TCS-BENCH: Benchmarking State-of-the-Art Generative AI Theoretical Computer Science Research Ability

Vincent Cohen-Addad[*,†] Dimitris Paparas [*,†] Ernest van Wijland[*,‡,†]
Max Springer[§] Julien Canitrot-Paradis[¶] Honghao Lin[†] David Woodruff[†]
Adarsh Kumarappan[†,‖] Rajesh Jayaram[†] Rudrajit Das[†] Lalit Jain[†]
Ola Svensson[†] Silvio Lattanzi[†] Mislav Balunovic[**] Theophane Weber[**]
Vahab Mirrokni[†]

Abstract

We introduce TCS-Bench, a benchmark for evaluating Large Language Models (LLMs) on research-level Theoretical Computer Science (TCS) proof generation. TCS-Bench consists of theorem-proving tasks from papers published at top theoretical computer science venues (STOC, FOCS, and SODA). Each task provides the necessary context to derive a self-contained proof for a target result. We evaluate state-of-the-art models on this benchmark. We verify the correctness of generated proofs via a verification agent, and further benchmark the verifier against human-expert proof judgements on a set of target statements and generated proofs pairs. Our reference verifier achieves over 90% accuracy on the expert labeled set.

1 Introduction

Over the last two years, Large Language Models (LLMs) have achieved superhuman performance on a variety of standardized benchmarks, from professional exams to programming competitions [2, 4, 10, 17, 29, 30, 36]. These successes have extended into the formal domain of mathematics, where specially primed models have demonstrated the ability to solve problems at the level of an International Mathematical Olympiad (IMO) gold medalist [8], suggesting that LLMs can emulate rigorous mathematical logic.

Recently, these capabilities have crossed the threshold into open-ended scientific discovery with models actively contributing to expert-level mathematical and scientific breakthroughs. For example, Google's Gemini Deep Think and its advanced variants have successfully collaborated with human

[*]co-First author. Names appear in alphabetical order
[†]Google Research
[‡]CNRS, IRIF, Université Paris-Cité
[§]Princeton University, Department of Computer Science
[¶]Université Paris-Saclay, CEA, List, Palaiseau, France
[‖]California Institute of Technology, Department of Computer Science
[**]Google DeepMind

researchers to solve open problems, refute conjectures, and generate novel proofs across theoretical computer science, optimization, and physics [34],[11]. An internal version of OpenAI's ChatGPT disproved the unit-distance conjecture [22] and other long-standing mathematical and theoretical computer science problems [23]. Anthropic's models have similarly been utilized to independently discover counterexamples to long-standing conjectures [1].

However, these frontier-level successes highlight a critical limitation in how the AI community evaluates language models: a significant gap has emerged between what state-of-the-art agents can achieve in active research and what our standardized benchmarks can actually measure. We find the research field to be a rich, sustainable, and challenging testbed for evaluating the next generation of language models. Existing benchmarks, while valuable, fail to capture the core difficulties of real-world mathematical research for several key reasons. First, competition problems like those found in the IMO are typically self-contained, whereas research theorems are deeply embedded in the context of bespoke definitions, notations, and previously established lemmas. Second, research results frequently involve constructing a scaffold of interconnected results, a task that goes far beyond finding a single, clever insight. To truly measure progress, we need challenging benchmarks that more accurately reflect the work of human researchers.

1.1 Our Contributions

To this end, we introduce TCS-Bench, a new benchmark for evaluating an LLM's ability to prove theorems from cutting-edge Theoretical Computer Science (TCS) research papers. The task is grounded in scientific practice: a model is presented with a target statement extracted from a paper published at a top-tier conference, and is tasked with generating a proof. The challenging part is to provide all required "basic" mathematical context required to prove the target statement. Our key contribution is to provide a self-contained task that can be solved without access to the internet, and that evaluates the ability of the models to come up with a proof from first principles and the intermediate, state-of-the-art lemmas provided in the context. This is obtained by processing the paper the task is extracted from and the target proof.

