Library Learning Doesn't: The Curious Case of the Single-Use "Library"

Ian Berlot-Attwell
University of Toronto
Vector Institute
ianberlot@cs.toronto.edu

Frank Rudzicz
Dalhousie University
Vector Institute
frank@dal.ca

Xujie Si
University of Toronto
Vector Institute
six@cs.toronto.edu

Abstract

Advances in Large Language Models (LLMs) have spurred a wave of LLM library learning systems for mathematical reasoning. These systems aim to learn a reusable library of *tools*, such as formal Isabelle lemmas [Paulson, 1994] or Python programs that are tailored to a family of tasks. Many of these systems are inspired by the human structuring of knowledge into reusable and extendable concepts [Ellis et al., 2021], but do current methods actually learn reusable libraries of tools?

We study two library learning systems for mathematics which both reported increased accuracy: LEGO-Prover [Wang et al., 2024a] and TroVE [Wang et al., 2024b]. We find that function reuse is extremely infrequent on miniF2F [Zheng et al., 2022] and MATH [Hendrycks et al., 2021]. Our followup ablation experiments suggest that, rather than reuse, self-correction and self-consistency are the primary drivers of the observed performance gains. Our code and data are available at `https://github.com/ikb-a/curious-case`.

1 Introduction

Mathematical progress is made by building with, and building upon, the tools of those who came before. Consequently, it is no surprise that there is research interest in developing systems that can automatically learn such reusable mathematical tools. Recently, LLMs have enabled new tool-learning methods with improved performance [Wang et al., 2024a,b, Zhang et al., 2024a, Yuan et al., 2024] – but are these systems truly learning generalized, reusable knowledge or is performance improved through other mechanisms? In this work, we study two prior systems: LEGO-Prover which aims to learn reusable formal Isabelle lemmas, and TroVE which aims to learn reusable Python functions. For both, our analysis of the model's behaviour reveals that direct reuse is negligible. Furthermore, we perform two ablation studies supporting our position that function reuse plays a limited role in these systems' improved mathematical reasoning.

2 Related Work

LLM library learning, i.e., creating and reusing tools, depends on LLMs' ability to use tools. Prior evaluations of tool-use (typically assuming tools as REST APIs) [Qu et al., 2024] included real-world queries [Yan et al., 2024], dedicated test environments [Li et al., 2023], and metrics ranging from LLM-as-a-judge [Guo et al., 2024] to tracking task-checkpoint completion [Lu et al., 2024].

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Table 1: Lemma reuse in LEGO-Prover released logs. Note that **lemma reuse is very uncommon**, and **no lemma reused twice**. For each split, we report the number of problems solved, the number of unique lemmas occurring in the PROVER's input prompts, the number of lemmas reused verbatim once, or more than once, and the number of lemmas whose *name* is reused once, or more than once. A lemma is reused N times if it appears in $N + 1$ solutions (i.e., the initial use, and then N reuses).

			Verbatim reused		Name reused	
Split	Problems Solved	Lemmas in Prompts	1	2+	1	2+
valid+GPT	127	374	0	0	1	0
valid+Human	135	265	0	0	1	0
test+GPT	111	255	0	0	2	0
test+Human	122	339	1	0	2	0

In contrast, the evaluation of library learning systems has been limited. Accuracy is the metric of choice [Wang et al., 2024a,b, Zhang et al., 2024a, Yuan et al., 2024], but cannot capture the extent or quality of reuse: an excellent library is useless to a weak reasoner, and a powerful reasoner can ignore a useless library and derive results from first principles. Prior attempts to evaluate library learning have been limited to static measures of individual functions such as cyclomatic complexity [McCabe, 1976, Zhang et al., 2024a] and abstract syntax tree depth [Wang et al., 2024b], or have answered specific questions such as the ease of human verification [Wang et al., 2024b], accuracy under domain transfer [Zhang et al., 2024a, Qian et al., 2023], or performance in the sub-problem of refactoring ground truth solutions[Lin et al., 2024].

In this study, we evaluate two library learning systems for mathematical reasoning: LEGO-Prover, and TroVE (see Sections 2.1 and 2.2). For a review of library learning systems, see Appendix A.

2.1 LEGO-Prover: Purpose & Architecture

LEGO-Prover consumes a set of proposed theorems to produce corresponding formal Isabelle [Paulson, 1994] proofs. It was evaluated on the miniF2F [Zheng et al., 2022] dataset: each problem was attempted 100 times, and the system obtained feedback from the Isabelle verifier after each attempt. LEGO-Prover was designed to perform library learning. Using the term *skills* in place of *tools*, Wang et al. [2024a] claimed that "LEGO-Prover enables LLMs to utilize existing skills retrieved from the library" and "[m]odular and reusable skills are constantly added to the library to enable tackling increasingly intricate mathematical problems." LEGO-Prover performs library learning via two LLM systems: 1) The PROVER which uses the library to create proofs, and 2) the EVOLVER which iteratively refines the library. They communicate through shared databases, such as the *request db* which stores proposed lemmas to be proven and added to library.

2.2 TroVE: Purpose & Architecture

TroVE is a "method for inducing a toolbox of reusable functions to use in solving programmatic tasks," designed to receive a stream of word problems without a ground truth or verifier [Wang et al., 2024b]. For each problem, it attempts to produce a Python program that prints the correct solution. TroVE's mathematical reasoning was evaluated with the MATH dataset Hendrycks et al. [2021]. Each problem is considered once: an LLM generates 15 solutions, and the best is selected based on self-consistency (i.e., majority vote) [Wang et al., 2023]. In generation, 5 solutions ignore the library and directly generate a program (SKIP mode), 5 create a reusable helper function for inclusion in the library (CREATE mode), and 5 use a function from the library (IMPORT mode).

3 Analysis of LEGO-Prover

We begin by analyzing the publicly released LEGO-Prover evaluation log files [1] [Wang et al., 2024a]. These logs are a subset of the unreleased PROVER logs corresponding to the final attempts on the

[1] `https://github.com/wiio12/LEGO-Prover/blob/357672c7751cd0c84aff6bf72a3d1bf97614e81d/result/lego_result.zip`

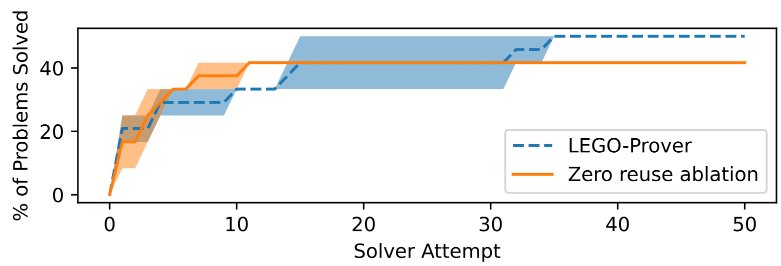

Figure 1: LEGO-Prover performance on a subset of the miniF2F validation split. The ablated model cannot reuse lemmas and performs similarly. The shaded region is one standard deviation, capturing variations in LLM output and race conditions.

successfully solved problems. Note that LEGO-Prover was evaluated on 4 data splits, and learned over 20,000 lemmas overall [Wang et al., 2024a].

We find that only 1,233 lemmas ($\sim$6%) are used in the final solving step (i.e., are inputs to the PROVER). Of these, exactly one lemma is reused by the PROVER, and it is reused once (i.e., appears verbatim in two solutions). As the PROVER may be adjusting a lemma (e.g., paraphrasing, commenting, etc...) we repeat the analysis, checking only for the lemma's name. Again, lemma reuse is rare, and no lemma is reused more than once (i.e., no lemma has its *name* appear in 3 or more solutions). See Table 1 for details. For an example of verbatim vs. name use, see Appendix B.

Given these findings, there are only two possibilities by which LEGO-Prover may be performing reuse: 1) indirect reuse (e.g., the learned tools are useful, reusable exemplars, rather than directly used in the final solution), or 2) direct reuse occurs in the EVOLVER.

