Neuro-Symbolic Lemma Conjecturing

Yousef Alhessi[1], Sólrún Halla Einarsdóttir[2,3], Emily First[1], George Granberry[2,3], Moa Johansson[2,3], and Nicholas Smallbone[2,3]

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

Abstract. We present ongoing work in combining Large Language Models (LLMs) and symbolic tools for lemma conjecturing. Our aim is to develop a neuro-symbolic lemma conjecturing tool leveraging the best of both symbolic and neural methods.

Keywords: Lemma conjecturing · Large Language Models · Theory exploration

1 Introduction

Theory exploration is the automatic discovery of interesting conjectures and lemmas about mathematical objects. Previously, we have developed symbolic tools for theory exploration [8, 3] which have been used to successfully discover, for example, lemmas needed in automated (co)-inductive provers [5, 2, 1]. However, these tools are limited in the shape and size of lemmas they can generate and do not scale well to larger sets of inputs. In light of the recent impressive results achieved by Large Language Models (LLMs) in various text-generation tasks, we examine how LLMs can be used for lemma generation in a theory exploration setting, and how they can be combined with symbolic tools for optimal results.

LLMs are remarkably good at learning patterns from their training data and generating output that fits a similar pattern for a given query context. Therefore, they can potentially be trained to generate lemmas similar to those previously seen for mathematical definitions analogous to those given, if exposed to the right kind of training data. A weakness of neural models such as LLMs is that they may be prone to generating repetitive or redundant lemmas and fail to discover more novel and useful lemmas. Another flaw that must be addressed when using LLMs in this context is the fact that there are no correctness guarantees on the LLM's output, so the generated lemmas may simply be false. Both of these challenges have been encountered in previous work on neural conjecturing [9, 7, 6]. Unlike neural methods, symbolic methods can be designed and programmed to generate only true statements and avoid repetition and redundancy. However, symbolic methods will only generate lemmas that fit a predefined specification from within a specified search space, and tend to scale poorly to a larger search space.

To address these shortcomings, we propose a neuro-symbolic lemma conjecturing tool with the following implementation: An LLM is trained to generate lemma templates that describe the shape of a lemma rather than generating complete lemmas. In these lemma templates the function symbols have been abstracted away and replaced by holes. For example, the template $?F(?F(X,Y),Z) = ?F(X,?F(Y,Z))$ describes an associative binary function $?F$. Symbolic methods are used to fill in the holes with function symbols in the exploration scope to form well-typed conjectures. In this way, we leverage the best of both neural and symbolic methods, using the LLM to capture the intuition and suggest appropriate patterns and symbolic methods to ensure correctness and novelty. As far as we are aware, this is the first work focusing on neuro-symbolic conjecturing of novel lemmas.

2 Ongoing Experiments

We set out to answer the following research questions:

RQ1 Can an LLM be trained to generate useful *lemmas* for a given set of function definitions?
RQ2 Can an LLM be trained to generate useful *lemma templates* to be filled in symbolically using a tool like RoughSpec?
RQ3 What level of contextual information is useful for an LLM to generate lemmas and lemma templates?

Generation tasks To find answers to the questions above, we set up our pipeline to perform the following lemma generation tasks:

1. Generate one or more lemmas for a given set of functions. We fine-tuned our model on training data consisting of Isabelle functions and their definitions as inputs, and lemma statements concerning those functions as outputs.
2. Generate one or more lemma templates for a given set of functions. We removed function names from the lemma statements in the training data, leaving a more abstract lemma template.
3. Repeat 1. and 2. above but give the model more context. Our prior work Baldur [4] showed that LLMs, when generating a proof of a given theorem, benefit from the Isabelle file context, which includes related theorems and their proofs. Thus, contextual information such as theorems about functions of interest could help the LLM generate templates.

Evaluation tasks For each of the lemma generation methods described above, we evaluate the generated lemmas in the following manner:

1. Syntax correctness: Is this valid Isabelle code/a valid template according to our template grammar?
2. Counterexample check: Can a counterexample-finder disprove this conjecture by finding a counterexample?
3. Proof: Can we prove the lemma using automated proof tactics? Is it a trivial consequence of previously proved facts, or does it require a more complicated proof?

References

1. Einarsdóttir, S.H., Hajdu, M., Johansson, M., Smallbone, N., Suda, M.: Lemma discovery and strategies for automated induction. In: Benzmüller, C., Heule, M.J., Schmidt, R.A. (eds.) Automated Reasoning. pp. 214–232. Springer Nature Switzerland, Cham (2024)
2. Einarsdóttir, S.H., Johansson, M., Pohjola, J.Å.: Into the infinite - theory exploration for coinduction. In: Proceedings of AISC 2018. pp. 70–86 (01 2018). https://doi.org/10.1007/978-3-319-99957-9_5
3. Einarsdóttir, S.H., Smallbone, N., Johansson, M.: Template-based theory exploration: Discovering properties of functional programs by testing. In: Proceedings of the 32nd Symposium on Implementation and Application of Functional Languages. p. 67–78. IFL '20, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3462172.3462192, https://doi.org/10.1145/3462172.3462192
4. First, E., Rabe, M., Ringer, T., Brun, Y.: Baldur: Whole-Proof Generation and Repair with Large Language Models. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. p. 1229–1241. ESEC/FSE 2023, Association for Computing Machinery, New York, NY, USA (Nov 2023). https://doi.org/10.1145/3611643.3616243, https://doi.org/10.1145/3611643.3616243
5. Johansson, M., Rosén, D., Smallbone, N., Claessen, K.: Hipster: Integrating theory exploration in a proof assistant. In: Proceedings of CICM. pp. 108–122. Springer (2014)
6. Johansson, M., Smallbone, N.: Exploring mathematical conjecturing with large language models. In: 17th International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2023 (2023)
7. Rabe, M.N., Lee, D., Bansal, K., Szegedy, C.: Mathematical reasoning via self-supervised skip-tree training. In: Proceedings of ICLR (2021)
8. Smallbone, N., Johansson, M., Claessen, K., Algehed, M.: Quick specifications for the busy programmer. Journal of Functional Programming **27** (2017)
9. Urban, J., Jakubův, J.: First neural conjecturing datasets and experiments. In: Proceedings of CICM (2020). https://doi.org/10.1007/978-3-030-53518-6_24