What can Large Language Models do for Theorem Proving and Formal Methods?

Moa Johansson

Chalmers University of Technology

Abstract. With the introduction of large language models, AI for natural language have taken a leap. These systems are now also being used for tasks that has previously been dominated by symbolic methods, such as program synthesis and even to support formalising mathematics and assist theorem provers. We survey some recent applications in theorem proving, focusing on how they combine neural networks with symbolic systems, and report on a case-study of using GPT-4 for the task of automated conjecturing a.k.a. theory exploration.

1 Introduction

With recent developments of very capable Large Language Models (LLMs) such as GPT-4 [14], many of us now investigate how to best combine the generative capabilities of LLMs with symbolic systems like theorem provers. LLMs are sometimes unreliable and prone to hallucinate, simply "invent" stuff, and occasionally fail on seemingly trivial problems. While there is work on using LLMs to directly reason in mathematics, via chain-of-thought prompting and similar techniques e.g., [12], we believe a more reliable way forward is to induce the LLM to provide inputs to symbolic systems, like theorem provers to do the actual reasoning. Work on *autoformalisation* [17], concerns the task of translating problems expressed in natural or informal language into the input languages of interactive theorem provers such as Mizar [10] and recently also, by using LLMs, Isabelle/HOL [19] or Coq [2]. There is also work on getting LLMs to generate proof-scripts, or parts thereof [6, 5, 20]. These proof scripts may of course contain errors, but these will be flagged when run through the corresponding proof-assistant and can be patched by a human user or (symbolic) automated proof repair tools. Finally, we also mention that GPT-4 has recently been fitted with the capability to interact with external tools like Wolfram Alpha, via its Code Interpreter interface. This way, certain problems can be outsourced to external systems, for which suitable symbolic inputs are generated. Early results on problems from maths and science were mixed [3].

2 Automated Conjecturing

We are interested in the task of inventing suitable lemmas or conjectures, which can help theorem provers by enriching their background theories, in particular for automating proofs by induction [8, 9, 16]. Given a set of datatypes and function definitions, what interesting properties can be discovered? The problem of automated conjecturing is certainly not new to research in (symbolic) AI: there are several early systems based on specialised heuristics such as AM and Graffiti [11, 4], followed later by HR (integer sequences) and MATHsAiD (algebra) [1, 13]. The above-mentioned systems all use search, but there is also work on neural network driven methods [18, 15] which has managed to generate some lemmas, however also producing many repetitions of known lemmas and non-theorems.

2.1 Case Study: Theory Exploration in GPT-4

In a recent case-study (described in more detail in [7]), we wanted to investigate how the GPT-4 system would perform on a lemma discovery task, zero-shot. We prompted GPT-4 with theories written in the syntax of the Isabelle/HOL proof assistant and asked it to provide lemmas also in this syntax. This approach requires some care: as GPT-4 is trained on code from GitHub, we were aware that our benchmarks used from our Hipster theory exploration system [9], very likely were in the training data (which also was confirmed by GPT-4 occasionally producing close copies to Hipster's output), together with existing libraries for Isabelle/HOL. We thus instead chose to port a Haskell library about drawings and geometry to Isabelle. Some interesting observations can be made: Generally, GPT-4 was very consistently producing outputs in correct Isabelle syntax, which could be pasted into the proof-assistant with little or no further editing. Each run will differ slightly, as the system is probabilistic, but it consistently produces "generic" lemmas, such as associativity, commutativity and distributivity for binary functions and unary functions being their own inverses. As these properties are quite common, it is probably sensible suggestions, but does not take anything else from the function definitions into account, and hence lead to many non-theorems (false statements). Occasionally though, GPT-4 came up with other useful lemmas, for instance relationships between rotating drawings varying degrees.

Compared to a symbolic theory exploration system, GPT-4 can use the semantics of function names to occasionally "hallucinate" some additional function which could be of interest, but the user had not thought of including. It also does not have any restrictions of the size and shape of the lemmas generated, so no special treatment is needed for implications compared to equalities, unlike symbolic systems. On the downside, GPT-4 is less predictable than a symbolic system and will need to be run several times to achieve decent coverage. It is also difficult to fairly evaluate, as we don't know exactly what is in its training data.

