Lemmanaid: Neuro-Symbolic Lemma Conjecturing

Yousef Alhessi[1], Sólrún Halla Einarsdóttir[2], George Granberry[2], Emily First[1], Moa Johansson[2], Sorin Lerner[1], and Nicholas Smallbone[2]

[1] University of California, San Diego, USA.
[2] Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden.

1 Introduction

Learning to construct new, interesting, and useful lemmas for proof assistants is an important yet underexplored area in AI for mathematical reasoning [11]. Such lemmas can aid a human user working on a mathematical formalization, as well as strengthen automated theorem provers. In this work, we examine how LLMs can be used for lemma generation, and how they can be combined with symbolic tools for optimal results. Our aim is to to provide a first tool towards generic conjecturing over a broad range of mathematical theories, which is practically useful for users of proof assistants.

A weakness of LLMs is that they sometimes generate repetitive or redundant lemmas, fail to discover more novel and useful lemmas, or hallucinate undefined symbols in the formalization. Furthermore, there are no correctness guarantees on the LLM's output, so the generated lemmas may simply be false. These challenges have been encountered in previous work on neural conjecturing [10, 7, 5]. Symbolic methods, on the other hand, can be designed and programmed to avoid repetition and redundancy. However, symbolic methods will only generate lemmas that fit a predefined specific search space, and tend to scale poorly to a larger search space. Previous symbolic tools [9, 3, 8] have been used to successfully discover, for example, lemmas needed in automated (co-)inductive provers [4, 2, 1, 6]. However, these tools are limited in the shape, size and domain of lemmas they can generate, and do not scale well to larger sets of inputs.

To address these shortcomings, we propose a novel neuro-symbolic lemma conjecturing approach and tool: LEMMANAID. An LLM is trained to generate *lemma templates* that describe the shape of a family of analogous lemmas, rather than directly generating complete lemmas. Symbolic synthesis methods are then used to fill in the details. In this way, we leverage the best of both neural and symbolic methods. The LLM suggests appropriate analogous lemma-patterns likely to be relevant for the theory at hand. The symbolic engine ensures correctness and novelty, while keeping the search space manageable. As far as we are aware, this is the first work focusing on neuro-symbolic lemma conjecturing.

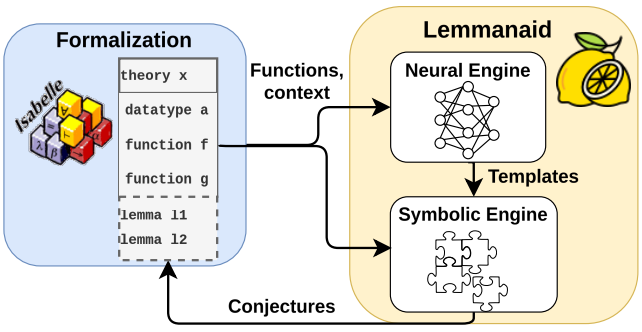

Figure 1: High-level overview of LEMMANAID.

2 Evaluation and Preliminary Results

As a challenging first evaluation step we measure the coverage of (test-set) lemmas that can be recovered by LEMMANAID from Isabelle's HOL library[1] and from its Archive of Formal Proofs (AFP)[2]. We compare the results of LEMMANAID configured with different LLMs (DeepSeek, Llama3), focusing on small LLMs as our ultimate aim is to create something that is accessible for regular proof-assistant users without need for huge compute resources.

We create a file-wise split of the HOL library so that we may evaluate the in-distribution capabilities of LLM-based approaches for lemma conjecturing tasks. Next, we supplement our training data with all projects from the AFP2024 that are published prior to 2024. We then create a new test set called AFP-test comprised of 31 AFP projects published in 2024 (and thus disjoint from the HOL+AFP training set). Training models on HOL+AFP-train and evaluating on HOL-test allows us to understand the effect of more training data as opposed to training on only HOL-train. Training models on HOL+AFP-train and evaluating on AFP-test allows us to evaluate an out-of-distribution task.

We define *lemma success rate* as the percentage of these lemma prediction tasks for which the given method is able to successfully generate (as part of the set of lemmas it generates) the ground-truth lemma (where we compare lemmas syntactically). This overall metric measures the performance of a method *end-to-end*.