Furthermore, a central aspect in our task generation process is the ability to generate harder and harder tasks by having a tight control over the generated context. Indeed, by masking intermediate Lemmas used in the proof of the target statement, the task becomes harder. One can thus increase the difficulty of the tasks by simply masking more and more intermediate results, all the way to define tasks that consist in proving the main result of the paper.

Then successfully resolving a task requires a model to comprehend the surrounding context, understand the intricate connections between lemmas and theorems, and generate a logically sound proof that would provide a similar amount of details as the peer-reviewed proof extracted from the paper. This format challenges models to perform the context-dependent reasoning that is a hallmark of scientific discovery. Our contributions are as follows:

- TCS-Bench, a benchmark composed of 300 theorem-proving tasks from papers published at the top theoretical computer science venues, FOCS, STOC, and SODA, between 2020 and 2026.
- We develop and validate an automated proof verification system, achieving more than 90% accuracy against human expert judgments on a held-out set of 100 human-labeled proofs.

Task Construction Overview. The foundation of our benchmark is a curriculum of statement-proving tasks from publicly available conference proceedings and that can and should be solved without requiring access to the internet. At a high level, we scrape and process the LaTeX source of a paper, build the dependency graph of the statements (theorems, lemmas, claims, etc), and for each of the statements we assemble multiple versions of the context needed to attempt the proof, hiding varying subsets of dependencies. Each task is then packaged as a self-contained JSON entry comprising three fields: a context, a target statement and a ground-truth proof. Full technical details of the benchmark construction are given in Section 3.

Benchmark Overview. Tasks are selected to span a broad range of TCS subfields and, critically, a broad range of difficulty levels. This stratification ensures that TCS-Bench is informative across the full capability spectrum of current and future models, rather than saturating at either extreme. A model is evaluated on TCS-Bench by generating a proof for each task in the benchmark and submitting those proofs to an automated (and validated) verifier. We specify a reference verifier and report all our baseline results using this to ensure reproducibility.

2 Related Work

Our work is situated at the intersection of benchmarks for mathematical reasoning, formal proof generation, and the broader evaluation of AI in scientific research workflows.

Competition Mathematics Benchmarks. A significant body of work has focused on evaluating LLMs on competition-style mathematics problems. Benchmarks in this area often draw from sources like the International Mathematical Olympiad (IMO), testing a model's ability to find clever insights for well-defined, self-contained problems [5, 10, 20, 21, 29]. A notable success in this domain is AlphaGeometry2, which demonstrated performance equivalent to an IMO gold medalist [14]. While these benchmarks are invaluable for measuring discrete problem-solving abilities, their self-contained nature does not reflect the challenges of genuine research [27]. In contrast, TCS-Bench sources its problems directly from published literature. This requires the model not just to solve a problem, but to reason within the rich, interdependent theoretical context established by the paper's definitions, notations and prior lemmas.

Research-Level Mathematical Benchmarks. Recent work has begun to address the gap between competition mathematics and genuine research. HorizonMath [32] presents over 100 predominantly unsolved problems from computational and applied mathematics, leveraging a generator-verifier gap where solutions are hard to produce but efficient to verify through numerical comparison or deterministic constraint checking. LemmaBench [26] takes a complementary approach by automatically extracting and contextualizing lemmas from recent arXiv preprints, creating an updatable benchmark immune to contamination through continuous refreshment from new publications. BrokenMath [25] evaluates a different failure mode—sycophancy in theorem proving—by perturbing valid mathematical statements into plausible but false versions, revealing that even frontier models attempt to prove false statements 29-70% of the time. TCS-Bench shares with these works the focus on research-level mathematics beyond competition problems, but differs in targeting proof completion within the rich contextual dependencies of published theoretical computer science papers,

where success requires not just solving isolated problems but reasoning within an interconnected scaffold of definitions and prior results.

