



Benchmarking LLMs Against TPTP: A Vision for the Near Future

Andrija Urošević¹, Gordan Đurić², and Sana Stojanović Đurđević¹

¹ Faculty of Mathematics, University of Belgrade

andrija.urosevic@matf.bg.ac.rs; sana.stojanovic.djurdjevic@matf.bg.ac.rs

² École Polytechnique Fédérale de Lausanne (EPFL)

gordan.duric@epfl.ch

Abstract

We propose the TPTP World as a benchmark for evaluating LLMs' reasoning in which success is a TPTP derivation verified by independent verifiers. We describe the design of *Proof Assistant for Agents* (PAFA), an interface in which an AI agent states claims and premises, justified by a primitive rule or by automation of configurable strength. A finished proof is serialized directly as the TPTP derivation. We give the operation set for PAFA, a pipeline, and an evaluation protocol, and leave the underlying proof system, the automation boundary, and the proof state's retrieval mechanism open.

Motivation There is a huge effort in integrating machine learning models and, recently, Large Language Models (LLMs) in automated theorem proving. There are at least 147 articles since 2024 on the latter [6]. Many fall into the category of using LLMs within proof assistants. One successful result is LEANCOPLOT [9], which runs LLM inference inside LEAN to suggest the following proof step, select relevant premises, and even complete proofs. Recently, a significant result was achieved when Erdős Problem #707 was resolved through the construction of a counterexample, whose correctness was machine-verified by more than 6,000 lines of *vibe coded* LEAN formalization [1].

LLMs are evaluated and compared on different types of benchmarks, including mathematical benchmarks. At the moment, three of the most popular assessments are the MATH-500 dataset [7, 8], the MATHARENA platform [5], and the PUTNAMBENCH benchmark [16]. MATH-500 evaluates correctness by matching the model's output to the final numerical answer. However, a major flaw with this static approach is that newer LLMs may have already been exposed to solutions during training, which can lead to data contamination. MATHARENA takes a different approach, dynamically evaluating LLMs on the latest sets of problems. One of its benchmarks tests LLMs in a formal setting in LEAN, where they currently achieve significantly lower scores compared to standard natural language math evaluations. PUTNAMBENCH is a benchmark that evaluates LLMs on the Putnam competition problems in the proof assistants LEAN, ISABELLE, and ROCQ.

The Thousands of Problems for Theorem Provers (TPTP) World [13] is an ecosystem for ATP research. The TPTP problems [11] have several properties that make them attractive as

an LLM benchmark: (1) a difficulty rating between 0.0 and 1.0 calibrated against ATPs; (2) TPTP language supporting a hierarchy of logics for problems and solutions [12]; (3) solution verification [14]; (4) problems from diverse domains; and (5) a partial resistance to memorization and contamination. Moreover, the paper “A Chat with Bard” [15] illustrates that the LLM was capable of producing the right answer but made unsound arguments. This marks motivation for a benchmark that evaluates LLMs on the correctness of their reasoning rather than just the final answer.

Proposed Architecture We envision a pipeline in which LLMs interact with the TPTP World through an AI agent interacting with the *Proof Assistant for Agents* (PAFA), which serves as a layer between natural-language reasoning and formal reasoning: (1) PAFA parses a raw TPTP problem into a structured, agent-readable proof state; (2) an AI agent interacts with PAFA, extracts proof states, applies explicit logical steps, and receives updates or error messages; (3) once the proof is completed, PAFA outputs the TPTP derivation graph; and finally (4) one or more independent TPTP verifiers check the derivation graph.

We propose research directions in developing PAFA as an agent-focused proof environment that natively supports TPTP logic languages. Constructing a bidirectional translation layer between TPTP languages and the language of the chosen proof assistant will be significantly more challenging.