Instead, we hypothesize that reuse is not significantly boosting performance. We propose that self-correction [Pan et al., 2023] via the *request db* is the main mechanism of action. Note that the PROVER populates the *request db* by: 1) adding lemmas that the LLM suggests may be helpful sub-steps, and 2) adding lemmas from solution attempts that Isabelle could not verify. The EVOLVER uses the *request db* to modify existing tools to "aid in solving requests", and to "resolv[e] decomposed sub-goals" using the library [Wang et al., 2024a]. Thus, the performance gains may be due to a combination of chain-of-thought [Wei et al., 2022] (through the PROVER's proposal of helpful lemmas for the EVOLVER to solve) and self-correction (through the EVOLVER's retrying of failed lemmas).

To test whether any form of reuse is increasing performance, we ablate LEGO-Prover to remove cross-problem sharing: each theorem is solved with its own independent state and databases. E.g., in place of a global *request db*, each problem now has its own independent *request db*. We evaluate on a random size 12 subset of the validation split and use 50 attempts per problem. We perform our ablation using OpenAI's GPT-4o-mini as the original results were published using now deprecated versions of GPT-3.5-Turbo; see Appendix E for full details of the ablation. Running 2 trials, we find that the ablation's performance is strong, solving only 1 question less than the baseline (see Figure 1). Studying the problems solved by only the baseline, we find that only the simplest of the input lemmas are possibly used (namely $a^2 \geq 0$ and $ax^2 + bx + c = 0 \Rightarrow c = -(ax^2 + bx)$; see Appendix C). It is unclear as these facts are not treated as lemmas, and are given different justifications. This suggests that: 1) the LLM may be too weak if it needs examples of basic facts 2) the LLM struggles at reuse as it does not copy the given, verified, proofs.

4 Analysis of TroVE

As TroVE logs were not released, we re-ran TroVE on MATH, achieving accuracy within $\pm 2\%$ (absolute) of reported (see Appendix, Table 3). Note that the TroVE library also learns import statements; we ignore these in our analysis for two reasons. Firstly, our interest is in whether the system learns and reuses non-trivial tools, unlike statements such as "`import math`" and "`from sympy import symbols`". Secondly, as TroVE includes the entire library as part of the IMPORT prompt, and import statements are innately simple, it is impossible to determine whether an import statement is included in the LLM output due to reuse, or the LLM's innate knowledge.

Table 2: TroVE performance on MATH for the ablation and the baseline. Mean and standard deviation over 5 trials are reported. The variations arise from LLM output. † indicates that mean ablation performance is significantly strictly higher than the baseline's, at the Bonferroni-corrected 0.05 level, using a 2-sample 1-sided Welch's t-test (note, this test assumes approximate normality).

	Accuracy on MATH test split			
Model	count	geo	inte	num
TroVE Reproduced	0.236 ± 0.008	$\mathbf{0.058} \pm 0.004$	0.120 ± 0.006	0.258 ± 0.007
No Reuse Ablation	$\mathbf{0.250} \pm 0.000$†	0.050 ± 0.000	$\mathbf{0.134} \pm 0.014$	$\mathbf{0.290} \pm 0.014$†

Analyzing the logs, we find that TroVE's final libraries only contain 15 learned functions, having learned functions for only 3 of the 7 MATH subject test splits: counting, number, and pre-algebra. No functions are learned in the algebra, geometry, intermediate algebra, or pre-calculus splits. Of the 15 learned functions, only 2 are reused in a correct solution: `is_perfect_square(n)` is reused in one correct solution and `is_prime(num)` is reused in two correct solutions.

Given 3 successful reuses in 3,201 test questions, we believe that TroVE's improvements over the baselines are not due to function reuse. Instead, we believe that ensembling and self-consistency are responsible. To test this, we ablate the model by disabling IMPORT mode, but maintaining the 15 solution attempts: we generate 8 solutions ignoring the library (i.e., SKIP mode) and 7 attempting to create a helper function (i.e., CREATE mode). As in the original work we use `CodeLlama-7b-Instruct-hf` [Rozière et al., 2023]; see Appendix F for the full ablation details. Ablating IMPORT mode prevents reuse as the library never appears in the model's input, thus also preventing library learning of import statements. As to why this ablation could still be performant, prior work established the benefits of self-consistency and increased sampling [Brown et al., 2024], and it's known that library-less tool-creation can boost performance by forcing abstract reasoning [Yuan et al., 2024].

We evaluate our ablated model on the intermediate_algebra test split (reportedly the largest performance gain over non-reuse baselines), and the geometry, number, and count test splits. On the intermediate_algebra, number, and count splits, our ablation exceeds the baseline's performance, with the improvement being statistically significant on two splits (See Table 2). On only the geometry split does the base model perform slightly better, though the learned libraries only contains import statements. From this we can conclude that library learning *import statements* can be slightly beneficial, but only for certain domains. Typically, TroVE's library learning degrades its performance.

5 Conclusions

In this study, we find that both TroVE and LEGO-Prover do not directly reuse the tools they learn. Furthermore, the results of our ablations suggest that their performance gains cannot be solely attributed to indirect reuse either.

We intend that this paper be a call for the better understanding of the limitations of current library learning systems, and for improved evaluation. We show that accuracy is misleading in isolation: the system's reuse behaviour is paramount, and careful ablation is critical. Both papers studied made sensible claims as the created systems were deliberately designed for library learning and were tested against ablations that were not unreasonable – however they also relied heavily on accuracy as a metric instead of directly observing the systems' use of the library, and both chose ablations that in hindsight were too aggressive. It is clear that, particularly for ablations of library learning systems, minimal changes are preferable, and considerable thought should be put into other possible causes of improvements. There is a clear need for a broadly applicable framework for the evaluation of library learning specifically; this framework must rely on more than task accuracy and ablations to evaluate library learning and reuse.

Finally, considering library learning for mathematics in general: are LLMs capable learning tools and performing direct, verbatim reuse? Given that the observed improvements do not come from direct reuse, would direct reuse actually improve systems for mathematical reasoning, or is it overly brittle making soft reuse desirable? These important questions follow from our findings, and should inform the design of future research into library learning systems.

6 Limitations & Broader Impact

Due to resource constraints, our ablation studies could be more thorough. Most obviously, we only study two models, and on two datasets. The LEGO-Prover ablation is not ideal, as library learning is disadvantaged by operating on a subset of the questions; this was necessary due to resource constraints. Another limitation is that LEGO-Prover's databases are pre-loaded with the full dataset of problems; consequently, the EVOLVERs are exposed to other problem statements – note, however, that the impact on testing reuse is minimal. Firstly, the PROVER cannot attempt to solve any of these other problems, thus the *request db* cannot gain pending lemmas related to other problems. Secondly, under the ablated model, tasks cannot share lemmas – any performance gains would come from having access to other sample problems instead of reuse.

While we demonstrate that the performance gains in mathematical reasoning seen by TroVE and LEGO-Prover cannot be attributed to the direct learning and reuse of tools, there is a very important but *subtly different* question which remains unanswered: whether these systems are at all capable of library learning. It is possible that these systems have the capacity to learn reusable functions and lemmas, but the datasets do not provide the opportunity. Manually inspecting the MATH dataset, our tentative conclusion is that the dataset is intrinsically not amenable to function learning with Python – we suspect the questions are too diverse, with the shared components already being captured by standard libraries. How this could be more formally demonstrated remains an important open question that is beyond the scope of this work.

This work has no immediate societal impact, rather, it highlights current limitations and challenges assumptions in this field. However, deploying tool-learning systems may carry a security risk from executing LLM-generated code (we sandboxed TroVE). More generally, library learning systems are self-improving through code generation, an approach that has raised concerns [Zelikman et al., 2023]. Unexpected behaviours may develop, thus requiring sandboxing and monitoring, at the very least.

Acknowledgments and Disclosure of Funding

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute `www.vectorinstitute.ai/partnerships/`. Generous support was also provided by the Microsoft Accelerating Foundation Models Research (AFMR) program.