Formal Theorem Proving. Another major direction of research focuses on benchmarking LLMs for proof generation in formal languages using interactive theorem provers like Lean or Coq [4, 6, 10, 18, 19]. This work evaluates a model's ability to generate sequences of tactics to construct a machine-verifiable proof, often within established formal mathematics libraries. Similarly, benchmarks like MathConstruct test a model's ability to generate a specific mathematical object whose properties can be formally verified by an automated function [9]. This line of work is critical for ensuring logical rigor. TCS-Bench complements these efforts by focusing on a different but equally important task: generating complete, human-readable proofs in natural language. The evaluation challenge in our work shifts from formal machine-verifiability to assessing logical soundness within the implicit argumentative structure of a research paper—a task that is more aligned with the daily workflow of human mathematicians.

AI for Scientific Discovery. As LLMs advance, the focus of evaluation is shifting from solving established problems to assessing their potential to accelerate scientific discovery [11, 12, 16, 33, 34, 35]. Recent work has demonstrated their utility across numerous domains, from chemistry and biology to astrophysics [7, 13, 15, 24, 31]. Our work aligns with a new class of benchmarks designed to evaluate AI agents on complex, long-horizon scientific tasks. For example, PaperBench evaluates an agent's ability to replicate an entire AI research paper, a process that includes understanding the paper, developing a codebase, and executing experiments to reproduce its empirical results [28]. This paradigm assesses a model's practical utility in a realistic research workflow. TCS-Bench adapts this "research-as-benchmark" paradigm to the domain of theoretical computer science. While PaperBench focuses on replicating empirical results, TCS-Bench is the first to focus on replicating theoretical contributions. By isolating the task to proof synthesis within the context of a research paper, our benchmark provides a targeted measure of the sophisticated, context-dependent reasoning required to contribute to the frontiers of mathematical knowledge.

3 The TCS Benchmark

TCS-Bench evaluates a model's ability to prove theorems drawn from cutting-edge theoretical computer science research. Each task in the benchmark is a self-contained proof-completion problem wherein the model receives a curated context comprised of definitions, prior lemmas and condensed external references. Given this primer knowledge together with a target statement, the model is tasked with producing a complete proof. Crucially, all tasks are derived from published papers whose proofs have been systematically removed, so that success requires chains of complex mathematical reasoning rather than simple memorization and recall.

3.1 Data Acquisition and Preprocessing

We construct the benchmark from papers published at FOCS, STOC, and SODA, between 2020 and 2026. For each paper, we obtain the LaTeX source from arXiv, limiting our dataset to papers released under permissive licenses (CC-0 or CC-BY-4.0) to ensure all benchmark content is legally distributable.

Research proofs routinely invoke results from prior literature. To ensure tasks are self-contained, we skip target statements whose proofs invoke external results without restating them.

3.2 Dependency Structure Construction

The core of our benchmark construction is a structural analysis that extracts the logical dependency structure of each paper. This process produces a directed acyclic graph (DAG) over all formal statements, where an edge from statement A to statement B indicates that the proof of B depends on A.

To extract this graph, we first use a deterministic LaTeX parser to identify all theorem-like environments (theorem, lemma, definition, corollary, etc.) and assign each a unique identifier. We then employ an LLM-based analysis pass to:

- Map each proof environment to the statement it proves (non-trivial since proofs may appear out of order or span multiple environments)
- Analyze the full paper to construct dependency edges—identifying when the proof of statement B invokes statement A

We programmatically verify acyclicity and reject any paper for which the extracted graph contains a cycle, as this indicates an extraction error.

From the DAG, we compute the **rank** of each statement as the length of the longest directed path terminating at that node. Definitions and axioms have rank 0, and each subsequent layer of derived results increments the rank. This ranking serves two purposes: it stratifies tasks by difficulty (higher-rank proofs require reasoning about longer chains of dependencies), and it prevents information leakage during context assembly (we can systematically hide all results of rank $\geq r$ when constructing a task for a rank-r statement).

Figure 1 illustrates an example dependency DAG extracted from a paper. The graph shows how a main theorem (rank 3) depends on intermediate lemmas (ranks 1-2), which in turn depend on foundational definitions (rank 0). Each blue node represents a proof-completion task in our benchmark. Note that only statements that have corresponding `\begin{proof}\end{proof}` tags in the paper are turned into tasks for our benchmark.