PAFA’s proof state is a partial directed acyclic graph (DAG), where each node is an annotated formula, as in TPTP derivation graphs. Once a node is *justified*, it carries an inference record. Nodes that have not yet been justified are *open*. This design enables backward reasoning from the conjecture, forward reasoning from the axioms, and the introduction of lemmas in any order, which can be wired in later. The proof state presented to the agent is a partial view over the open nodes and the proven facts, since both sets can grow as the derivation proceeds, eventually overflowing LLMs’ context window. A proof is complete when the conjecture and all of its ancestors are justified.

Operation Set The agent drives PAFA through the six operations listed below.

Operation	Signature	Effect
<code>get_state</code>	$() \rightarrow (\text{open}, \text{facts})$	A view over open nodes and proven facts.
<code>introduce</code>	$(\varphi) \rightarrow \text{NodeId}$	Adds a new, unjustified open node.
<code>fresh</code>	$(\text{kind}, \text{symbol}, \varphi) \rightarrow \text{NodeId}$	Adds an open node and checks freshness for definition, eigenvariable, or Skolem symbol.
<code>justify</code>	$(\text{NodeId}, \text{premises}, \text{rule}, \text{level}) \rightarrow \text{inference}$	Checks a primitive rule or performs automation up to the given level.
<code>retract</code>	$(\text{NodeId}) \rightarrow ()$	Reopens a justified node.
<code>complete</code>	$() \rightarrow \text{derivation or open nodes}$	Outputs the derivation if the conjecture’s ancestors are justified, else the open nodes.

Operations `fresh`, `justify`, and `retract` can fail on a declared symbol, an inapplicable rule, unsuccessful automation, a detected cycle, and on an unjustified or unknown node. Each failure is feedback for the agent. The `retract` operation reopens a justified node without deleting its old inference record, keeping the interaction log a single append-only ledger that doubles as a cost-accounting and failure-localization trail.

Open Questions PAFA fixes this interaction model and leaves three design choices open:

1. *Underlying proof system.* Resolution and superposition are closest to what existing TPTP verifiers already expect, but for the same reason, they are closest to published TSTP solutions, leading to a contamination tradeoff. Natural deduction is plausibly closer to LLM priors since pen-and-paper proofs, and hence most training data, are typically stepwise

deductive arguments. Sequent calculus allows more formulas to be justified at once, so it requires extending nodes (exactly what SC-TPTP [2] does) at the cost of a more complex interface. Proof-term systems check strongest against a small independent kernel, but that same machinery has the highest implementation cost for PAFA to output TPTP derivations.

2. *Automation boundary.* We envision discrete automation levels, fixed for the run: primitive rules only; plus propositional closure; plus ground equality; plus bounded matching; plus an external ATP call, each under a deterministic resource bound. Since rating 0.0 already means every state-of-the-art ATP solves the problem, the top level trivializes such problems and is informative only on harder ones. Left open is how an automation-closed gap is itself serialized into the TPTP derivation.
3. *The proof state’s relevance retrieval.* The agent sees only a relevance-scoped view over the open nodes and previously proven facts, so context growth is bounded by design rather than left to chance. Left open is the retrieval mechanism itself, symbol-based, DAG-proximity-based, or agent-queried, along with the soundness edge: if `justify` accepts any premise regardless of whether `get_state` ever surfaced it, retrieval quality is performance concern; if it does not, retrieval quality becomes part of what the verifier must trust.

On Soundness PAFA is not part of the trusted base, which comprises of: (1) the *leaf/root check* that pins the derivation’s leaves to the original problem file’s axioms and the root to the file’s conjecture; (2) the *verifiers*, and the ATPs they call for step verification are different from ATPs for level-5 automation; (3) the pipeline’s aggregation of verifier outputs. The operation set is correct by construction in the sense that no call adds an axiom or an unchecked fresh symbol, edits the root conjecture, closes an open node without a justification, or creates a cycle. PAFA bugs remain possible, which is why it relies on TPTP verifiers.

