# *Project Proposal:* Keeping LLMs in Check by Automated Reasoning*

Qiqi Jason Gu, Jan Hůla, and Mikoláš Janota

Czech Technical University in Prague, Prague, Czechia

## 1 Introduction

Large language models (LLMs) are demonstrably useful, but at the same time, there is a growing concern that their users easily obtain and use incorrect results [7].

In a recent experiment, Sutcliffe et al. suggest that LLMs can easily fail on basic logic puzzles [6]. The authors study an example where the LLM's answer is in fact correct but the reasoning steps are not. This is of course an issue because the users are also interested in the explanation but because the text looks convincing, they might easily overlook that the explanation is incorrect.

In this project we will focus on checking individual steps of an LLM by automated reasoners, such as satisfiability modulo theory solvers (SMT) [1] and automated theory provers (ATP) [4]. Our goal is to augment the LLM with a formal verification step that can be used to find errors in the reasoning steps of the LLM and use these errors as a feedback to improve the LLM.

## 2 Project Plan

Correctness checking of LLMs is a task of astronomical dimensions; control experiments are needed at this stage of the research. As a starting point, we will use the TPTP [5] puzzle category.[1] The advantage is that the problems contain an English description as well as its formalization. In fact, the experiment by Sutcliffe et al. already focuses on `PUZ001+1.p`, which deals with the "Who killed aunt Agatha?" puzzle [3].
The project will be carried out in the following steps.

1. Collect and generate puzzle problems.
2. Evaluate correctness of answers for various proprietary and open-source LLMs and check their failure modes.
3. Ask LLMs to generate formalization of steps and use ATPs to check them.
4. Integrate with *AI agent protocols* [9] to LLM directly use the reasoners as guidance. Inspirations will be sought in the work by Szeider, who used the *Model Context Protocol* to interact with constraint programming solvers [8].
5. Fine-tune a local LLM based on wrong answers.
6. Implement a learning loop in which the LLM generates solutions and learns from the feedback provided by the solver.

*Supported by the Czech MEYS under the ERC CZ project no. LL1902 *POSTMAN*, by the European Union under the project *ROBOPROX* (reg. no. CZ.02.01.01/00/22_008/0004590).

[1] https://tptp.org/cgi-bin/SeeTPTP?Category=Problems&Domain=PUZ

# 3 Preliminary Data

| Puzzle ID | ChatGPT | Claude | Gemini Flash | DeepSeek |
|---|---|---|---|---|
| 024-1 | Pass | Pass | Pass | Pass |
| 025-1 | Pass | Pass | Pass | Pass |
| 026-1 | Pass | Pass | Pass | Pass |
| 028-1 | Pass | Pass | Pass | Pass |
| 029-1 | Fail | | Fail | Fail |
| 031_10 | Pass | | Fail | Fail |
| 032-1 | Pass | | Pass | Pass |
| 033-1 | Pass | | Pass | Pass |
| 054-1 | Pass | Fail | Pass | Pass |
| 078+1 | Fail | Pass | | Fail |
| 082^1 | Pass | Pass | | Pass |
| 087^1 | Pass | Pass | | Fail |
| 088^5 | Pass | Pass | Pass | Pass |
| 128+1 | Pass | Fail | | Pass |
| 129+2 | Pass | Pass | | Pass |
| 130_10 | Pass | Pass | | Pass |
| 131_10 | Pass | Pass | Pass | Pass |
| 132+1 | Pass | Pass | Pass | Pass |
| 134_10 | Pass | Pass | Pass | Pass |
| 135_10 | Pass | Pass | Pass | Fail |
| 147^1 | Pass | Pass | Pass | Pass |
| 148^1 | Pass | Fail | Pass | Pass |
| 150^18 | Pass | Pass | Fail | Fail |
| **precision** | **91%** | **84%** | **82%** | **74%** |

Table 1: LLMs' answer correctness on 23 puzzles from TPTP

As a preliminary experiment, we collected 200 puzzles from TPTP, but only 23 of them have an English description. To test LLMs on these puzzles, we ran ChatGPT unpaid version on `chatgpt.com`, Claude AI 3.5 Sonnet on `claude.ai`, Gemini on `gemini.google.com`, and DeepSeek-V3-0324 on GitHub Models. Gemini 2.5 Pro has a low rate limit so we chose to use Gemini 2.5 Flash. ChatGPT gave correct answers to 91% queries. Claude AI 3.5 Sonnet solves 84%; Gemini 2.5 Flash 82%; DeepSeek 74%. The results are displayed in Table 1.

The results indicate that the existing LLMs perform well on the existing puzzles. However, currently, we do know not know whether the reasoning steps are also correct and how well the LLMs perform uner reformulations of the problems. Since the number of puzzles collected is low, we intend to generate new puzzles—this will be done by first generating a problem at the formal level that has a unique solution and the informalizing it, cf. [2].

# References

[1] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of*

*Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1267–1329. IOS Press, 2021. doi:10.3233/FAIA201017.

[2] Jasper Dekoninck, Mislav Balunovic, Nikola Jovanović, Ivo Petrov, and Martin Vechev. MathConstruct: Challenging LLM reasoning with constructive proofs. In *ICLR 2025 Workshop: VerifAI: AI Verification in the Wild*, 2025. URL: https://openreview.net/forum?id=nHW2tiGMrb.

[3] Francis Jeffry Pelletier. Seventy-five problems for testing automatic theorem provers. *Journal of Automated Reasoning*, 2(2):191–216, June 1986. URL: http://dx.doi.org/10.1007/BF02432151, doi:10.1007/bf02432151.

[4] John Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press, 2001.

[5] G. Sutcliffe. Stepping stones in the TPTP world. In C. Benzmüller, M. Heule, and R. Schmidt, editors, *Proceedings of the 12th International Joint Conference on Automated Reasoning*, number 14739 in Lecture Notes in Artificial Intelligence, pages 30–50, 2024. doi:10.1007/978-3-031-63498-7_3.

[6] Geoff Sutcliffe, Jack McKeown, and Alexander Steen. A chat with Bard. In Konstantin Korovin, Stephan Schulz, and Michael Rawson, editors, *Proceedings of the 14th and 15th International Workshops on the Implementation of Logics*, volume 21 of *Kalpa Publications in Computing*, pages 8–14. EasyChair, 2025. doi:10.29007/r9l2.

[7] Mirac Suzgun, Matthew Dahl, Daniel E. Ho, and Varun Magesh. Large legal fictions: Profiling legal hallucinations in large language models. *ArXiv*, abs/2401.01301, 2024. URL: https://api.semanticscholar.org/CorpusId:266725450.

[8] Stefan Szeider. MCP-Solver: Integrating language models with constraint programming systems, 2025. URL: https://arxiv.org/abs/2501.00539, arXiv:2501.00539.

[9] Yingxuan Yang, Huacan Chai, Yuanyi Song, Siyuan Qi, Muning Wen, Ning Li, Junwei Liao, Haoyi Hu, Jianghao Lin, Gaowei Chang, Weiwen Liu, Ying Wen, Yong Yu, and Weinan Zhang. A survey of AI agent protocols, 2025. arXiv:2504.16736.