3.3 Task Construction

Each proof-completion task consists of three components: a *context*, a *target statement*, and a *ground-truth proof* (withheld during evaluation). The key challenge in constructing high-quality tasks is producing a context that is both self-contained (containing all information logically necessary to derive the proof) and concise enough to fit within standard context windows. We achieve this through the following procedure.

Initial Context Assembly. For a target statement s of rank r, we initialize the context by concatenating (i) all resolved external reference digests, (ii) the paper's text truncated at the start of s's proof, with two categories of redaction applied: all statements of rank $\geq r$ are hidden to prevent information leakage from later results, and all proof environments are removed. If any dependency of s falls outside the truncated portion of the paper, we re-insert its statement into the context.

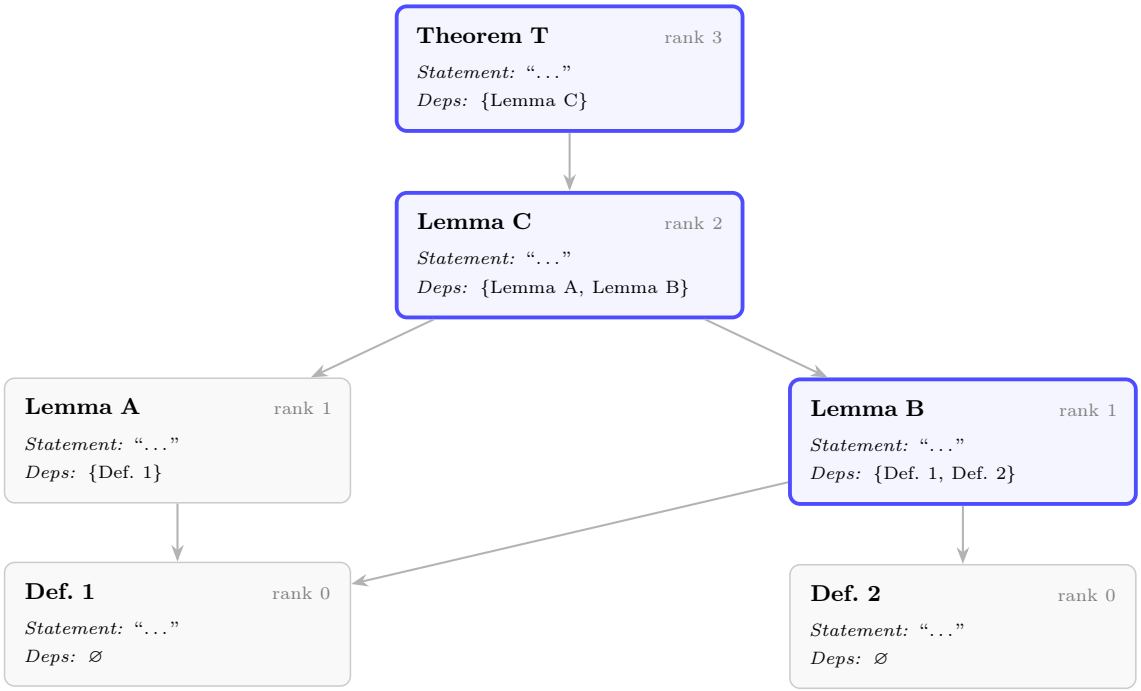

Figure 1: DAG example

Scalable Difficulty Tasks. To create tasks spanning a range of difficulties, we exploit the dependency structure to generate multiple variants of each proof task. Starting from a base task where all dependencies are provided in the context, we systematically withhold intermediate results, requiring the model to discover and prove them on the way to the main target.

Example. Consider proving Theorem T from Figure 1, which depends on Lemma C, which in turn depends on Lemmas A and B. The base difficulty for prompting a model to construct a proof would be to supply the model with all dependent results within the context (as depicted in Figure 2). Furthermore, we can easily increase the task complexity by omitting a subset of the dependencies (example prompting in Figure 3). Thus, forcing the model to prove such intermediary results along the way to the final claim. This procedure generates a spectrum of difficulties: at one extreme, all dependencies are provided and the model need only combine them; at the other, the model must reconstruct substantial portions of the paper's proof architecture.

