normal mode uses only blueprint lemmas, Easy mode uses all project lemmas. Projects are: PFR, FLT3 (Fermat's Last Theorem for n=3), Carleson (Carl.), FLT (Fermat's Last Theorem), TLB (testing-lower-bounds), PNT (Prime Number Theorem And).

Model	Mode	p@k	PFR	FLT3	Carl.	FLT	TLB	PNT	Total
Llemma 7B	Normal	p@1	0.69	3.57	0.00	0.00	3.23	0.00	1.25
		p@32	0.69	9.52	0.91	3.85	6.45	3.03	4.08
		p@128	0.69	14.29	0.91	5.77	8.87	3.03	5.59
	Easy	p@1	0.00	2.38	0.00	1.92	4.03	1.01	1.56
		p@32	0.69	9.52	0.91	9.62	6.45	3.03	5.04
		p@128	1.39	13.10	0.91	15.38	9.68	5.05	7.58
DeepSeek-Prover-V1.5-RL	Normal	p@1	0.00	2.38	0.91	1.92	3.23	2.02	1.74
		p@32	1.39	19.05	0.91	9.62	4.84	2.02	6.30
		p@128	2.08	22.62	0.91	9.62	6.45	2.02	7.28
	Easy	p@1	0.69	5.95	0.00	0.00	1.61	2.02	1.71
		p@32	0.69	16.67	0.91	19.23	12.90	8.08	9.75
		p@128	3.47	23.81	0.91	21.15	14.52	9.09	12.16
DeepSeek-Prover-V2-7B	Normal	p@1	0.69	11.90	0.00	5.77	2.42	1.01	3.63
		p@32	2.08	25.00	1.82	11.54	8.87	2.02	8.56
		p@128	2.08	32.14	2.73	11.54	10.48	3.03	10.33
	Easy	p@1	0.69	7.14	0.91	9.62	4.84	3.03	4.37
		p@32	3.47	27.38	1.82	23.08	19.35	10.10	14.20
		p@128	4.86	32.14	3.64	23.08	22.58	14.14	16.74
Goedel-Prover-SFT	Normal	p@1	0.00	3.57	0.00	1.92	0.81	0.00	1.05
		p@32	0.69	8.33	0.00	3.85	3.23	0.00	2.68
		p@128	0.69	13.10	0.00	9.62	4.03	2.02	4.91
	Easy	p@1	0.00	1.19	0.00	0.00	0.81	0.00	0.33
		p@32	0.00	8.33	0.00	13.46	4.84	2.02	4.78
		p@128	0.69	10.71	0.00	17.31	6.45	3.03	6.37
KiminaProver-7B	Normal	p@1	0.00	3.57	0.00	1.92	2.42	0.00	1.32
		p@32	0.00	10.71	0.00	7.69	3.23	1.01	3.77
		p@128	0.00	13.10	0.00	7.69	3.23	1.01	4.17
	Easy	p@1	0.00	3.57	0.00	1.92	0.00	0.00	0.92
		p@32	0.00	7.14	0.00	5.77	2.42	3.03	3.06
		p@128	0.00	9.52	0.00	5.77	3.23	3.03	3.59

Table 6: Detailed pass@k rates (%) for Neural Theorem Proving (NTP) on RLMEval projects. Normal mode uses only blueprint lemmas, Easy mode uses all project lemmas. Projects are: PFR, FLT3 (Fermat's Last Theorem for n=3), Carleson (Carl.), FLT (Fermat's Last Theorem), TLB (testing-lower-bounds), PNT (Prime Number Theorem And).

