Natural Synthesis: Outperforming Reactive Synthesis Tools with Large Reasoning Models

Frederik Schmitt[1], Matthias Cosler[1], Niklas Metzger[1], Julian Siber[1], Vladimir Krsmanović[1], Mohamed Ghanem[1], and Bernd Finkbeiner[1,2]

[1] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
[2] Technical University of Munich, Munich, Germany
firstname.lastname@cispa.de

Abstract. Reactive synthesis is a challenging problem for two reasons: It is algorithmically hard, and writing formal specifications by hand is notoriously difficult. In this extended abstract, we report on current advances in tackling both sides of the problem with Large Reasoning Models (LRMs). On the algorithmic side, we present a neuro-symbolic approach that couples LRMs with model checkers to iteratively repair a synthesized Verilog implementation via sound symbolic feedback. Our approach solves more benchmark instances than last year's SYNTCOMP winner and extends to constructing parameterized systems. On the specification side, we introduce an autoformalization step that shifts the specification task from temporal logic to natural language and introduce a dataset of natural-language specifications for evaluation. We demonstrate performance comparable to that of starting from formal specifications, establishing *natural synthesis* as a viable end-to-end workflow.

1 Introduction

Large Language Models (LLMs) and Large Reasoning Models (LRMs) have recently demonstrated the ability to bridge informal and formal languages [14,1] and have achieved remarkable success in formal reasoning tasks [9,3]. Those abilities make them promising tools for hardware design and in particular for synthesis applications. Synthesis lends itself to neuro-symbolic methods given that generated implementations can be automatically verified and hence can be used to guide the model predictions. So far, neural and symbolic integrations have only been studied for comparatively small neural networks trained from scratch [13,2] and in an LRM evaluation without verification feedback [4]. In this extended abstract, based on [12], we report on recent results on the capabilities of LRMs for the reactive synthesis problem and provide an empirical perspective on how modern LRMs fare against state-of-the-art synthesis algorithms. We first present a combination of LRMs with feedback from formal verification tools to generate provably correct hardware circuits from temporal logic specifications (Section 2). We then apply the same methodology to parameterized synthesis and extend it to handle natural-language specifications (Section 3).

2 Counterexample-Guided Synthesis with Large Reasoning Models

In this section, we consider the classical reactive synthesis problem for linear-time temporal logic. We closely follow the problem formulation posed in the annual reactive synthesis competition (SYNTCOMP) [6] with specifications expressed in Temporal Logic Synthesis Format (TLSF) [7]. We take, however, a fundamentally different approach than algorithms in the competition: We use LRMs to directly generate implementations in a hardware description language and introduce a feedback loop via model checking.

Specifically, we first prompt an LRM to emit a Verilog module that satisfies a given specification in TLSF. We intentionally stay at a high abstraction level for both specification and implementation, allowing the LRM to use more of the reasoning context window to iterate and refine solutions. To handle both realizable and unrealizable specifications, we instruct the model to either generate a Verilog module satisfying the specification or to produce a Verilog module serving as an environment strategy. The resulting Verilog module is then automatically verified by a model checker. In case the specification is violated, we provide the counterexample to the LRM as feedback creating a loop in which sound symbolic feedback complements the LRM's unsound reasoning.

In Table 1, we compare with the winner and runner-up of the 2025 SYNTCOMP. We evaluated both Gemini 3.1 Pro [5] and GPT-5.5 [10] configured with their highest reasoning setting. Both LRMs clearly outperform state-of-the-art tools, with GPT-5.5 performing best and solving 170 more instances than `ltlsynt`. The results show that the models benefit from sound feedback from a model checker. The effect is most pronounced for the Gemini model, with 162 additional instances solved after including the counterexamples provided by the model checker. In comparison with prior work by Egolf et al. [4], we highlight that supporting unrealizable specifications, generating Verilog modules, providing counterexamples, and using higher reasoning budgets are key.

Table 1. Reactive synthesis results on SYNTCOMP'25 comparing LRMs with algorithms `SemML` and `ltlsynt`. LRMs were run with their highest reasoning configurations.

Type	Name	CEX Iterations	Solved
Algorithm	`SemML` [8]		1295 / 1586
	`ltlsynt` [11]		1297 / 1586
LRM	`GPT-5` [4]		229 / 1586
	`Gemini 3.1 Pro`	0	1193 / 1586
		1	1311 / 1586
		2	1355 / 1586
	`GPT-5.5`	0	1392 / 1586
		1	1449 / 1586
		2	**1467** / 1586

3 Reactive Synthesis Beyond Decidability and Temporal Logics

Beyond classical reactive synthesis, we show that LRMs can generalize the problem in two important aspects: (a) synthesizing implementation templates for parameterized specifications; (b) starting from informal natural-language descriptions, circumventing the need for a formal logic specification.

Parameterized Synthesis. We follow the method for reactive synthesis and modify the LRM instruction to generate a parameterized Verilog module. Importantly, we can no longer automatically verify the parameterized implementation and therefore resort to testing individual parameter values. We still obtain a sound counterexample that we provide to the LRM as feedback, however, we can no longer guarantee correctness in the positive case. For evaluation, we derived 57 parameterized specifications from the SYNTCOMP benchmarks. Gemini 3.1 Pro and GPT-5.5 are both able to find parameterized implementations for 35 of the 57 benchmarks. The Gemini model benefits from additional verification feedback, solving up to 38 instances. The task of finding a parameterized implementation seems only moderately harder for LRMs than finding an implementation for a specific parameter value. For comparison, the models solve up to 41 instances when instantiating the problems with a parameter value.