Context Compression. To make the tasks short enough to fit the standard context window of $10,000$ tokens, we apply the following procedure.

First, we conduct *iterative section pruning*. Since many papers contains sections (e.g. related work, motivating context) that are irrelevant to a given target statement, we iteratively parse the section hierarchy of the assembled context and prompt an LLM to identify sections/subsections/-subsubsections that are entirely irrelevant to the target problem and its proof. Identified sections are removed, and the process repeats until no further pruning is possible. Second, we apply an *LLM*

[CONTEXT]
...
Lemma A....
Lemma C. *The sequence* $(u_n)_{n\in\mathbb{N}}$ *is upper-bounded.*
In the following, consider $\epsilon > 0$... ...

[TARGET STATEMENT]
Theorem T. *Algorithm 2 terminates in polynomial time.*

[GROUND-TRUTH PROOF]
Proof. We start by proving by induction that at the i-th iteration of the while-loop, at most u_i recursive calls are made.
...
Hence, by Lemma C, Algorithm 2 terminates in polynomial time. □

Figure 2: Base difficulty task (all dependencies provided)

shortner which, following section pruning condenses the remaining context (ie. shortening verbose passages or tightening exposition) while preserving all mathematically essential content.

3.4 Quality Filtering

The context assembly and compression phases can potentially lead to some essential notations or definitions to be removed from the context, either because of the LLM calls, or because the paper's structure is not well-suited to our dependency trimming. The final stage applies a comprehensive set of automated checks to ensure that each task is well-posed, self-contained, and free of information leakage. A task is rejected if any of the following conditions hold.

Structural Checks.

(i) The statement of ground-truth proof contains a figure or image reference, which cannot be faithfully represented in a text-only task.

(ii) Any label *introduced* (ie. via the `label` command) in the proof or target statement also appears in the context, indicating a potential leak of proof content.

(iii) The assembled task exceeds the 10,000 tokens limit after all compression passes.

(iv) The dependency set of the target statements is not fully covered by the union of the context and the ground-truth proof.

(v) Any label *referenced* (via `\ref` or `\cite`) in the proof or statement is introduced anywhere in the union of the context, statement, and proof, indicating an unresolved dependency.

`[CONTEXT]`
...
Lemma A....
In the following, consider $\epsilon > 0$... ...

`[TARGET STATEMENT]`
Theorem T. *Algorithm 2 terminates in polynomial time.*

`[GROUND-TRUTH PROOF]`
We start by proving:
Lemma C. *The sequence* $(u_n)_{n\in\mathbb{N}}$ *is upper-bounded.*
Proof of Lemma C. ...
Proof of Theorem T. We start by proving by induction that at the i-th iteration of the while-loop, at most u_i recursive calls are made.
...
Hence, by Lemma C, Algorithm 2 terminates in polynomial time. □

Figure 3: Task with omitted intermediary results to increase complexity.

Semantic Checks. Three independent LLM calls verify complementary aspects of task quality:

(vi) *Well-definedness*: all mathematical objects in the statement are defined in the context, and the statement is syntactically and semantically valid.

(vii) *Unambiguity and Correctness*: the target statement admits a unique interpretation given the context, and the ground-truth proof constitutes a valid proof of that statement.

(viii) *Context Coherence:* the context is mathematically coherent—it does not contain garbled text, broken references, or nonsensical artifacts introduced by the compression pipeline.

Tasks that fail any structural or semantic checks are excluded from the final benchmark.

3.5 Task Formulation Summary

Each released task provides the solver with:

1. A self contained LaTeX context ($\leq$ 10,000 tokens) comprising definitions, prior results (with proofs omitted), and condensed external reference digests.

2. A target statement to be proved.

The solver must produce a single, complete proof of the target statement, using LaTeX for mathematical notations. All justifications must follow from results present in the provided context and the solver is prohibited from accessing the original paper or external sources. The ground-truth proof is withheld and used exclusively for the evaluation.

4 Automated Proof Verification

A primary challenge in a benchmark like TCS-Bench is the need for a scalable and reliable method to evaluate the correctness of generated mathematical proofs. Manual verification by human experts is prohibitively slow and expensive. To address this, we developed a specialized verifier agent, tasked with deciding whether a candidate solutions constitutes a valid proof.