We would also like to thank Zhiruo Wang, Zhaoyu Li, William Cunningham, and our anonymous reviewers for their time and conversations that helped in various ways to shape and improve this work. Finally, the lead author would like to thank Frank Rudzicz for years of guidance and support, and Xujie Si for both encouraging this work as being of interest to the mathematical reasoning community, and for providing critical resources without which it could not have been possible. Thank you everyone for helping make this work possible.

References

Lawrence C. Paulson. *Isabelle - A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer, 1994. ISBN 3-540-58244-4. doi: 10.1007/BFB0030541. URL `https://doi.org/10.1007/BFb0030541`.

Kevin Ellis, Catherine Wong, Maxwell I. Nye, Mathias Sablé-Meyer, Lucas Morales, Luke B. Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: bootstrapping inductive program synthesis with wake-sleep library learning. In Stephen N. Freund and Eran Yahav, editors, *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*, pages 835–850. ACM, 2021. doi: 10.1145/3453483.3454080. URL `https://doi.org/10.1145/3453483.3454080`.

Haiming Wang, Huajian Xin, Chuanyang Zheng, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, Jian Yin, Zhenguo Li, and Xiaodan Liang. LEGO-Prover: Neural theorem proving with growing libraries. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL `https://openreview.net/forum?id=3f5PALef5B`.

Zhiruo Wang, Graham Neubig, and Daniel Fried. TroVE: Inducing verifiable and efficient toolboxes for solving programmatic tasks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024b. URL https://openreview.net/forum?id=DCNCwaMJjI.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: a cross-system benchmark for formal Olympiad-level mathematics. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. *NeurIPS*, 2021.

Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. Offline training of language model agents with functions as learnable weights. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=2xbkWiEuR1.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi Fung, Hao Peng, and Heng Ji. CRAFT: Customizing LLMs by creating and retrieving from specialized toolsets. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=G0vdDSt9XM.

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. *CoRR*, abs/2405.17935, 2024. doi: 10.48550/ARXIV.2405.17935. URL https://doi.org/10.48550/arXiv.2405.17935.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Berkeley function calling leaderboard. https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html, 2024.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3102–3116. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.187. URL https://doi.org/10.18653/v1/2023.emnlp-main.187.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 11143–11156. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-acl.664.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. *CoRR*, abs/2408.04682, 2024. doi: 10.48550/ARXIV.2408.04682. URL https://doi.org/10.48550/arXiv.2408.04682.

T.J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, 1976. doi: 10.1109/TSE.1976.233837.

Cheng Qian, Chi Han, Yi Ren Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. CREATOR: Tool creation for disentangling abstract and concrete reasoning of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 6922–6939. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.462. URL https://doi.org/10.18653/v1/2023.findings-emnlp.462.

Xiaohan Lin, Qingxing Cao, Yinya Huang, Zhicheng Yang, Zhengying Liu, Zhenguo Li, and Xiaodan Liang. ATG: Benchmarking automated theorem generation for generative language models. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 4465–4480. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-NAACL.279. URL https://doi.org/10.18653/v1/2024.findings-naacl.279.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *CoRR*, abs/2308.03188, 2023. doi: 10.48550/ARXIV.2308.03188. URL https://doi.org/10.48550/arXiv.2308.03188.

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 Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code Llama: Open foundation models for code. *CoRR*, abs/2308.12950, 2023. doi: 10.48550/ARXIV.2308.12950. URL https://doi.org/10.48550/arXiv.2308.12950.

Bradley C. A. Brown, Jordan Juravsky, Ryan Saul Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large Language Monkeys: Scaling inference compute with repeated sampling. *CoRR*, abs/2407.21787, 2024. doi: 10.48550/ARXIV.2407.21787. URL https://doi.org/10.48550/arXiv.2407.21787.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. *CoRR*, abs/2310.02304, 2023. doi: 10.48550/ARXIV.2310.02304. URL https://doi.org/10.48550/arXiv.2310.02304.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=qV83K9d5WB.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024, 2024c. URL https://openreview.net/forum?id=ehfRiF0R3a.

Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, Ruyi An, Molei Qin, Chuqiao Zong, Longtao Zheng, Yujie Wu, Xiaoqiang Chai, Yifei Bi, Tianbao Xie, Pengjie Gu, Xiyun Li, Ceyao Zhang, Long Tian, Chaojie Wang, Xinrun Wang, Börje F. Karlsson, Bo An, Shuicheng Yan, and Zongqing Lu. Cradle: Empowering foundation agents towards general computer control, 2024. URL https://arxiv.org/abs/2403.03186.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. OS-Copilot: Towards generalist computer agents with self-improvement. *CoRR*, abs/2402.07456, 2024. doi: 10.48550/ARXIV.2402.07456. URL https://doi.org/10.48550/arXiv.2402.07456.

Haiteng Zhao, Chang Ma, Guoyin Wang, Jing Su, Lingpeng Kong, Jingjing Xu, Zhi-Hong Deng, and Hongxia Yang. Empowering large language model agents through action learning. *CoRR*, abs/2402.15809, 2024. doi: 10.48550/ARXIV.2402.15809. URL `https://doi.org/10.48550/arXiv.2402.15809`.

Zhenfang Chen, Rui Sun, Wenjun Liu, Yining Hong, and Chuang Gan. GENOME: Generative neuro-symbolic visual reasoning by growing and reusing modules. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=MNShbDSxKH`.

Chun-Yi Kuan, Chih-Kai Yang, Wei-Ping Huang, Ke-Han Lu, and Hung-yi Lee. Speech-Copilot: Leveraging large language models for speech processing via task decomposition, modularization, and program generation. *CoRR*, abs/2407.09886, 2024. doi: 10.48550/ARXIV.2407.09886. URL `https://doi.org/10.48550/arXiv.2407.09886`.

Min Zhang, Jianfeng He, Shuo Lei, Murong Yue, Linhan Wang, and Chang-Tien Lu. Can LLM find the green circle? investigation and human-guided tool manipulation for compositional generalization. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024, Seoul, Republic of Korea, April 14-19, 2024*, pages 11996–12000. IEEE, 2024b. doi: 10.1109/ICASSP48485.2024.10446355. URL `https://doi.org/10.1109/ICASSP48485.2024.10446355`.

Gabriel Grand, Lionel Wong, Matthew Bowers, Theo X. Olausson, Muxin Liu, Joshua B. Tenenbaum, and Jacob Andreas. LILO: Learning interpretable libraries by compressing and documenting code. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=TqYbAWKMIe`.

Larry A. Rendell. Toward a unified approach for conceptual knowledge acquisition. *AI Mag.*, 4(4): 19–27, 1983. URL `https://ojs.aaai.org/index.php/aimagazine/article/view/413`.

Ray J. Solomonoff. A formal theory of inductive inference. Part I. *Inf. Control.*, 7(1):1–22, 1964. doi: 10.1016/S0019-9958(64)90223-2. URL `https://doi.org/10.1016/S0019-9958(64)90223-2`.

Yoshua Bengio and Nikolay Malkin. Machine learning and information theory concepts towards an AI mathematician. *CoRR*, abs/2403.04571, 2024. doi: 10.48550/ARXIV.2403.04571. URL `https://doi.org/10.48550/arXiv.2403.04571`.

Zhaoyu Li, Jialiang Sun, Logan Murphy, Qidong Su, Zenan Li, Xian Zhang, Kaiyu Yang, and Xujie Si. A survey on deep learning for theorem proving. *CoRR*, abs/2404.09939, 2024. doi: 10.48550/ARXIV.2404.09939. URL `https://doi.org/10.48550/arXiv.2404.09939`.

Jin Peng Zhou, Yuhuai Wu, Qiyang Li, and Roger Baker Grosse. REFACTOR: Learning to extract theorems from proofs. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=fgKjiVrm6u`.

Elias Stengel-Eskin, Archiki Prasad, and Mohit Bansal. ReGAL: Refactoring programs to discover generalizable abstractions. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=FovMAzXUpj`.

Appendix

A Extended Related Work

Current LLM-based library learning systems tend to fall into two main camps: systems designed for general word problem solving, typically including mathematical reasoning and typically generating Python functions (e.g., Cai et al. [2024], Yuan et al. [2024], Wang et al. [2024b]), and agentic systems designed to interact with a specific, complex environment (e.g., Wang et al. [2024c], Tan et al. [2024], Wu et al. [2024], Zhang et al. [2024a], Zhao et al. [2024]).