Synthesis from Natural Language. With LRMs, we can now address the entire reactive-synthesis pipeline – from informal requirements to formal temporal-logic specifications to verified implementations – end-to-end. First, we manually authored a dataset of natural-language specifications corresponding to instantiations of the parameterized specifications introduced above. Second, we perform natural synthesis by 1) autoformalizing the natural-language specification into a formal specification in TLSF format and then synthesizing a Verilog module, and 2) prompting the LRM to directly synthesize a Verilog module from the natural-language specification. Our results show that informal descriptions are a viable starting point, with autoformalization performing comparably to direct synthesis. With Gemini 3.1 Pro, 31 out of 57 are synthesized correctly for the direct route, 30 for the autoformalized route. On the same instances (but formal specifications) the best algorithmic tools stand at 19, while LRMs at 38.

4 Conclusion

We introduced *natural synthesis*, a neuro-symbolic approach to reactive synthesis that combines Large Reasoning Models with counterexample-guided reasoning. Our pipeline substantially outperforms state-of-the-art algorithms on SYNTCOMP, extends to parameterized synthesis, and supports synthesis from natural-language descriptions through autoformalization and direct circuit generation, all verified against temporal specifications. These results highlight the potential of neuro-symbolic methods to bring reactive synthesis into real-world hardware design workflows.

Acknowledgments. This work was partially supported by the European Union with ERC Grant HYPER (No. 101055412). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Cosler, M., Hahn, C., Mendoza, D., Schmitt, F., Trippel, C.: nl2spec: Interactively translating unstructured natural language to temporal logics with large language models. In: Enea, C., Lal, A. (eds.) Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II. pp. 383–396. Lecture Notes in Computer Science, Springer (2023). https://doi.org/10.1007/978-3-031-37703-7_18, https://doi.org/10.1007/978-3-031-37703-7_18
2. Cosler, M., Schmitt, F., Hahn, C., Finkbeiner, B.: Iterative circuit repair against formal specifications. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net (2023), https://openreview.net/forum?id=SEcSahl0Ql
3. DeepSeek-AI: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. CoRR **abs/2501.12948** (2025). https://doi.org/10.48550/ARXIV.2501.12948, https://doi.org/10.48550/arXiv.2501.12948
4. Egolf, D., Zhou, Y., Tripakis, S.: Can llms perform synthesis? CoRR **abs/2603.20264** (2026). https://doi.org/10.48550/ARXIV.2603.20264, https://doi.org/10.48550/arXiv.2603.20264
5. Google: Gemini 3.1 Pro Preview (Feb 2026), https://deepmind.google/models/model-cards/gemini-3-1-pro/
6. Jacobs, S., Pérez, G.A., Abraham, R., Bruyère, V., Cadilhac, M., Colange, M., Delfosse, C., van Dijk, T., Duret-Lutz, A., Faymonville, P., Finkbeiner, B., Khalimov, A., Klein, F., Luttenberger, M., Meyer, K.J., Michaud, T., Pommellet, A., Renkin, F., Schlehuber-Caissier, P., Sakr, M., Sickert, S., Staquet, G., Tamines, C., Tentrup, L., Walker, A.: The reactive synthesis competition (SYNTCOMP): 2018-2021. Int. J. Softw. Tools Technol. Transf. **26**(5), 551–567 (2024). https://doi.org/10.1007/S10009-024-00754-1, https://doi.org/10.1007/s10009-024-00754-1
7. Jacobs, S., Pérez, G.A., Schlehuber-Caissier, P.: The temporal logic synthesis format TLSF v1.2. CoRR **abs/2303.03839** (2023). https://doi.org/10.48550/ARXIV.2303.03839, https://doi.org/10.48550/arXiv.2303.03839
8. Kretínský, J., Meggendorfer, T., Prokop, M., Zarkhah, A.: Semml: Enhancing automata-theoretic LTL synthesis with machine learning. In: Gurfinkel, A., Heule, M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 31st International Conference, TACAS 2025, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2025, Hamilton, ON, Canada, May 3-8, 2025, Proceedings, Part I. pp. 233–253. Lecture Notes in Computer Science, Springer (2025). https://doi.org/10.1007/978-3-031-90643-5_12, https://doi.org/10.1007/978-3-031-90643-5_12

9. OpenAI: Openai o1 system card. CoRR **abs/2412.16720** (2024). https://doi.org/10.48550/ARXIV.2412.16720, https://doi.org/10.48550/arXiv.2412.16720
10. OpenAI: GPT-5.5 (Apr 2026), https://openai.com/index/gpt-5-5-system-card/, snapshot gpt-5.5-2026-04-23
11. Renkin, F., Schlehuber-Caissier, P., Duret-Lutz, A., Pommellet, A.: Dissecting ltlsynt. Formal Methods Syst. Des. **61**(2), 248–289 (2022). https://doi.org/10.1007/S10703-022-00407-6, https://doi.org/10.1007/s10703-022-00407-6
12. Schmitt, F., Cosler, M., Metzger, N., Siber, J., Krsmanovic, V., Ghanem, M., Finkbeiner, B.: Natural synthesis: Outperforming reactive synthesis tools with large reasoning models (2026), https://arxiv.org/abs/2605.15131
13. Schmitt, F., Hahn, C., Rabe, M.N., Finkbeiner, B.: Neural circuit synthesis from specification patterns. In: Ranzato, M., Beygelzimer, A., Dauphin, Y.N., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual. pp. 15408–15420 (2021), https://proceedings.neurips.cc/paper/2021/hash/8230bea7d54bcdf99cdfe85cb07313d5-Abstract.html
14. Wu, Y., Jiang, A.Q., Li, W., Rabe, M.N., Staats, C., Jamnik, M., Szegedy, C.: Autoformalization with large language models. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022 (2022), http://papers.nips.cc/paper_files/paper/2022/hash/d0c6bc641a56bebee9d985b937307367-Abstract-Conference.html