Each TPTP verifier outputs `VerifiedGood/VerifiedBad/Unknown/Timeout` in ProoVer’s vocabulary [3]. PAFA aggregates over TPTP verifiers’ outputs and labels each derivation as *verified* (≥ 1 `VerifiedGood`, no `VerifiedBad`), *rejected* (≥ 1 `VerifiedBad`, no `VerifiedGood`), *disputed* (both), or *unverified* (only `Unknown/Timeout`). The label *exploited* is a post-hoc audit on a disputed or verified case found to have gamed a verifier. Gaming a verifier is exactly ProoVer’s material, which charges -10 for accepting a bad proof. Hence, cheating is an output of the benchmark that can be useful for the ProoVer competition.

Status We draw on THEODORE [17], our own text-based natural-deduction assistant. In a preliminary experiment with The Natural Deduction Pack [4], both `codex` and `claude-code` produced THEODORE formalization and exported $\text{\LaTeX}$ proof tree arguments that we judged plausible upon reading.

As a null hypothesis, we prompted `claude-sonnet-5` with five (0.0 rated) FOF problems (due to limited resources) to produce TPTP derivations. Each derivation was verified successfully by GDV [10]. This shows that the pipeline works end to end on easy problems, but we argue that scaling requires PAFA.

Discussion An important long-term objective is to evaluate whether LLM agents can transfer reasoning strategies across different formal systems. If an agent can learn to reason through an abstract proof-state interface and successfully operate across multiple logics, which TPTP offers, this would provide stronger evidence for general formal reasoning capabilities. As LLMs continue to improve, they may eventually challenge even TPTP’s hardest, currently unsolved problems – which raises the question: will TPTP’s continual updates be able to deal with contamination successfully? And of course, the following question still remains: *When will LLMs conquer the whole TPTP World?*

References

- [1] Boris Alexeev and Dustin G. Mixon. Forbidden sidon subsets of perfect difference sets, featuring a human-assisted proof, 2025.
- [2] Julie Cailler and Simon Guilloud. Sc-tptp: An extension of the tptp derivation format for sequent-based calculus. In *International Conference on Automated Deduction*, pages 325–340, 2025.
- [3] Julie Cailler and Simon Guilloud. ProoVer competition 2026. <https://proover-competition.github.io/competitions/2026/>, 2026. Competition held at FLoC 2026, 27 July 2026.
- [4] Alastair Carr. *The Natural Deduction Pack*. University of Oxford, March 2018. Supplementary material for *The Logic Manual* by Volker Halbach.
- [5] Jasper Dekoninck, Nikola Jovanović, Tim Gehringer, Kári Rognvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. *arXiv preprint arXiv:2605.00674*, 2026.
- [6] Noah Gießing, Moritz Schubotz, and Hamid Rahkooy. A survey on LLMs in automated theorem proving. *Preprint*, 2026. Available on ResearchGate, March 2026.
- [7] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2021.
- [8] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- [9] Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Lean copilot: Large language models as copilots for theorem proving in lean. *arXiv preprint arXiv:2404.12534*, 2024.
- [10] Geoff Sutcliffe. Semantic derivation verification: Techniques and implementation. *International Journal on Artificial Intelligence Tools*, 15(6):1053–1070, 2006.
- [11] Geoff Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.
- [12] Geoff Sutcliffe. The Logic Languages of the TPTP World. *Logic Journal of the IGPL*, 2022.
- [13] Geoff Sutcliffe. Stepping Stones in the TPTP World. In C. Benz Müller, 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.
- [14] Geoff Sutcliffe, Frédéric Blanqui, and Guillaume Burel. Proof verification with gdv and lambdapi - it’s a matter of trust. *The International FLAIRS Conference Proceedings*, 38, 05 2025.
- [15] 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 *EPiC Series in Computing*, pages 8–14, 2024.
- [16] George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amityay Thakur, and Swarat Chaudhuri. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition, 2024.
- [17] Andrija Urošević and Gordan Đurić. Theodore: An Educational First-Order Logic Proof Assistant. <https://github.com/andrija-urosevic/theodore>, 2025. GitHub repository.