Model	Mode	p@k	PFR	FLT3	Carl.	FLT	TLB	PNT	Total
Llemma 7B	Normal	p@1	0.69	2.38	0.91	0.00	2.42	0.00	1.07
		p@32	0.69	8.33	0.91	5.77	6.45	2.02	4.03
		p@128	0.69	13.10	0.91	5.77	6.45	2.02	4.82
	Easy	p@1	0.00	2.38	0.91	1.92	1.61	2.02	1.47
		p@32	1.39	8.33	0.91	13.46	12.90	5.05	7.01
		p@128	2.08	11.90	0.91	21.15	16.13	5.05	9.54
DeepSeek-Prover-V1.5-RL	Normal	p@1	0.69	2.38	0.91	1.92	2.42	2.02	1.72
		p@32	1.39	14.29	0.91	5.77	6.45	3.03	5.31
		p@128	1.39	20.24	0.91	9.62	6.45	3.03	6.94
	Easy	p@1	0.69	2.38	0.00	7.69	3.23	2.02	2.67
		p@32	2.08	11.90	0.91	15.38	12.90	7.07	8.38
		p@128	2.78	20.24	0.91	21.15	15.32	8.08	11.41
DeepSeek-Prover-V2-7B	Normal	p@1	0.69	11.90	0.91	0.00	4.03	1.01	3.09
		p@32	0.69	22.62	1.82	9.62	7.26	1.01	7.17
		p@128	0.69	25.00	2.73	9.62	10.48	4.04	8.76
	Easy	p@1	0.00	9.52	0.00	11.54	6.45	3.03	5.09
		p@32	2.78	19.05	1.82	21.15	17.74	9.09	11.94
		p@128	4.86	22.62	1.82	25.00	21.77	12.12	14.70
Goedel-Prover-SFT	Normal	p@1	0.00	1.19	0.00	0.00	0.00	0.00	0.20
		p@32	0.69	7.14	0.00	7.69	4.84	0.00	3.39
		p@128	0.69	9.52	0.00	9.62	7.26	1.01	4.68
	Easy	p@1	0.00	4.76	0.00	0.00	3.23	0.00	1.33
		p@32	0.69	7.14	0.91	15.38	6.45	2.02	5.43
		p@128	0.69	11.90	0.91	15.38	8.87	4.04	6.97
KiminaProver-7B	Normal	p@1	0.00	3.57	0.00	5.77	0.81	0.00	1.69
		p@32	0.00	4.76	0.00	7.69	2.42	2.02	2.82
		p@128	0.00	5.95	0.00	7.69	3.23	2.02	3.15
	Easy	p@1	0.00	3.57	0.00	3.85	0.81	1.01	1.54
		p@32	0.00	4.76	0.00	11.54	3.23	3.03	3.76
		p@128	0.00	5.95	0.00	11.54	4.03	3.03	4.09