Generally, such systems access the library via in-context learning (ICL); some place the entire library in the context [Wang et al., 2024b, Zhang et al., 2024a], whereas others first use a semantic-similarity retrieval step to allow for larger libraries. Yuan et al. [2024] in particular uses a retrieval system that incorporates a LLM-generated description of the tool to be retrieved; LEGO-Prover behaves similarly by having several phases where the system alternates between proposing useful tools to be added to the library, attempting to create these tools, and possibly retrieving these tools.

These systems are typically bottom-up (iteratively developing a library over time), though a handful of top-down approaches exist. These top-down approaches instead decompose a high-level description of the tasks into reusable modules [Chen et al., 2024, Kuan et al., 2024, Zhao et al., 2024, Zhang et al., 2024b]; to the best of the authors' knowledge this approach is yet to be applied to mathematical reasoning.

These LLM-based systems typically attempt to produce reusable tools via ICL: prompting the LLM to generate "reusable functions". In comparison, an older family of library learning work (e.g., Dreamcoder [Ellis et al., 2021] and LILO [Grand et al., 2024]) instead frame library learning as a matter of compression. In principle a function that compresses a set of solutions must be broadly applicable, and in practice a high-level function reduces the symbolic search space for program induction. More generally, compression has been of long standing interest in the field of artificial intelligence. Rendell [1983] defined conceptual knowledge as the ability to compress a raw space of possibilities into useful classes, and there are long-standing connections between compression and inductive reasoning. Framing inductive reasoning as the task of capturing the underlying pattern in a provided substring for the purposes of prediction, Solomonoff [1964] formalized induction as Bayesian reasoning under a prior favouring low Kolmogorov complexity. In other words, formalizing the concept of Occam's razor – that the simplest solution, that which can be highly compressed into a short description, is more likely. For a recent treatise on the value of compression, specifically within the area of mathematical reasoning, see Bengio and Malkin [2024].

Turing our attention to mathematics, deep learning in general and LLMs in particular have found broad application in theorem proving [Li et al., 2024]. Considering library learning specifically, a very closely related branch of work considers the problem of refactoring a collection of ground-truth solutions into reusable components. ATG [Lin et al., 2024] and REFACTOR [Zhou et al., 2024] train models to extract reusable formal lemmas from a provided set of ground-truth formal proofs. Similarly, ReGAL [Stengel-Eskin et al., 2024] refactors ground-truth Python solutions for the MATH dataset into a reusable library. These systems are valuable and may represent a better first step towards reusable knowledge, but their dependence on ground-truth solutions prevents them from being conventional library learning systems. In comparison, LEGO-Prover attempts to learn reusable lemmas and produce formal proofs from only formal problem statements, and informal natural language proofs – furthermore, Wang et al. [2024a] demonstrated that the latter could be automatically generated by ChatGPT with only a small degradation in system performance.

B Example of Verbatim Use versus Name Use by LEGO-Prover

Figure 2 is an example of verbatim use where an input lemma to the PROVER is used verbatim in the outputted solution.

In contrast, Figure 3 is an example of name use, where the name of the input lemma appears in the solution. In this case, the contents of the lemma are similar, but have significant differences. Note that an instance of verbatim use would, necessarily, also be an instance of name use.

A lemma is reused N times if it is used $N + 1$ times – i.e., if the lemma is used in $N + 1$ solutions.

C LEGO-Prover Solutions not Found by Reuse-Free Ablation

We performed two runs of the original model, in both cases it outperformed the ablation by solving one additional problem. We present the found proofs and input lemmas in Figures 4 and 6. For improved legibility, we also provide a typeset approximation in Figures 5 and 7. In addition to the

PROVER input	PROVER output

PROVER input

````
[... System prompt ...]

Here some useful skill for reference:
###### useful skill 1: ######
```isabelle
lemma step3:
  fixes x y ::real
  shows "x^2 + y^2 + (x^2 * y^2) + 1 \<ge> 1"
proof -
  have "x^2 + y^2 + (x^2 * y^2) + 1 - 1 = x^2 +
y^2 + (x^2 * y^2)"
    by simp
  then have "x^2 + y^2 + (x^2 * y^2) + 1 - 1 \<ge>
0"
    by simp
  then show ?thesis
    by simp
qed
```

###### useful skill 2: ######
```isabelle
lemma algebra_sqineq_2xyxysq:
  fixes x y :: real
  shows "x^2 + y^2 \<ge> 2 * x * y"
proof -
  have "(x - y)^2 \<ge> 0"
    by simp
  then have "x^2 - 2 * x * y + y^2 \<ge> 0"
    by (simp add: algebra_simps power2_diff)
  then have "x^2 + y^2 \<ge> 2 * x * y"
    by simp
  then show ?thesis
    by simp
qed
```

[... Additional skills and example problems ...]

## Problems
What is the least possible value of $(xy-1)^2+
(x+y)^2$ for real numbers $x$ and $y$?
$\textbf{(A)} ~0\qquad\textbf{(B)} ~\frac{1}
{4}\qquad\textbf{(C)} ~\frac{1
}{2} \qquad\textbf{(D)} ~1 \qquad\textbf{(E)} Show
that it is \textbf{(D)} ~1.

## Informal proof
Step 1: Expand the expression $(xy-1)^2+(x+y)^2$
to obtain $x^2+2xy+y^2+x^2y^2-2xy+1$.
Step 2: Simplify the expression to
$x^2+y^2+x^2y^2+1$.
Step 3: Observe that all squares are nonnegative,
so the minimum value of the expression is 1.
Step 4: Show that the minimum value of 1 can be
achieved when $x=y=0$.

## Formal statement
```isabelle
theorem amc12a_2021_p7:
  fixes x y ::real
  shows "1 \<le> ((x * y) - 1)^2 + (x + y)^2"
  apply (auto simp:algebra_simps power2_eq_square)
```

## Proof
````

PROVER output

````
```isabelle
theory amc12a_2021_p7
  imports Complex_Main
begin

(* lemma 1. A lemma that shows the distribution
properties of real numbers will be helpful for
step 1, which requires this lemma to expand the
equation *)
lemma step1:
  fixes x y :: real
  shows "(x * y - 1)^2 + (x + y)^2 = x^2 + 2 * x *
y + y^2 + x^2 * y^2 - 2 * x *
 y + 1"
  by (simp add: power2_eq_square algebra_simps)

(* lemma 2. A lemma that shows the properties of
quadratic equation will be useful for step 2,
which requires this lemma to simplify the equation
*)
lemma step2:
  fixes x y ::real
  shows "x^2 + 2 * x * y + y^2 + x^2 * y^2 - 2 * x
* y + 1 = x^2 + y^2 + x^2 * y^2 + 1"
proof -
  have "x^2 + 2 * x * y + y^2 + (x^2 * y^2) - 2 *
x * y + 1 = x^2 + y^2 + (x^2 * y^2) + 1"
    by (simp add: algebra_simps)
  then show ?thesis
    by simp
qed

(* lemma 3. A lemma that shows the properties of
quadratic equation will be useful for step 3,
which requires this lemma to conclude the equation
*)
lemma step3:
  fixes x y ::real
  shows "x^2 + y^2 + (x^2 * y^2) + 1 \<ge> 1"
proof -
  have "x^2 + y^2 + (x^2 * y^2) + 1 - 1 = x^2 +
y^2 + (x^2 * y^2)"
    by simp
  then have "x^2 + y^2 + (x^2 * y^2) + 1 - 1 \<ge>
0"
    by simp
  then show ?thesis
    by simp
qed

(* formal statement copy from the input *)
theorem amc12a_2021_p7:
  fixes x y ::real
  shows "1 \<le> ((x * y) - 1)^2 + (x + y)^2"
  apply (auto simp:algebra_simps power2_eq_square)
  using step1 step2 step3 by auto

end
```

This is the proof for the given problem.
````

Figure 2: Example of verbatim reuse by the LEGO-Prover. The input lemma is reproduced exactly in the PROVER's output.

