Building Benchmarks for Natural-Language Proof Verifiers

Chutong Yang*

Nov 3rd, 2026

Abstract

As LLMs become increasingly powerful at solving math problems. An important question is how well LLMs can detect errors in natural language proofs. We consider two kinds of errors. One kind is errors in purely human-written proofs, and the other is LLM-generated proofs. To address these, we describe a benchmark with corrections in arXiv history and LLM-altered synthetic proofs. We form a small dataset for both types as a demonstration, and our methodology can be generalized to larger-scale data generation. Using our dataset, we find that even models like Astra achieve human-expert-level proof generation and proof-checking, yet they are still not perfect at detecting proof errors.

1 Setting

A natural-language proof verifier reads a mathematical proof and identifies incorrect arguments or proof steps. In this work, we do not give LLM verifiers internet access. We run the verifiers in two settings: with the proof itself and together with previous work mentioned in the paper. We noticed that prior work provides limited help with error detection. In this work, we primarily studied `gpt-6-astra` and `gpt-5.6-sol`, which are the strongest models in terms of math reasoning ability.

For a historical paper, we include only references available before the version under review. For a synthetic proof, we exclude the paper from which it was adapted. We withhold solutions, error labels, and earlier reviews in both settings.

2 Build the Benchmark

The benchmark repository on GitHub lists the 62 historical papers and provides code for generating the synthetic proofs.

Historical errors from arXiv revisions. On arXiv, a paper's modification history is preserved and easy to access. Authors of a conference paper tend to submit high-quality drafts. However, mistakes are inevitable, and responsible authors tend to make modifications in newer versions. In many of these cases, the authors will mention the mistake they spotted and how they fixed the error. For data collection, we can follow these steps: 1. go through accepted papers of famous conferences and 2. check their revision history and find the papers with errors fixed. 3. use the early version to test the proof verifier.

*University of Texas at Austin, cyang98@utexas.edu

Synthetic errors. We start from a theorem in TCSAlgBench [3]. A domain expert or an LLM proposes a change to its proof, such as dropping an assumption, misusing a cited result, altering a parameter bound, or reporting a false computational certificate. An LLM can then draft the altered proof.

3 Experiment result

Our pilot compared `gpt-6-astra` and `gpt-5.6-sol`, both at maximum reasoning effort. We used our strategy mentioned above. The historical collection contained 62 papers from STOC, FOCS, SODA, and COLT 2020-2026. The synthetic collection contained 24 proofs from six families: 16 consistent planted error cases, seven main-theorem controls, and one specimen with inconsistent assumptions. In the proof only setting, Astra detected 51/62 historical targets and 13/16 consistent synthetic targets; Sol detected 44/62 and 12/16, respectively.

Historical error category	Astra max	Sol max
Missing hypotheses or overstated conclusions	9/10	7/10
False lemmas, identities, or inequalities	18/19	17/19
Omitted cases or incomplete proofs	3/6	3/6
Incorrect parameter or complexity bounds	9/12	8/12
Incorrect constructions	6/7	6/7
Multiple kinds of error	4/4	2/4
Correction insufficiently specified	2/4	1/4
Total	51/62	44/62

Table 1: Full historical target detections in the manuscript-only setting.

Synthetic proofs: the mathematical alteration. Table 2 groups the deliberately introduced errors. Independence mutations make random choices dependent while retaining an argument that needs independence. Asymptotic mutations use an estimate outside its valid parameter regime, for example when a previously fixed parameter varies with sample size. Other mutations change the norm or error criterion, strengthen a guarantee, replace a stable discretization, or omit the cost of writing the output. These are distinct mathematical tests, even when they originated in the same source paper.

What changed when prior work is provided? On 13 historical papers selected because Sol had missed the target, Sol improved from zero to one hit. Astra stayed at five hits. For the synthetic experiment, each model reran its own non-hit cases; neither model gained a full target hit.

4 Related work

Barkallah et al. [1] verify a proof by rewriting it as blocks, each with its own premises and conclusion, and checking each block independently. Their ArxivMathGradingBench uses errors that authors corrected in later arXiv revisions. We use a similar strategy to select historical examples from theoretical computer science. Hard2Verify [2] uses expert annotations to locate the first error in

Synthetic error category	Astra max	Sol max
False computational steps	2/4	1/4
Incorrect asymptotic or tail analysis	3/3	3/3
Invalid independence or conditioning	3/3	3/3
Invalid norm or error-metric conversion	1/2	1/2
Unjustified pointwise or last-iterate guarantees	2/2	2/2
Unstable discretization	1/1	1/1
Omitted output bit complexity	1/1	1/1
Total	13/16	12/16

Table 2: Full detections of consistent planted errors in the proof-only setting. Both models also have one partial detection in the norm/error-metric group. The two graph variants share a defect and are not independent examples.

model-generated solutions. In our synthetic examples, we choose the error in advance. We score whether the verifier identifies the specified error and, on selected cases, ask whether it can do so with access to the cited literature.

References

[1] Slim Barkallah, Luke Bailey, Kaiyue Wen, Mohammed Abouzaid, and Tengyu Ma. Pseudo-Formalization for Automatic Proof Verification. arXiv:2605.20531v1, 2026. https://arxiv.org/abs/2605.20531v1.

[2] Shrey Pandit, Austin Xu, Xuan-Phi Nguyen, Yifei Ming, Caiming Xiong, and Shafiq Joty. Hard2Verify: A Step-Level Verification Benchmark for Open-Ended Frontier Math. arXiv:2510.13744, 2025. https://arxiv.org/abs/2510.13744.

[3] Chutong Yang, Xiyuan Zhang, Yu Huang, Boran Han, Soonho Kong, Shuai Zhang, Vihang Prakash Patil, Zhen Han, Michael Bohlke-Schneider, and Bernie Wang. TCSAlgBench: Benchmarking Automated Proving for Research-Level Theoretical Computer Science. arXiv:2609.35606, 2026. https://arxiv.org/abs/2609.35606.