3 Conclusions and Further Work

Using LLMs for lemma discovery has some complementary features to fully symbolic systems. How to customise and use such systems for making analogies between theories, use them to make analogies between theories and potentially combine them with symbolic lemma discovery systems is interesting further work. Lemma discovery systems have so far been limited to simple toy theories and simple functional programs, of little use to formal methods and more advanced mathematics, where proofs typically require highly contextualised lemmas of larger size than symbolic systems deal with well. Whether LLMs can help narrowing this gap is still an open question.

Bibliography

[1] S. Colton. The HR program for theorem generation. In A. Voronkov, editor, *Automated Deduction—CADE-18*, pages 285–289, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[2] G. Cunningham, R. C. Bunescu, and D. Juedes. Towards autoformalization of mathematics and code correctness: Experiments with elementary proofs, 2023.

[3] E. Davis and S. Aaronson. Testing GPT-4 with Wolfram Alpha and Code Interpreter plug-ins on math and science problems, 2023.

[4] S. Fajtlowicz. On conjectures of Graffiti. *Annals of Discrete Mathematics*, 38:113–118, 1988.

[5] E. First, M. N. Rabe, T. Ringer, and Y. Brun. Baldur: Whole-proof generation and repair with large language models, 2023. URL https://arxiv.org/abs/2303.04910.

[6] A. Q. Jiang, W. Li, S. Tworkowski, K. Czechowski, T. Odrzygóźdź, P. Miłoś, Y. Wu, and M. Jamnik. Thor: Wielding hammers to integrate language models and automated theorem provers. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=fUeOyt-2EOp.

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

[8] M. Johansson, L. Dixon, and A. Bundy. Conjecture synthesis for inductive theories. *Journal of Automated Reasoning*, 47(3):251–289, Oct 2011. ISSN 1573-0670. https://doi.org/10.1007/s10817-010-9193-y. URL https://doi.org/10.1007/s10817-010-9193-y.

[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] C. Kaliszyk, J. Urban, and J. Vyskocil. System description: Statistical parsing of informalized mizar formulas. In *2017 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 169–172, 2017. https://doi.org/10.1109/SYNASC.2017.00036.

[11] D. B. Lenat. AM, an artificial intelligence approach to discovery in mathematics as heuristic search. 1976.

[12] A. Lewkowycz, A. J. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu,

B. Neyshabur, G. Gur-Ari, and V. Misra. Solving quantitative reasoning problems with language models. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

[13] R. L. McCasland, A. Bundy, and P. F. Smith. MATHsAiD: Automated mathematical theory exploration. *Applied Intelligence*, Jun 2017. ISSN 1573-7497. https://doi.org/10.1007/s10489-017-0954-8. URL https://doi.org/10.1007/s10489-017-0954-8.

[14] OpenAI. GPT-4 technical report. Technical report, 2023. URL https://cdn.openai.com/papers/gpt-4.pdf.

[15] M. N. Rabe, D. Lee, K. Bansal, and C. Szegedy. Mathematical reasoning via self-supervised skip-tree training. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YmqAnY0CMEy.

[16] N. Smallbone, M. Johansson, K. Claessen, and M. Algehed. Quick specifications for the busy programmer. *Journal of Functional Programming*, 27, 2017. https://doi.org/10.1017/S0956796817000090.

[17] C. Szegedy. A promising path towards autoformalization and general artificial intelligence. In C. Benzmüller and B. Miller, editors, *Intelligent Computer Mathematics*, pages 3–20, Cham, 2020. Springer International Publishing. ISBN 978-3-030-53518-6.

[18] J. Urban and J. Jakubův. First neural conjecturing datasets and experiments. In C. Benzmüller and B. Miller, editors, *Intelligent Computer Mathematics*, pages 315–323, Cham, 2020. Springer International Publishing. ISBN 978-3-030-53518-6.

[19] Y. Wu, A. Q. Jiang, W. Li, M. N. Rabe, C. E. Staats, M. Jamnik, and C. Szegedy. Autoformalization with large language models. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IUikebJ1Bf0.

[20] K. Yang, A. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. Prenger, and A. Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. *arXiv preprint arXiv:2306.15626*, 2023.