PROVER input

```
[... System prompt ...]

Here some useful skill for reference:
###### useful skill 1: ######
```isabelle
theorem algebra_2rootspoly_apatapbeq2asqp2ab:
  fixes a b :: complex
  shows "(a+a) * (a+b) = 2 * a^2 + 2 * (a*b)"
proof -
  have "(a+a) * (a+b) = (a+a)*a + (a+a)*b"
    by (simp add: distrib_left)
  also have "... = 2*a^2 + 2*a*b"
    by (metis calculation distrib_left
mult.commute mult_2 numeral_2_eq_2
power2_eq_square)
  finally show ?thesis
    by simp
qed
```

useful skill 2:
```isabelle
lemma multi_distrib_complex:
  fixes a b c d :: complex
  shows "(a + b) * (c + d) = a * c + a * d + b * c
+ b * d"
  by (simp add: distrib_left distrib_right)
```

useful skill 3:
```isabelle
lemma algebra_1_generalized:
  fixes a b :: nat
  shows "a^2 + 2*a*b + b^2 = (a+b)^2"
  by (smt (z3) ab_semigroup_add_class.add_ac(1)
add.assoc add.commute add.left_commute power2_sum)
```

[... Additional skills and example problems ...]

Problems
Simplify $(a-1)(a+1)(a+2) - (a-2)(a+1).$ Show that
it is a^3 + a^2.

Informal proof
Step 1: Expand the expression $(a-1)(a+1)(a+2) -
(a-2)(a+1)$ by multiplying the binomials.
Step 2: Simplify the expression to obtain $(a^2-1)
(a+2)-(a-2)(a+1)$.
Step 3: Expand further to get $(a^3 + 2a^2 - a -
2) - (a^2 -a -2)$.
Step 4: Combine like terms to simplify the
expression to $a^3 + a^2$.
Step 5: Conclude that the simplified expression is
equal to $a^3 + a^2$.

Formal statement
```isabelle
theorem mathd_algebra_568:
  fixes a :: real
  shows "(a-1) * (a+1) * (a+2) - (a-2) * (a+1) =
a^3 + a^2"
```

Proof
```

PROVER output

```
```isabelle
theory mathd_algebra_568
  imports Complex_Main
begin

(* Lemma 1: Distributive property of
multiplication to simplify the main equation *)
lemma algebra_1_generalized:
  fixes a b c :: real
  shows "(a - b) * (a + c) = a^2 + a * c - b * a -
b * c"
  by (simp add: algebra_simps)

(* formal statement copy from the input *)
theorem mathd_algebra_568:
  fixes a :: real
  shows "(a-1) * (a+1) * (a+2) - (a-2) * (a+1) =
a^3 + a^2"
proof -
  (* Step 1: Expand the expression $(a-1)(a+1)
(a+2) - (a-2)(a+1)$ by multiplying the binomials.
*)
  have c0: "(a-1) * (a+1) * (a+2) - (a-2) * (a+1)
= (a^2 - 1) * (a+2) - (a-2) * (a+1)"
    by (simp add: algebra_simps)

  (* Step 2: Simplify the expression to obtain $
(a^2-1)(a+2)-(a-2)(a+1)$. *)
  have c1: "(a^2 - 1) * (a+2) - (a-2) * (a+1) =
(a^3 + 2*a^2 - a - 2) - (a^2 - a - 2)"
    using algebra_1_generalized[of "a^2" 1 2]
algebra_1_generalized[of "a" 2 1]

    by (simp add: algebra_simps)

  (* Step 3: Expand further to get $(a^3 + 2a^2 -
a - 2) - (a^2 -a -2)$ *)
  have c2: "(a^3 + 2*a^2 - a - 2) - (a^2 - a - 2)
= a^3 + a^2"
    by simp

  (* Step 4: Combine like terms to simplify the
expression to $a^3 + a^2$. *)
  show ?thesis using c0 c1 c2
    by simp
qed

end
```
```

Figure 3: Example of name reuse by the LEGO-Prover. Only the name of the input lemma needs to be reproduced exactly in the output. In this case, the body of the input lemma has been significantly adjusted. Note Figure 2 is also an example of name reuse, as the input lemma's name appears in the solution (in that particular case, along with the rest of the lemma).

Table 3: TroVE performance on MATH. For comparison with Wang et al. [2024b], all reported numbers are best over 5 trials. Variation between trials arises from the stochastic sampling of the underlying LLM.

	Best-of-5 accuracy on MATH test split			
Model	count	geo	inte	num
TroVE, Reported	**0.26**	**0.08**	0.11	0.25
TroVE Reproduced (ours)	0.24	0.06	0.13	0.27
TroVE, Reported CREATE-only ablation	0.14	0.06	0.05	0.16
No Reuse Ablation (ours)	0.25	0.05	**0.15**	**0.31**

Table 4: LEGO-Prover hyperparameters

Hyperparameter	value
Solution attempts per problem (num_attempts)	50
Number of PROVER processes (num_prover)	3
Number of EVOLVER processes (num_evolver)	8
Temperature (temperature)	0.7

observations in the main paper, it should be noted that there is redundancy among the retrieved lemmas – deduplication and retrieval of lemmas remain areas for improvement.

D TroVE MATH reproduction

See table 3 for the best-of-five accuracies reported by TroVE, and achieved by our reproduction of their results.

E LEGO-Prover Hyperparameters and Experiment Details

At the time of publication, the LEGO-Prover logs released by Wang et al. [2024a] and used in our analysis are available at `https://github.com/wiio12/LEGO-Prover/blob/357672c7751cd0c84aff6bf72a3d1bf97614e81d/result/lego_result.zip`.

LEGO-Prover is built on OpenAI's GPT-3.5-Turbo and the 2022 release of the Isabelle proof assistant, specifically using its abilities as a proof verifier. Note that due to the deprecation of the LLMs originally used by LEGO-Prover (`gpt-3.5-turbo-0301`, `gpt-3.5-turbo-0613`, `gpt-3.5-turbo-16k`, `gpt-3.5-turbo-16k-0613`, `gpt-3.5-turbo-16k`, `gpt-3.5-turbo-16k-0613`), we upgrade the underlying LLM from GPT-3.5-Turbo to GPT-4o-mini.

We use the default LEGO-Prover hyperparameters, except for the number of retry attempts which, following Wang et al. [2024a]'s ablations, we reduce to 50. See Table 4 for details.

Note that the LEGO-Prover is initialized with a seed library of tools, and our ablation retains this initialization. The core claim we aim to disprove is that the model's performance gains predominantly come from reusable lemmas, and our ablation prevents any cross-task reuse.

The specific 12 problems chosen uniformly at random for our ablation study are: aime_1991_p6.json, algebra_2varlineareq_xpeeq7_2xpeeq3_eeq11_xeqn4.json, amc12a_2008_p15.json, amc12a_2013_p8.json, amc12a_2021_p7.json, amc12b_2002_p3.json, amc12b_2003_p9.json, mathd_algebra_31.json, mathd_algebra_109.json, mathd_algebra_116.json, mathd_numbertheory_149.json, and numbertheory_sqmod4in01d.json

Note that LEGO-Prover requires both the problem statement, and an informal natural language proof for conversion. We use the same human-generated informal proofs as Wang et al. [2024a]. The authors bundled said informal proofs inside of the miniF2F .json files listed above, available for download from `https://github.com/wiio12/LEGO-Prover/tree/`