4.1 Verifier Design

The verifier's goal is to make a binary decision (correct or incorrect) on a candidate proof for a specific statement within a given context. It is aligned to tolerate trivial omissions common in academic literature (e.g., "the rest follows by simple algebra") but reject proofs with critical logical gaps or errors.

The verifier is provided with three key inputs: the task's context and target statement that were passed to the solver, and additionally the ground-truth proof. Then, four calls are made to Gemini 3.1 Flash, a cheap model, and a candidate proof is deemed correct if and only if at least three of the four verdicts mark it as correct. Our prompt is provided in Appendix A.1.

4.2 Verifier Calibration

To design the verifier prompt, we generated proofs by running the solver on a set of tasks, disjoint from the benchmark. Then, human experts reviewed them, and produced a set of 50 correct proofs and 50 incorrect proofs. Finally, we ran the GEPA [3] improvement pipeline on this alignment task to produce a prompt that achieves an accuracy of more than 90%.

5 Experimental Setup

We evaluate frontier language models on TCS-Bench to establish baseline performance on research-level mathematical proof generation. Our evaluation focuses on measuring the current capabilities of state-of-the-art models when presented with proof-completion tasks drawn from cutting-edge theoretical computer science research.

5.1 Experimental Setup

We assess the following frontier models: Gemini 3.1 Pro and Gemini 3.1 DeepThink, Opus 5, and GPT 5.6 Pro. We also present an internal harness, Colosseumthat we evaluate with both Gemini 3.1 Pro, Gemini 3.6 and a combination of both. All models were accessed via their respective API endpoints or web interfaces using the most recent versions available at the time of evaluation. For each task, we query the model with the context and target statement as described in Section 3. We provide the exact prompt formatting in appendix A.1.

For models offering extended reasoning capabilities, we use the maximum publicly available thinking budget to allow models to fully explore the problem space. Generated proofs are evaluated using our automated verifier (Section 4). Each model attempts all 300 tasks in TCS-Bench.

Table 1: Model performance on TCS-BENCH measured by accuracy.

Model	**Accuracy** (↑)
Opus 5	32.77
Gemini 3.1 Pro	30.3
Gemini 3.1 DeepThink	52
GPT 5.6 Pro (max)	68

Results & Analysis. Table 1 presents the overall performance of each model on TCS-BENCH. The strongest performing model, GPT 5.6 Pro, achieves an accuracy score of 68%, successfully proving 204 of 300 tasks. Notably, Opus 5, exhausts its token budget of 128K tokens before obtaining an answer for 162 out of the 300 tasks, which negatively affects its performance.

Results & Analysis with COLOSSEUM The results above evaluate base models directly. We also report results for COLOSSEUM, an agentic proof-search harness we run on top of a base model: for each problem it explores several candidate proof strategies, decomposes the target statement into subproblems, solves them, and assembles and revises a final proof. We do not describe COLOSSEUM in detail here; the only property that matters below is that it is parameterised by its base model, so the same pipeline can be run on different models to obtain independent proofs of the same problem. COLOSSEUM also carries an internal verifier, which may decline to certify the proof it has produced; a declined run still submits its best proof, so every problem receives a submission from every arm.

Selecting between two runs. Running COLOSSEUM on two different public base models yields two independent proofs of every problem, and we select one of them automatically. Writing M_1 for Gemini 3.1 Pro and M_2 for Gemini 3.6 Flash, we submit M_2's proof when either

1. COLOSSEUM's internal verifier declined to certify M_1's proof, or
2. at most half of 8 independent critiques drawn from M_2 judge M_1's proof to be correct,

and M_1's proof otherwise.

This critic is a component of COLOSSEUM and is distinct from the automated grader of Section 4. The grader is used only to score final submissions; the selection has no access to it, to ground truth, or to any human judgment. Prompts and per-run inference budget are unchanged from the single-model setting.