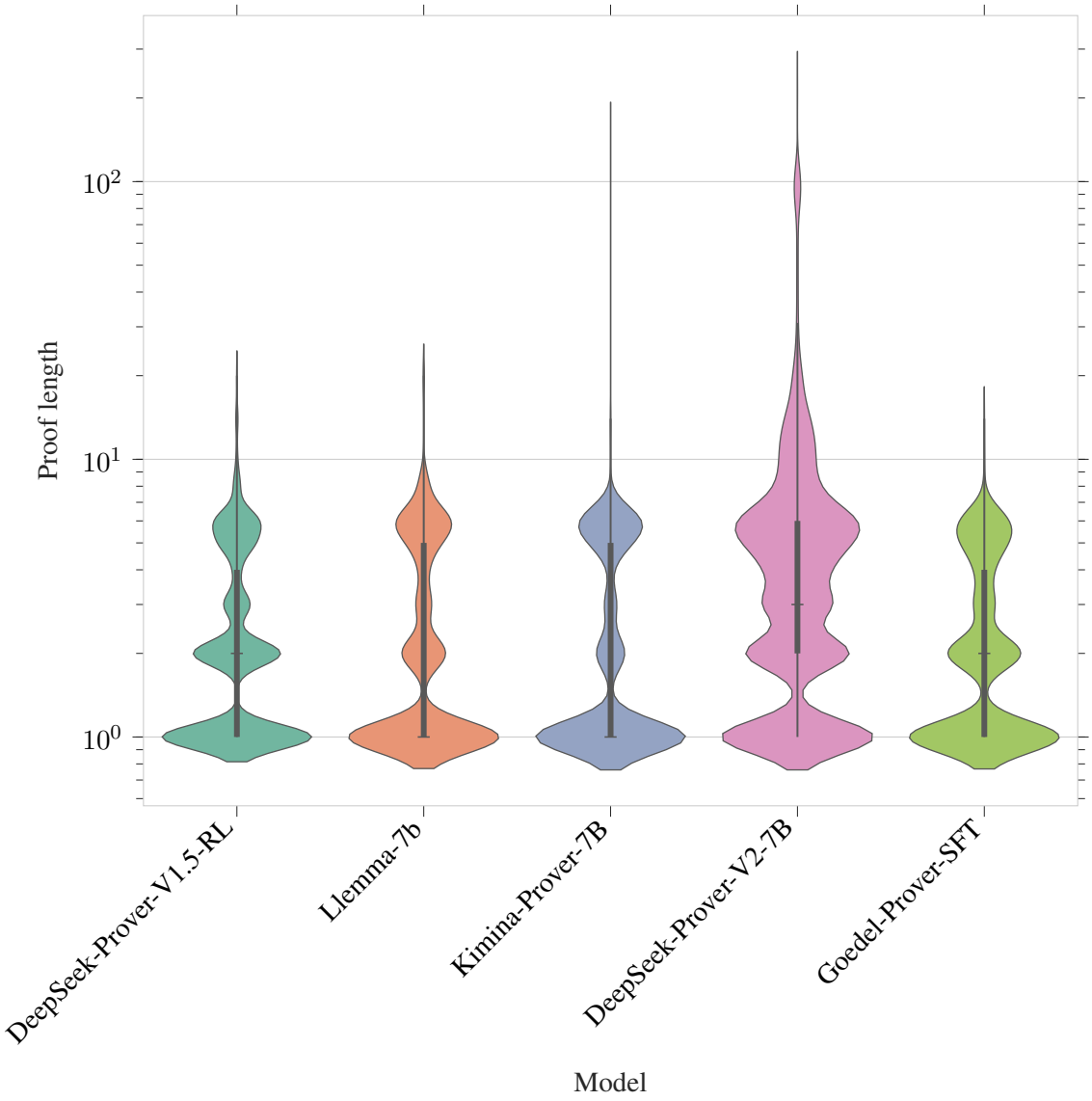

Figure 3: Proof length distribution of the different evaluated models on RLMEval. Proof length is measured in lines of code, comments trimmed. Data aggregated from all experiments: NTP and proof autoformalization, normal and easy modes.

A.3 Samples from the RLMEval benchmark

FLT3 sample - One of the simplest entry from RLMEval

Name: `lmm:lambda_not_dvd_Y`
File: `FLT3/FLT3.lean`

Theorem. *Given 'S : Solution', we have that 'λ' does not divide 'S.Y'.*

Formal statement:

```
lemma lambda_not_dvd_Y : ¬ λ | S.Y
```

Informal proof:
By contradiction we assume that $\lambda \mid Y$, then, by the properties of divisibility, $\lambda \mid u_2 Y^3$, which implies, by `def:Solution_u1_u2_u3_u4_u5_X_Y_Z`, that $\lambda \mid y$. However, this contradicts `lmm:lambda_not_dvd_y` forcing us to conclude that $\lambda \nmid Y$.
Formal proof:

```
intro h
have hyp := dvd_mul_of_dvd_right h (S.u₂ * S.Y^2)
rw [show ↑(u₂ S) * Y S ^ 2 * Y S = ↑S.u₂ * S.Y^3 by ring] at hyp
rw [← u₂_Y_spec] at hyp
apply lambda_not_dvd_y S
simp [hyp]
```

Carleson sample - Typical difficult entry of RLMEval

Name: `tile-sum-operator`
File: `Carleson/FinitaryCarleson.lean`

Theorem. *We have for all $x \in G \setminus G'$*

$$\sum_{\mathfrak{p} \in \mathfrak{P}} T_{\mathfrak{p}} f(x) = \sum_{s=\sigma_1(x)}^{\sigma_2(x)} \int K_s(x,y) f(y) e(Q(x)(y) - Q(x)(x))\, d\mu(y). \tag{1}$$

Formal statement:

```
theorem tile_sum_operator {G' : Set X} {f : X → ℂ}
  {x : X} (hx : x ∈ G \ G') : Σ (p : P X), carlesonOn p f x =
  Σ s in Icc (σ₁ x) (σ₂ x), ∫ y, Ks s x y * f y * exp (I * (Q x y - Q x x))
```

Informal proof:
Fix $x \in G \setminus G'$. Sorting the tiles $\mathfrak{p}$ on the left-hand-side of (1) by the value $\mathrm{s}(\mathfrak{p}) \in [-S, S]$, it suffices to prove for every $-S \le s \le S$ that

$$\sum \mathfrak{p} \in \mathfrak{P} : \mathrm{s}(\mathfrak{p}) = s T\mathfrak{p} f(x) = 0 \tag{2}$$

if $s \notin [\sigma_1(x), \sigma 2(x)]$ and

$$\sum \mathfrak{p} \in \mathfrak{P} : \mathrm{s}(\mathfrak{p}) = s T\mathfrak{p} f(x) = \int K s(x,y) f(y) e(Q(x)(y) - Q(x)(x)), d\mu(y). \tag{3}$$