Input Lemmas

```
###### useful skill 1: ######
lemma quadratic_root_substitution:
  fixes a b c k x :: real
  assumes "a * x^2 + b * x + c = 0"
  shows "c = - (a * x^2 + b * x)"
proof -
  obtain lhs where eq: "lhs = a * x^2 + b * x + c"
using assms by simp
  have "lhs = 0" using assms by (metis eq)
  thus ?thesis by (simp add: eq)
qed

###### useful skill 2: ######
lemma sqrt_limit_general:
  fixes x :: real
  assumes "n > 0" "k > 0" "k = sqrt(x + k)"
  shows "x = k^2 - k"
proof -
  have "k^2 = x + k" using assms(3) by (smt
(verit) assms(2) less_eq_real_def real_sqrt_le_iff
real_sqrt_pow2_iff real_sqrt_zero)
  then show ?thesis by auto
qed

###### useful skill 3: ######
lemma sqrt_difference:
  fixes a b :: real
  assumes "a >= 0" "b >= 0"
  shows "sqrt a - sqrt b = (a - b) / (sqrt a +
sqrt b)"
proof -
  have "sqrt a - sqrt b = (sqrt a + sqrt b) *
(sqrt a - sqrt b) / (sqrt a + sqrt b)"
    by (metis add.left_cancel
add_cancel_right_right add_le_same_cancel2
assms(1) assms(2)
cancel_comm_monoid_add_class.diff_cancel
dual_order.eq_iff eq_divide_eq mult.commute
nonzero_mult_div_cancel_right
real_sqrt_eq_zero_cancel_iff real_sqrt_ge_0_iff
real_sqrt_le_0_iff real_sqrt_zero)
  also have "... = (a - b) / (sqrt a + sqrt b)"
    by (smt (verit) assms(1) assms(2)
divide_cancel_right real_sqrt_le_mono
real_sqrt_less_mono real_sqrt_mult_self
square_diff_square_factored)
  finally show ?thesis
    by (simp add: field_simps)
qed

###### useful skill 4: ######
lemma sqrt_limit_generalized:
  fixes x :: real
  assumes "k = sqrt(x + k)" "k > 0"
  shows "x = k^2 - k"
proof -
  have "k^2 = x + k" using assms by (metis
add.commute real_sqrt_ge_0_iff real_sqrt_ge_zero
real_sqrt_gt_0_iff real_sqrt_pow2
verit_comp_simplify1(3) zero_le_mult_iff
zero_le_square)
  hence "x = k^2 - k" by simp
  thus ?thesis by auto
qed
```

Final Proof

```
theory mathd_algebra_116
  imports Complex_Main
begin
(* lemma 1: A lemma to calculate the square of a difference *)
lemma square_of_difference:
  fixes a b :: real
  shows "(a - b)^2 = a^2 - 2 * a * b + b^2"
proof -
  have "a - b = a + -b" by simp
  then show ?thesis by sos
qed

(* formal statement copy from the input *)
theorem mathd_algebra_116:
  fixes k x :: real
  assumes h0 : "x = (13 - sqrt 131) / 4"
    and h1 : "2 * x^2 - 13 * x + k = 0"
  shows "k = 19/4"
proof -
  (* Step 1: Substitute x into the quadratic equation to
express it in terms of k *)
  have "2 * x^2 - 13 * x + k = 0" using h1 by simp
  then have "k = 13 * x - 2 * x^2" by simp

  (* Step 2: Calculate 2 * x^2 *)
  have "x^2 = ((13 - sqrt 131) / 4)^2"
    using h0 by blast
  then have "x^2 = (13 - sqrt 131)^2 / 16"
    by (simp add: power2_eq_square)
  then have "2 * x^2 = 2 * (13 - sqrt 131)^2 / 16"
    by simp
  then have "2 * x^2 = (13 - sqrt 131)^2 / 8"
    by simp

  (* Step 3: Expand (13 - sqrt 131)^2 *)
  have "(13 - sqrt 131)^2 = 169 - 26 * sqrt 131 + 131"
    using square_of_difference[of 13 "sqrt 131"] by simp
  then have "(13 - sqrt 131)^2 = 300 - 26 * sqrt 131"
    by simp

  (* Step 4: Substitute back to find 2 * x^2 *)
  have "2 * x^2 = (300 - 26 * sqrt 131) / 8"
    by (metis \<open>(13 - sqrt 131)\<^sup>2 = 300 - 26 * sqrt
131\<close> \<open>2 * x\<^sup>2 = (13 - sqrt 131)\<^sup>2 /
8\<close>)

  (* Step 5: Calculate 13 * x *)
  have "13 * x = 13 * ((13 - sqrt 131) / 4)"
    using h0 by simp
  then have "13 * x = (169 - 13 * sqrt 131) / 4"
    by simp

  (* Step 6: Adjust the expression for k *)
  have "k = (169 - 13 * sqrt 131) / 4 - (300 - 26 * sqrt
131) / 8"
    using `k = 13 * x - 2 * x^2` by (metis \<open>13 * x =
(169 - 13 * sqrt 131) / 4\<close> \<open>2 * x\<^sup>2 = (300
- 26 * sqrt 131) / 8\<close> diff_add_cancel diff_diff_eq
diff_diff_eq2 diff_zero divide_numeral_1 eq_diff_eq
eq_iff_diff_eq_0 frac_eq_eq mult.commute mult.left_commute
mult_2_right mult_eq_0_iff mult_numeral_1_right
nonzero_mult_div_cancel_right power2_eq_square
right_diff_distrib' zero_neq_numeral)

  (* Step 7: Combine the fractions *)
  have "k = (169 - 13 * sqrt 131) / 4 - (300 - 26 * sqrt
131) / 8"
    by (metis \<open>k = (169 - 13 * sqrt 131) / 4 - (300 - 26
* sqrt 131) / 8\<close>)
  then have "k = (2 * (169 - 13 * sqrt 131) - (300 - 26 * sqrt
131)) / 8"
    by (simp add: field_simps)

  (* Step 8: Simplify the combined expression *)
  have "2 * (169 - 13 * sqrt 131) = 338 - 26 * sqrt 131"
    by simp
  then have "k = (338 - 26 * sqrt 131 - 300 + 26 * sqrt 131) /
8"
    by (smt (verit) \<open>k = (2 * (169 - 13 * sqrt 131) -
(300 - 26 * sqrt 131)) / 8\<close>)
  then have "k = (338 - 300) / 8" by simp
  thus ?thesis by simp
qed
end
```

Figure 4: LEGO-Prover input lemmas (left) and found proof (right). The proof proves that $\forall k \in \mathbb{R}$: if $x = (13-\sqrt{131})/4$ and $2x^2 - 13x + k = 0$ then $k = 19/4$. See Figure 5 for a typeset approximation, and commentary of LEGO-Prover's use (and failure to use) the input lemmas.

