# Frontier of Formal Theorem Proving with Large Language Models: Insights from the DeepSeek-Prover

Huajian Xin[1*]

University of Edinburgh
h.xin-3@sms.ed.ac.uk
https://xinhuajian.wordpress.com/

Recent advancements in large language models have significantly influenced mathematical reasoning and theorem proving in artificial intelligence. Despite notable progress in natural language domains, language models still encounter substantial challenges in formal theorem proving, *e.g.* using Lean [5] and Isabelle [7], which requires rigorous derivations satisfying formal specifications of the verification system. Even advanced models like GPT-4 [6] struggle with complex formal proofs, underscoring the intricate nature of both the coding and the mathematics involved. A formal theorem proving model must not only grasp the syntax and semantics of formal systems like the Lean theorem prover but also align abstract mathematical reasoning with precise formal representation.

Language models in formal theorem proving typically employ two strategies: proof-step generation [2, 4, 11, 13] and whole-proof generation [10, 14]. The proof-step generation approach is motivated by the interactive nature of Lean's tactic mode, in which the compiler provides the access to the tactic state, *i.e.*, a structured representation summarising the current status of the proof, including all the relevant information such as the local context of hypotheses and pending goals. Given the intermediate tactic state, the proof-step generation approach predicts each subsequent tactic and verifies it using the formal verifier to obtain updated information about the current tactic state. This interactive process often employs tree search techniques to compose valid proofs through several iterations of tactic generation [8]. In contrast, the whole-proof generation approach treats the construction of formal proofs as a general code completion task. This branch of methods aims to generate the entire proof code based on the theorem statement and perform verification only at the end of the generation process. The simplicity of the whole-proof generation paradigm has been proven to offer high scalability [12] from the perspectives of both model training and inference deployment. In addition, the whole-proof generation model is trained to perform long-term planning for theorem proving, facilitating the integration and utilisation of the model's capabilities in natural language mathematical reasoning [3].

We present a unified approach that combines the strengths of both proof-step and whole-proof generation paradigms. We begin by training a whole-proof generation model, incorporating several auxiliary tasks to enhance its capabilities in

*This work was completed by Huajian Xin during his internship at DeepSeek AI.

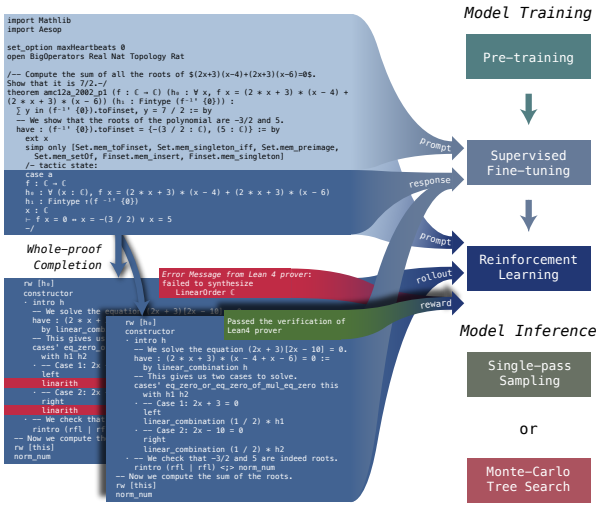


Fig. 1: Overall Framework of DeepSeek-Prover-V1.5.

mathematical reasoning and long-horizon planning, meanwhile empowering it to recognise information from Lean's proof assistant feedback. The model is named DeepSeek-Prover-V1.5, as it builds upon the prior work of DeepSeek-Prover-V1 [12]. We then employ a truncate-and-resume mechanism to decompose the whole-proof generation into a tactic-level proof search scheme. Figure 1 presents an illustration of our approach. The process begins with standard whole-proof generation, where the language model completes the proof code following the theorem statement prefix. The Lean assistant then verifies this code. If an error is detected, the code is truncated at the first error message, and any subsequent code is discarded. The successfully generated proof code is then used as a prompt for the generation of next proof segment. The latest tactic state from the Lean prover is appended at the end of the prompt as a comment block to provide intermediate guidance for the construction of long proofs. Notably, our method is not restricted to resuming from the last successfully applied tactic. We formalise the truncate-and-resume mechanism within the framework of Monte-Carlo tree search (MCTS) [1] in which the truncation points are scheduled by the tree search policy. In addition, we propose a novel reward-free exploration algorithm for MCTS to address the reward sparsity issue of proof search. We assign the tree search agent intrinsic motivation, *a.k.a.* curiosity [9], to extensively explore the tactic state space. These algorithmic modules extend the functionality of our whole-proof generation model to become a flexible tool for interactive theorem proving, which can effectively utilise the proof assistant feedback and generate diverse solution candidates. In experiments, we demonstrate substantial improvement of our proposed approach over baseline models, achieving new state-of-the-art results on the test set of the high school level miniF2F benchmark (63.5%) and the undergraduate level ProofNet benchmark (25.3%).

# Bibliography


[1] Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. In: International conference on computers and games. pp. 72–83. Springer (2006)

[2] Jiang, A.Q., Li, W., Tworkowski, S., Czechowski, K., Odrzygóźdź, T., Miłoś, P., Wu, Y., Jamnik, M.: Thor: wielding hammers to integrate language models and automated theorem provers. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. pp. 8360–8373 (2022)

[3] Jiang, A.Q., Welleck, S., Zhou, J.P., Lacroix, T., Liu, J., Li, W., Jamnik, M., Lample, G., Wu, Y.: Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In: The Eleventh International Conference on Learning Representations (2022)

[4] Lample, G., Lachaux, M.A., Lavril, T., Martinet, X., Hayat, A., Ebner, G., Rodriguez, A., Lacroix, T.: Hypertree proof search for neural theorem proving. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. pp. 26337–26349 (2022)

[5] Moura, L.d., Ullrich, S.: The lean 4 theorem prover and programming language. In: Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28. pp. 625–635. Springer (2021)

[6] OpenAI: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)

[7] Paulson, L.C.: Isabelle a Generic Theorem Prover. Springer Verlag (1994)

[8] Polu, S., Sutskever, I.: Generative language modeling for automated theorem proving. arXiv preprint arXiv:2009.03393 (2020)

[9] Schmidhuber, J.: Formal theory of creativity, fun, and intrinsic motivation (1990–2010). IEEE transactions on autonomous mental development **2**(3), 230–247 (2010)

[10] Wang, H., Xin, H., Zheng, C., Liu, Z., Cao, Q., Huang, Y., Xiong, J., Shi, H., Xie, E., Yin, J., et al.: Lego-prover: Neural theorem proving with growing libraries. In: The Twelfth International Conference on Learning Representations (2023)

[11] Wu, Z., Wang, J., Lin, D., Chen, K.: Lean-github: Compiling github lean repositories for a versatile lean prover. arXiv preprint arXiv:2407.17227 (2024)

[12] Xin, H., Guo, D., Shao, Z., Ren, Z., Zhu, Q., Liu, B., Ruan, C., Li, W., Liang, X.: Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. arXiv preprint arXiv:2405.14333 (2024)

[13] Yang, K., Swope, A.M., Gu, A., Chalamala, R., Song, P., Yu, S., Godil, S., Prenger, R., Anandkumar, A.: Leandojo: theorem proving with retrieval-augmented language models. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. pp. 21573–21612 (2023)

[14] Zhao, X., Li, W., Kong, L.: Decomposing the enigma: Subgoal-based demonstration learning for formal theorem proving. arXiv preprint arXiv:2305.16366 (2023)