On Lemma Conjecturing using Neural, Symbolic and Neuro-symbolic approaches

Sólrún Halla Einarsdóttir[1], Yousef Alhessi[2], Emily First[2], and Moa Johansson[1]

[1] Chalmers University of Technology, Gothenburg, Sweden.
{slrn, moa.johansson}@chalmers.se
[2] University of California, San Diego, USA. {emfirst,yalhessi}@ucsd.edu

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.

1 Introduction

Theory exploration is the automatic discovery of interesting conjectures and lemmas. Previously, we have developed symbolic tools for theory exploration [13, 6] which have been used to successfully discover, for example, lemmas needed in automated (co)-inductive provers [9, 3, 2].

There has also been prior work on using purely neural methods for conjecturing. An early result of using large language models (LLMs) for the task of lemma generation used a GPT-2 model trained on Mizar theories [14]. Rabe et al. [12] experimented with a self-supervised approach. Our pilot study on automated conjecturing with LLMs was presented in [10], and used GPT-3.5 and GPT-4 out of the box via ChatGPT. Recent work [11] explores conjecturing using a constrained-decoding approach, guaranteeing well-formed conjectures. However, a common issue when using purely neural methods for conjecturing is that many of the generated lemmas can be duplicates, renamings or simply false. This is not the case with symbolic methods, but they may, on the other hand, miss large conjectures outside their specified search space [10].

Our aim is to develop a neuro-symbolic lemma conjecturing tool leveraging the best of both symbolic and neural methods. We are now following up on [10] with additional experiments on neural conjecturing, but also going further to develop a neuro-symbolic approach through combining the LLM with our work on data-driven conjecturing as presented at AITP 2022 [5].

For a neuro-symbolic approach, the symbolic component will come from an updated version of our template-based conjecturing tool RoughSpec [6], which restricts its search space to properties of specific shapes using *templates*. For example, the template $?F(?F(X,Y),Z) = ?F(X,?F(Y,Z))$ describes an associative binary function $?F$. In the original version of the tool, the human user decided which templates to use. In [4], we extracted a dataset of lemma templates from Isabelle's Archive of Formal Proofs[1] (AFP). We have now updated RoughSpec to parse templates from files in the format used in [4], so that it now can be run automatically without user intervention when given a file containing function definitions and a file containing templates as input. It can also now be run on input functions defined in Isabelle or SMT-LIB format, not only Haskell functions as previously.

We hypothesize that template-based conjecturing may be suitable as a component of a neuro-symbolic system, where the neural part suggests suitable templates and the symbolic part fills in the templates to produce conjectures, discarding any conjecture which is trivial, trivially false, or already known.

[1] https://www.isa-afp.org/index.html

2 Ongoing Experiments

We are interested in comparing the results achievable using purely neural conjecturing, purely symbolic conjecturing, and various neuro-symbolic combination approaches.

2.1 Neural conjecturing

We are experimenting with purely neural lemma generation, letting the LLM predict lemmas directly given function definitions, similar to [10].

As a first step, using the open-source 7B-parameter Llemma model [1], (a variant of LLama2 fine-tuned on e.g. proof assistant data) we prompted the model with an example of QuickSpec output (conjectured equational properties) for a set of function definitions and asked it to generate such output for a different set of function definitions. Our preliminary results indicated that this is not sufficient to generate useful conjectures. Although the output looks syntactically correct and many of the conjectures seem to hold, we notice a great deal of repetition and redundancy. This is not surprising, but rather served as a base-line seeing what results are achievable using available open-source models "out of the box."

We're aware that we can most likely achieve much better results if we fine-tune the models for lemma conjecturing. In order to do this, we have collected fine-tuning data consisting of function definitions and lemmas about them from Isabelle's AFP using the Portal-to-Isabelle API [8]. To create our training example input-target pairs, we have the target be the lemma statement and the input be a concatentation of the definitions and constants appearing in that lemma statement. We have fine-tuned the Facebook OPT 1.3B-parameter pre-trained model [15] on this data, and sampled from the model to predict relevant lemmas.

2.2 Neuro-symbolic conjecturing

Given function definitions, we want to ask the model to predict lemma templates that are useful for this context. We can then use symbolic methods (RoughSpec) to fill in the templates, ensuring we do not get repetitions and false conjectures.

We can also extend this approach to iterate in several rounds (i.e. several calls to the LLM), interleaved with counter-example checking. Our prior work Baldur [7] 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, after each round, the contextual information gathered, such as theorems about functions of interest, could help the LLM generate templates in subsequent rounds.

2.3 Evaluation

To evaluate the results of our experiments, we first consider the generated conjectures and find how many of them are 1) syntactically correct 2) true (no counter-example found by checker).

We plan to evaluate whether the lemma is provable by some chosen methods (such as Sledgehammer). We will compare the generated conjectures to the output of purely symbolic conjecturing with QuickSpec and consider the benefits and drawbacks of each respective method such as how much time and computing resources they need to run.

For further evaluation of the quality of generated conjectures, we can consider coverage, i.e. how many of the lemmas in a library can we generate (although here one must consider training data leakage), and evaluate the usefulness of the conjectured lemmas in automated proofs.

References

[1] Z. Azerbayev, H. Schoelkopf, K. Paster, M. Dos Santos, S. McAleer, A. Q. Jiang, J. Deng, S. Biderman, and S. Welleck. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.06786*, 2023.

[2] 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.

[3] 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.

[4] S. H. Einarsdóttir, M. Johansson, and N. Smallbone. Lol: A library of lemma templates for data-driven conjecturing. In *Work-in-progress papers presented at the 15th Conference on Intelligent Computer Mathematics (CICM 2022) Informal Proceedings*, page 22, 2022.

[5] S. H. Einarsdóttir, M. Johansson, and N. Smallbone. Towards neuro-symbolic conjecturing, 2022. Extended abstract accepted for presentation at the 7th Conference on Artificial Intelligence and Theorem Proving, AITP 2022.

[6] 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.

[7] E. First, M. Rabe, T. Ringer, and Y. Brun. 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*, ESEC/FSE 2023, page 1229–1241, New York, NY, USA, Nov. 2023. Association for Computing Machinery.

[8] A. Q. Jiang, W. Li, J. M. Han, and Y. Wu. Lisa: Language models of isabelle proofs. *6th Conference on Artificial Intelligence and Theorem Proving*, 2021.

[9] 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.

[10] 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.

[11] G. Poesia, D. Broman, N. Haber, and N. D. Goodman. Learning formal mathematics from intrinsic motivation. *arXiv preprint arXiv:2407.00695*, 2024.

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

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

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

[15] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.