if $s \in [\sigma_1(x), \sigma_2(x)]$. If $s \notin [\sigma_1(x), \sigma 2(x)]$, then by definition of $E(\mathfrak{p})$ we have $x \notin E(\mathfrak{p})$ for any $\mathfrak{p}$ with $\mathrm{s}(\mathfrak{p}) = s$ and thus $T\mathfrak{p} f(x) = 0$. This proves (2). Now assume $s \in [\sigma_1(x), \sigma 2(x)]$. By `coverdyadic`, `subsetmaxcube`, `eq-vol-sp-cube`, the fact that $c(I_0) = o$ and $G \subset B(o, \frac{1}{4} D^S)$, there is at least one $I \in \mathcal{D}$ with $s(I) = s$ and $x \in I$. By `dyadicproperty`, this I is unique. By `eq-dis-freq-cover`, there is precisely one $\mathfrak{p} \in \mathfrak{P}(I)$ such that $Q(x) \in \Omega(\mathfrak{p})$. Hence there is precisely one $\mathfrak{p} \in \mathfrak{P}$ with $\mathrm{s}(\mathfrak{p}) = s$ such that $x \in E(\mathfrak{p})$. For this $\mathfrak{p}$, the value $T\mathfrak{p}(x)$ by its definition in `definetp` equals the right-hand side of (3). This proves the lemma.
Formal proof:

```
rw [P_biUnion, Finset.sum_biUnion]; swap
· exact fun s _ s' _ hss' A hAs hAs' p pA ↦ False.elim <| hss' (s_eq (hAs pA) ▷ s_eq (hAs' pA
    ))
rw [← (Icc (-S : ℤ) S).toFinset.sum_filter_add_sum_filter_not (fun s ↦ s ∈ Icc (σ₁ x) (σ₂ x
    ))]
rw [Finset.sum_eq_zero sum_eq_zero_of_nmem_Icc, add_zero]
refine Finset.sum_congr (Finset.ext fun s ↦ ⟨fun hs ↦ ?_, fun hs ↦ ?_⟩) (fun s hs ↦ ?_)
· rw [Finset.mem_filter, ← mem_toFinset] at hs
exact hs.2
· rw [mem_toFinset] at hs
rw [toFinset_Icc, Finset.mem_filter]
exact ⟨Finset.mem_Icc.2 (Icc_σ_subset_Icc_S hs), hs⟩
· rcases exists_Grid hx.1 hs with ⟨I, Is, xI⟩
obtain ⟨p, IpI, Qp⟩ : ∃ (p : P X), I p = I ∧ Q x ∈ Ω p := by simpa using biUnion_Ω ⟨x, rfl⟩
have pPXs : p ∈ PX_s s := by simpa [s, IpI]
have : ∀ p' ∈ PX_s s, p' ≠ p → carlesonOn p' f x = 0 := by
  intro p' p'PXs p'p
  apply indicator_of_not_mem
  simp only [E, mem_setOf_eq, not_and]
  refine fun x_in_Pp' Qp' ↦ False.elim ?_
  have s_eq := s_eq pPXs ▷ s_eq p'PXs
  have : ¬ Disjoint (I p' : Set X) (I p : Set X) := not_disjoint_iff.2 ⟨x, x_in_Ip', IpI ▷ xI⟩
  exact disjoint_left.1 (disjoint_Ω p'p <| Or.resolve_right (eq_or_disjoint s_eq) this) Qp'
    Qp
rw [Finset.sum_eq_single_of_mem p pPXs this]
have xEp : x ∈ E p :=
  ⟨IpI ▷ xI, Qp, by simpa only [toFinset_Icc, Finset.mem_Icc, s_eq pPXs] using hs⟩
simp_rw [carlesonOn_def', indicator_of_mem xEp, s_eq pPXs]
```

PFR sample - Example of a relatively uninformative informal proof without broader context

Name: `cond-trial-ent`
File: `PFR/ForMathlib/Entropy/Basic.lean`

Theorem. *If ‘X, Y‘ are conditionally independent over ‘Z‘, then ‘H[X, Y, Z] = H[X, Z] + H[Y, Z] - H[Z]‘.*

Formal statement:

```
lemma ent_of_cond_indep (hX : Measurable X) (hY : Measurable Y) (hZ : Measurable Z)
     (h : CondIndepFun X Y Z μ) [IsZeroOrProbabilityMeasure μ]
     [FiniteRange X] [FiniteRange Y] [FiniteRange Z] :
     H[⟨X, ⟨Y, Z⟩⟩ ; μ] = H[⟨X, Z⟩; μ] + H[⟨Y, Z⟩; μ] - H[Z; μ]
```

Informal proof:
Immediate from `conditional-vanish` and `conditional-mutual-alt`.
Formal proof:

```
have hI : I[X : Y | Z ; μ] = 0 := (condMutualInfo_eq_zero hX hY).mpr h
rw [condMutualInfo_eq hX hY hZ] at hI
rw [entropy_assoc hX hY hZ, chain_rule _ (hX.prod_mk hY) hZ, chain_rule _ hX hZ, chain_rule _
    hY hZ]
linarith [hI]
```