Input Lemmas	Output Proof
Useful skill 1: $\forall a, b, c, k, x \in \mathbb{R}:$ $ax^2 + bx + c = 0 \Rightarrow c = -(ax^2 + bx)$ Demonstrates: assms, simp, (metis eq), (simp add: eq)	Define Lemma square_of_difference: $\forall a, b \in \mathbb{R} : (a - b)^2 = a^2 - 2ab + b^2$ Proof of Lemma: $a - b = a + (-b)$ Method: simp Lemma square_of_difference follows using method sos
Useful skill 2: $\forall x \in \mathbb{R} : n > 0, k > 0$ $k = \sqrt{x + k} \Rightarrow x = k^2 - k$ Demonstrates: assms, (smt (verit) assms(2) less_eq_real_def real_sqrt_le_iff real_sqrt_pow2_iff real_sqrt_zero), auto	Define theorem mathd_algebra_116: $\forall k \in \mathbb{R}:$ Assume $x = (13 - \sqrt{131})/4$ Assume $2x^2 - 13x + k = 0$ Then: $k = 19/4$ Proof: $2x^2 - 13x + k = 0$ $k = 13x - 2x^2$ Method: simp
Useful skill 3: $\forall a, b \in \mathbb{R} : \sqrt{a} - \sqrt{b} = (a - b)/(\sqrt{a} + \sqrt{b})$ Demonstrates: (metis add.left_cancel add_cancel_right_right add_le_same_cancel2 assms(1) assms(2) cancel_comm_monoid_add_class.diff_cancel dual_order.eq_iff eq_divide_eq mult.commute nonzero_mult_div_cancel_right real_sqrt_eq_zero_cancel_iff real_sqrt_ge_0_iff real_sqrt_le_0_iff real_sqrt_zero), (smt (verit) assms(1) assms(2) divide_cancel_right real_sqrt_le_mono real_sqrt_less_mono real_sqrt_mult_self square_diff_square_factored), (simp add: field_simps)	$x^2 = ((13 - \sqrt{131})/4)^2$ Method: blast $x^2 = (13 - \sqrt{131})^2/16$ Method: (simp add: power2_eq_square) $2x^2 = 2(13 - \sqrt{131})^2/16$ $2x^2 = (13 - \sqrt{131})^2/8$ Method: simp $(13 - \sqrt{131})^2 = 169 - 26\sqrt{131} + 131$ Method: using lemma square_of_difference and sos
Useful skill 4: $\forall x \in \mathbb{R} : k = \sqrt{x + k}, k > 0 \Rightarrow x = k^2 - k$ Demonstrates: assms, (metis add.commute real_sqrt_ge_0_iff real_sqrt_ge_zero real_sqrt_gt_0_iff real_sqrt_pow2 verit_comp_simplify1(3) zero_le_mult_iff zero_le_square), simp, auto	$(13 - \sqrt{131})^2 = 300 - 26\sqrt{131}$ Method: simp $2x^2 = (300 - 26\sqrt{131})/8$ Method: (metis \<open>(13 - sqrt 131)\<^sup>2 = 300 - 26 * sqrt 131\<close>\<open>2 * x\<^sup>2 = (13 - sqrt 131)\<^sup>2 / 8\<close>)
	$13x = 13((13 - \sqrt{131})/4)$ $13x = (169 - 13\sqrt{131})/4$ Method: simp $k = (169 - 13\sqrt{131})/4 - (300 - 26\sqrt{131})/8$ Method: (metis \<open>13 * x = (169 - 13 * sqrt 131) / 4\<close>\<open>2 * x\<^sup>2 = (300 - 26 * sqrt 131) / 8\<close>diff_add_cancel diff_diff_eq diff_diff_eq2 diff_zero divide_numeral_1 eq_diff_eq eq_iff_diff_eq_0 frac_eq_eq mult.commute mult.left_commute mult_2_right mult_eq_0_iff mult_numeral_1_right nonzero_mult_div_cancel_right power2_eq_square right_diff_distrib' zero_neq_numeral) $k = (169 - 13\sqrt{131})/4 - (300 - 26\sqrt{131})/8$ Method: (metis \<open>k = (169 - 13 * sqrt 131) / 4 - (300 - 26 * sqrt 131) / 8\<close>) $k = (2(169 - 13\sqrt{131}) - (300 - 26\sqrt{131}))/8$ Method: (simp add: field_simps) $2(169 - 13\sqrt{131}) = 338 - 26\sqrt{131}$ Method: simp $k = (338 - 26\sqrt{131} - 300 + 26\sqrt{131})/8$ Method: (smt (verit) \<open>k = (2 * (169 - 13 * sqrt 131) - (300 - 26 * sqrt 131)) / 8\<close>) $k = (338 - 300)/8$ Method: simp. Theorem follows.

Figure 5: A typset approximation of LEGO-Prover input lemmas (left) and found proof (right). The proof proves that $\forall k \in \mathbb{R}$: if $x = (13 - \sqrt{131})/4$ and $2x^2 - 13x + k = 0$ then $k = 19/4$. See Figure 4 for the original Isabelle lemmas and proof. Note that skill 1 may have been indirectly used in rewriting $2x^2 - 13x + k = 0$ as $k = 13x - 2x^2$. Skills 2, 3 and 4 do not seem to be used directly or indirectly; furthermore skills 2 and 4 are the same lemma. Their use of smt and metis may have encouraged the prover to use these same tools.

Input Lemmas

```
Here some useful skill for reference:

###### useful skill 1: ######
lemma nonneg_square:
  fixes a :: real
  shows "a^2 \<ge> 0"
proof -
  have "a^2 = a * a" by (simp add:
power2_eq_square)
  thus ?thesis by (simp add: mult_nonneg_nonneg)
qed


###### useful skill 2: ######
lemma square_non_negative:
  fixes a :: real
  shows "a^2 >= 0"
proof -
  have "a^2 = a * a" by sos
  then show ?thesis by auto
qed


###### useful skill 3: ######
lemma square_sum_nonneg:
  fixes x y :: real
  shows "x^2 + y^2 \<ge> 0"
  by (simp)


###### useful skill 4: ######
lemma min_sum_of_squares:
  fixes x y :: real
  shows "x^2 + y^2 >= 0"
proof -
  have "x^2 >= 0" by auto
  moreover have "y^2 >= 0" by auto
  ultimately have "x^2 + y^2 >= 0 + 0" by (simp
add: add_nonneg_nonneg)
  thus "x^2 + y^2 >= 0" by simp
qed
```

Final Proof

```
theory amc12a_2021_p7
  imports Complex_Main
begin

(* Lemma: The expression can be expressed in terms
of squares to show its non-negativity. *)
lemma expression_nonneg:
  fixes x y :: real
  shows "((x * y) - 1)^2 + (x + y)^2 >= 1"
proof -
  (* Step 1: Define the expression we want to
minimize as \( E = (xy - 1)^2 + (x + y)^2 \). *)
  let ?E = "((x * y) - 1)^2 + (x + y)^2"

  (* Step 2: Expand the expression \( E \): \( (xy
- 1)^2 + (x + y)^2 \) *)
  have "((x * y) - 1)^2 = (x^2 * y^2 - 2 * x * y +
1)" by sos
  have "(x + y)^2 = x^2 + 2 * x * y + y^2" by sos

  (* This simplifies to: \( E = x^2y^2 + x^2 + y^2
+ 1 \) *)
  have "?E = (x^2 * y^2 - 2 * x * y + 1) + (x^2 +
2 * x * y + y^2)"
    by sos

  (* Thus, we have: \( E = x^2 * y^2 + x^2 + y^2 +
1 \) *)
  have "?E = x^2 * y^2 + x^2 + y^2 + 1" by sos

  (* Step 3: Recognize that \( x^2y^2 \), \( x^2
\), and \( y^2 \) are all non-negative. *)
  have "x^2 * y^2 >= 0" by auto
  have "x^2 >= 0" by auto
  have "y^2 >= 0" by auto

  (* Therefore, the minimum value of \( E \)
occurs when each of these terms is minimized. *)
  (* The minimum occurs when \( x = 0 \) and \( y
= 0 \). *)
  have "?E >= 1" by sos

  (* Step 4: Set \( x = 0 \) and \( y = 0 \) to
achieve the minimum: \( E(0, 0) = 1 \). *)
  have "?E = 1" when "x = 0" and "y = 0"
    using that by auto

  (* Step 5: Conclude that the least possible
value of \( E \) is \( 1 \). *)
  thus ?thesis by sos
qed

(* formal statement copy from the input *)
theorem amc12a_2021_p7:
  fixes x y ::real
  shows "1 \<le> ((x * y) - 1)^2 + (x + y)^2"
  apply (auto simp:algebra_simps power2_eq_square)
  using expression_nonneg by sos

end
```

Figure 6: LEGO-Prover input lemmas (left) and found proof (right). The proof proves that $\forall x, y \in \mathbb{R} : 1 \leq (xy - 1)^2 + (x + y)^2$. See Figure 7 for a typeset approximation, and commentary of LEGO-Prover's use (and failure to use) the input lemmas.