On TCS-BENCH, the rule selects Gemini 3.1 Pro on 167 problems, with an accuracy of 84% compared with 54% before filtering, and Gemini 3.6 Flash on the remaining 133, with an accuracy of 47%.

Why cross-model. One might attribute the gain simply to a model being a poor judge of its own proofs. The results suggest a more specific picture: a model's self-*acceptance* carries little information, whereas its self-*rejection* is highly informative. Among cases where Gemini 3.1 Pro's verifier accepts its own proof, 93.4% receive unanimous acceptance across verifier runs, leaving little room for the acceptance signal to distinguish among proofs. In contrast, routing on self-rejection alone raises accuracy from 54.0% to 63.7%. Cross-model critique provides a second, largely independent signal precisely on the proofs that self-verification fails to distinguish. Gemini 3.6

	Solved / 300	Accuracy
Colosseum, Gemini 3.1 Pro	162	54.0%
Colosseum, Gemini 3.6 Flash	140	46.7%
Colosseum, cross-model selection between the two	203	67.7%
Oracle best-of-two (upper bound, not achievable)	217	72.3%

Table 2: Colosseum run on two different public base models, with automated cross-model selection between the resulting proofs. All rows are graded by the automated grader of Section 4. The oracle row reports the fraction of problems solved by at least one of the two runs; it requires knowing the answer and upper-bounds any selection rule.

Flash's critiques of Gemini 3.1 Pro's proofs separate correct from incorrect proofs with an AUC of 0.854, and incorporating this signal yields a further 4.0-point gain, bringing accuracy to 67.7%. The two signals are complementary rather than redundant. On 47 problems, the critique overturns a Gemini 3.1 Pro proof that its own verifier had accepted; among these cases, Gemini 3.6 Flash is correct 21 times, compared with only 9 for Gemini 3.1 Pro. The direction of cross-model critique matters more than the precise threshold. Gemini 3.6 Flash judging Gemini 3.1 Pro achieves an AUC of 0.854, whereas Gemini 3.1 Pro judging Gemini 3.6 Flash provides a weaker signal, with an AUC of 0.637. We therefore fix the stronger direction, selected using cross-model critique accuracy measured on TCS-Bench. By contrast, performance is largely insensitive to the critique threshold: accuracy remains between 67.3% and 67.7% for every threshold from 0/8 through 5/8.

While we lack comprehensive human baselines, we note that all tasks in TCS-BENCH have ground-truth proofs published by domain experts. The gap between the strongest model 68% and perfect performance (100%) represents the current frontier of automated mathematical reasoning.

6 Discussion

We have introduced TCS-Bench, the first benchmark evaluating LLMs on proof generation from cutting-edge theoretical computer science research. Unlike competition mathematics benchmarks testing isolated problems, TCS-Bench requires reasoning within the rich contextual scaffolding of research papers—navigating bespoke definitions, dependency structures, and chains of intermediate results across 300 tasks spanning multiple difficulty levels.

A key contribution is our automated proof verification system achieving over 90% accuracy against human expert judgments, enabling scalable evaluation without prohibitive manual verification costs. The verifier tolerates stylistic variations common in mathematical writing while rigorously detecting logical gaps—a balance essential for meaningful evaluation.

Our evaluation of five frontier models shows the strongest systems: Gemini 3.1 DeepThink solves 52%, GPT-5.6-Pro solves 68% and Colosseum with a cross-model approach solves 67.7%. While this demonstrates meaningful progress in automated mathematical reasoning, the gap to perfect performance reveals substantial room for advancement. The stable performance ceiling across all models suggests current architectures face fundamental limitations in constructing multi-step mathematical arguments within complex dependencies.

We lastly note that TCS-Bench is designed for longevity through continual addition of new papers that postdate model training cutoffs, scalable difficulty via dependency-hiding that spans

easy to extremely challenging variants, and rank-based stratification enabling fine-grained progress tracking.

References

[1] Claude Fable 5. The jacobian conjecture is false. `https://x.com/__alpoge__/status/2079028340955197566`, 2026. Accessed: 2026-08-07.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

[4] Team AlphaProof and Team AlphaGeometry. AI achieves silver-medal standard solving international 178 mathematical olympiad problems. *DeepMind blog*, 179:45, 2024.