	HOL-train		HOL+AFP-train	
Method	HOL-test	AFP-Test	HOL-test	AFP-Test
LEMMANAID	23.0%	6.5%	18.9%	6.8%
Neural	21.3%	4.3%	19.7%	7.0%
Combined	28.5%	8.0%	26.2%	10.1%

Figure 2: Lemma success rates.

The results found using the Deepseek-coder-1.3b model are shown in (Figure 2). The results found using the Llama3 model were similar but slightly lower numbers. We see that LEMMANAID trained on HOL-train outperform the respective neural baselines on all test sets. We also see that, while LEMMANAID does not greatly outperform neural methods, it is complementary to them, conjecturing more lemmas together. We see that on AFP-test, the performance drops for all variants trained on HOL-train. This is unsurprising, as the lemmas in AFP projects are more diverse than those in HOL. Still, LEMMANAID is complementary with neural-only methods on AFP-test. We see that when trained on HOL+AFP-train, both LEMMANAID's and the neural baseline's performance drop on HOL-test. Notably, the neural baseline's performance increases greatly on AFP-test when trained with more data. In ongoing further experiments we have seen the HOL-train/HOL-test result for LEMMANAID increase from 23.0% to 24.94% by adding more contextual information and further to 32.9% by using beam-search rather than greedy decoding, and we're looking forward to seeing how those changes may also improve results in the other columns.

We note that our experimental setup most likely under-reports results, as we measure matches with one specific target-lemma. It is entirely possible that LEMMANAID sometimes comes up with a different target lemma from the same theory, or even additional lemmas that are valid and useful but not present in the existing formalization. We have not yet explored the full potential of neuro-symbolic conjecturing for proof assistants.

[1] https://isabelle.in.tum.de/dist/library/HOL/index.html
[2] https://www.isa-afp.org

References

[1] S. H. Einarsdóttir, M. Hajdu, M. Johansson, N. Smallbone, and M. Suda. Lemma discovery and strategies for automated induction. In C. Benzmüller, M. J. Heule, and R. A. Schmidt, editors, *Automated Reasoning*, pages 214–232, Cham, 2024. Springer Nature Switzerland.

[2] S. H. Einarsdóttir, M. Johansson, and J. Å. Pohjola. Into the infinite - theory exploration for coinduction. In *Proceedings of AISC 2018*, pages 70–86, 01 2018.

[3] S. H. Einarsdóttir, N. Smallbone, and M. Johansson. Template-based theory exploration: Discovering properties of functional programs by testing. In *Proceedings of the 32nd Symposium on Implementation and Application of Functional Languages*, IFL '20, page 67–78, New York, NY, USA, 2021. Association for Computing Machinery.

[4] M. Johansson, D. Rosén, N. Smallbone, and K. Claessen. Hipster: Integrating theory exploration in a proof assistant. In *Proceedings of CICM*, pages 108–122. Springer, 2014.

[5] M. Johansson and N. Smallbone. Exploring mathematical conjecturing with large language models. In *17th International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2023*, 2023.

[6] C. Kurashige, R. Ji, A. Giridharan, M. Barbone, D. Noor, S. Itzhaky, R. Jhala, and N. Polikarpova. Cclemma: E-graph guided lemma discovery for inductive equational proofs. *Proc. ACM Program. Lang.*, 8(ICFP), Aug. 2024.

[7] M. N. Rabe, D. Lee, K. Bansal, and C. Szegedy. Mathematical reasoning via self-supervised skip-tree training. In *Proceedings of ICLR*, 2021.

[8] E. Singher and S. Itzhaky. Theory exploration powered by deductive synthesis. In A. Silva and K. R. M. Leino, editors, *Computer Aided Verification*, pages 125–148, Cham, 2021. Springer International Publishing.

[9] N. Smallbone, M. Johansson, K. Claessen, and M. Algehed. Quick specifications for the busy programmer. *Journal of Functional Programming*, 27, 2017.

[10] J. Urban and J. Jakubův. First neural conjecturing datasets and experiments. In *Proceedings of CICM*, 2020.

[11] K. Yang, G. Poesia, J. He, W. Li, K. Lauter, S. Chaudhuri, and D. Song. Formal mathematical reasoning: A new frontier in ai, 2024.