Input Lemmas	Output Proof
Useful skill 1: $\forall a \in \mathbb{R} : a^2 \geq 0$ Demonstrates: (simp add: power2_eq_square), (simp add: mult_nonneg_nonneg) Useful skill 2: $\forall a \in \mathbb{R} : a^2 \geq 0$ Demonstrates: sos, auto Useful skill 3: $\forall x, y \in \mathbb{R} : x^2 + y^2 \geq 0$ Demonstrates: simp Useful skill 4: $\forall x, y \in \mathbb{R} : x^2 + y^2 \geq 0$ Demonstrates: auto, (simp add: add_nonneg_nonneg), simp	Define Lemma expression_nonneg: $\forall x, y \in \mathbb{R} : (xy-1)^2 + (x+y)^2 \geq 1$ Proof of Lemma: Let $E = (xy-1)^2 + (x+y)^2$ $(xy-1)^2 = (x^2y^2 - 2xy + 1)$ $(x+y)^2 = x^2 + 2xy + y^2$ $E = (x^2y^2 - 2xy + 1) + (x^2 + 2xy + y^2)$ $E = x^2y^2 + x^2 + y^2 + 1$ Method: sos $x^2y^2 \geq 0$ $x^2 \geq 0$ $y^2 \geq 0$ Method: auto $E \geq 1$ Method: sos $E = 1$ when $x, y = 0$ Method: auto Lemma expression_nonneg follows using method sos Define theorem amc12a_2021_p7: $\forall x, y \in \mathbb{R} : 1 \leq (xy-1)^2 + (x+y)^2$ Proof: Follows Lemma. Method: sos, applying (auto simp:algebra_simps power2_eq_square)

Figure 7: Typeset approximation of LEGO-Prover input lemmas (left) and found proof (right). See Figure 6 for the original Isabelle lemmas and proof. The proof proves that $\forall x, y \in \mathbb{R} : 1 \leq (xy-1)^2 + (x+y)^2$. Skills 1 and 2 are the same; the fact that $x^2 \geq 0$ is used, though the exact proof differs from the lemmas. Skills 3 & 4 are also the same, though they do not seem to be used.

`357672c7751cd0c84aff6bf72a3d1bf97614e81d/data/full_data/valid` at the time of publication.

Note that the mean and standard deviation in Figure 1 are calculated using Python 3.8.9, numpy 1.22.2, `numpy.mean()` and `numpy.std()`.

Our experiments were run on an internal cluster, running one trial at a time. Each trial used 180 GB of RAM, 50 CPU cores, OpenAI credits, and ran within 24 hours. We upper bound the total compute time required to run our LEGO-Prover experiments at 96 hours. The full project required more compute than the experiments reported as one trial failed due to an out-of-memory error. Based on Wang et al. [2024a]'s estimate of \$300 per trial, we estimate the cost in OpenAI credits of our experiments to be \$7.38 per trial as we run half the number of attempts and one twentieth the number of questions. Under this estimate, the total cost of all our experiments is $\sim$\$30.

Our code is modified from the released LEGO-Prover code base, available at `https://github.com/wiio12/LEGO-Prover` [Wang et al., 2024b], released under an MIT License. Evaluation is done using the miniF2F Zheng et al. [2022] dataset, available at `https://github.com/openai/miniF2F/tree/main`, which was released under the Apache License Version 2.0.

Our code is documented and released, alongside the generated LEGO-Prover logs. It is a minor modification to the existing code base, and there is no training stage or new limitations. The code is released under the same license as the parent repository.

F TroVE Hyperparameters and Experiment Details

TroVE uses `CodeLlama-7b-Instruct-hf` [Rozière et al., 2023] interacting with the Python3 interpreter. We use the hyperparameters specified in the paper, outlined in Table 5. The same hyperparameters are used for the ablation, and our reproduction of baseline TroVE.

The mean and standard deviation of our 5 experiment runs are reported in Table 2. They are calculated using Python 3.8.9, numpy 1.22.2, `numpy.mean()` and `numpy.std()`. The 2-sided t-test reported the

Table 5: TroVE hyperparameters

Hyperparameter	value
Library trim frequency (trim_steps)	500
Solution execution timeout in seconds (exec_timeout)	100
top-p (top_p)	0.95
Samples per prompt (num_return_sequences)	5
Temperature (temperature)	0.6
Max decode length (max_new_tokens)	512

same table is performed using the same version of Python, scipy 1.8.1, `scipy.stats.ttest_ind()`, with the settings `equal_var=False` and `alternative='less'`.

Our experiments were run on an internal cluster, running up to 4 trials at once. Each trial used 1 Nvidia A40 GPU, 64 GB of RAM, 16 CPU cores, and ran within 12 hours. Smaller datasets completed more quickly. We upper bound the total compute time required to run our TroVE experiments at 480 hours. The full project required more compute than the experiments reported as we also tried running TroVE with quantized CodeLlama, CodeLlama 13B and 70B, and GPT-4o-mini.

Our code is modified from the released TroVE code base, available at `https://github.com/zorazrw/trove` [Wang et al., 2024b], which was released under the CC-BY-SA-4.0 license. Evaluation is done using the MATH Hendrycks et al. [2021] dataset, available at `https://github.com/hendrycks/math`, which was released under an MIT License.

Our code is documented and released, alongside the generated TroVE logs. It is a minor modification to the existing code base, and there is no training stage or new limitations. The code is released under the same license as the parent repository.

F.1 Additional TroVE experiments

We also ran baseline TroVE using the larger CodeLlama 13B model, and found similar results with very little direct function use. The key difference with the 7B model was that a single function was learned for the geometry split, but it was never reused in a correct solution.

We also attempted to run baseline TroVE using the 70B model, however we discarded the results as the LLM's ethical safeguards were frequently tripped (e.g., giving reasons such as "it is not appropriate or ethical to provide assistance with academic assignments or graded exercises").

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We analyze LEGO-Prover logs and ablate the model in Section 3, and we analyze the TroVE logs and ablate the model in Section 4. In both cases we find little direct reuse, and our ablation performs similarly.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 6. Primary limitations are scope (2 models and 2 datasets), and resource constraints on the ablations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This work is empirical.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Hyperparameters are reported in Appendices E and F, the TroVE and LEGO-Prover codebases are publicly available as are the MATH and miniF2F datasets, our ablations are described in Sections 3 and 4, and we release our code, logs, and log analysis code. As to the underlying LLMs, TroVE uses open source CodeLlama, and our LEGO-Prover ablation runs on a much smaller dataset to reduce the OpenAI API costs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: As explained in the previous question on reproducibility, we release our code along with the logs analyzed. Furthermore, the core TroVE and LEGO-Prover code bases are already publicly available, and can be easily modified to implement the ablations described.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Hyperparameters are in Sections E and F, there is no training data, and the TroVE test set is the same as Wang et al. [2024b], and the LEGO-Prover test set a subset of that used in Wang et al. [2024a]. The exact problems used in the subset are listed in the same section as the hyperparameters.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For the LEGO-Prover ablation, error regions of 1 standard deviation are displayed in Figure 1, the caption states that the source of variation is the LLM output and race conditions within the system; the method used to compute mean and standard deviation (numpy) is stated in Appendix E. For the TroVE ablation, we report the mean and standard deviation in Table 2. The best-of-five accuracy is reported in the Appendix, Table 3) so that our values are comparable to those reported in Wang et al. [2024b]. Both tables state that variation arises from sampling from the LLM. The method used to compute mean and standard deviation (numpy) is stated in Appendix F.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Outlined in Appendix E for the LEGO-Prover experiments, and Appendix F for the TroVE experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: There are no human subjects, to the best of our knowledge there are no data concerns, or immediate societal impact or harms (the possible future risks from deploying tool-learning systems, and the precautions that should be taken in future research in self-improving systems are outlined in Section 6), and to the best of our knowledge our work is reproducible and legal.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We do not anticipate any immediate societal impact or harms, but we do discuss the possible future risks from deploying tool-learning systems, and the precautions that should be taken in future research in self-improving systems in Section 6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We present ablations of already publicly available models (LEGO-Prover and TroVE), neither of which we believe has a higher risk for misuse than the constituent publicly available LLM.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The creators of TroVE [Wang et al., 2024b], LEGO-Prover [Wang et al., 2024a], the MATH dataset [Hendrycks et al., 2021], and miniF2F [Zheng et al., 2022] are all cited in the abstract. The URLs and licenses are stated in Appendices E and F.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our code is documented and released, alongside the log files used in our analysis. As new assets are minor modifications to existing code bases with no training or new limitations, we simply state as much in Appendices E and F; the code will be released under the same license as the parent repositories.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No crowdsourcing or human subjects was done.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: There were no human study participants.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.