[5] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on Uncontaminated Math Competitions. *arXiv preprint arXiv:2505.23281*, 2025.

[6] Barış Bayazıt, Yao Li, and Xujie Si. A Case Study on the Effectiveness of LLMs in Verification with Proof Assistants. *arXiv preprint arXiv:2508.18587*, 2025.

[7] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

[8] Yuri Chervonyi, Trieu H Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang Nguyen, Marcelo Menegali, Junehyuk Jung, Vikas Verma, Quoc V Le, and Thang Luong. Gold-medalist performance in solving olympiad geometry with alphageometry2. *arXiv preprint arXiv:2502.03544*, 2025.

[9] 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.

[10] Jasper Dekoninck, Ivo Petrov, Kristian Minchev, Mislav Balunovic, Martin Vechev, Miroslav Marinov, Maria Drencheva, Lyuba Konova, Milen Shumanov, Kaloyan Tsvetkov, et al. The Open Proof Corpus: A Large-Scale Study of LLM-Generated Mathematical Proofs. *arXiv preprint arXiv:2506.21621*, 2025.

[11] Tony Feng, Trieu H. Trinh, Garrett Bingham, Dawsen Hwang, Yuri Chervonyi, Junehyuk Jung, Joonkyung Lee, Carlo Pagano, Sang hyun Kim, Federico Pasqualotto, Sergei Gukov, Jonathan N. Lee, Junsu Kim, Kaiying Hou, Golnaz Ghiasi, Yi Tay, YaGuang Li, Chenkai Kuang, Yuan Liu, Hanzhao Lin, Evan Zheran Liu, Nigamaa Nayakanti, Xiaomeng Yang, Heng-Tze Cheng, Demis Hassabis, Koray Kavukcuoglu, Quoc V. Le, and Thang Luong. Towards autonomous mathematics research, 2026.

[12] Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

[13] Kaixuan Huang, Yuanhao Qu, Henry Cousins, William A Johnson, Di Yin, Mihir Shah, Denny Zhou, Russ Altman, Mengdi Wang, and Le Cong. CRISPR-GPT: An LLM agent for automated design of gene-editing experiments. *arXiv preprint arXiv:2404.18021*, 2024.

[14] Yichen Huang and Lin F Yang. Gemini 2.5 pro capable of winning gold at imo 2025. *arXiv preprint arXiv:2507.15855*, 2025.

[15] Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1):015022, 2022.

[16] Moksh Jain, Tristan Deleu, Jason Hartford, Cheng-Hao Liu, Alex Hernandez-Garcia, and Yoshua Bengio. Gflownets for ai-driven scientific discovery. *Digital Discovery*, 2(3):557–577, 2023.

[17] Tiffany H Kung, Morgan Cheatham, Arielle Medenilla, Czarina Sillos, Lorie De Leon, Camille Elepaño, Maria Madriaga, Rimel Aggabao, Giezel Diaz-Candido, James Maningo, et al. Performance of chatgpt on usmle: potential for ai-assisted medical education using large language models. *PLoS digital health*, 2(2):e0000198, 2023.

[18] Vanessa Lama, Catherine Ma, and Tirthankar Ghosal. Benchmarking automated theorem proving with large language models. In *Proceedings of the 1st Workshop on NLP for Science (NLP4Science)*, pages 208–218, 2024.

[19] Zenan Li, Zhaoyu Li, Wen Tang, Xian Zhang, Yuan Yao, Xujie Si, Fan Yang, Kaiyu Yang, and Xiaoxing Ma. Proving Olympiad Inequalities by Synergizing LLMs and Symbolic Reasoning. *arXiv preprint arXiv:2502.13834*, 2025.

[20] Junqi Liu, Xiaohan Lin, Jonas Bayer, Yael Dillies, Weijie Jiang, Xiaodan Liang, Roman Soletskyi, Haiming Wang, Yunzhou Xie, Beibei Xiong, et al. CombiBench: Benchmarking LLM capability for combinatorial mathematics. *arXiv preprint arXiv:2505.03171*, 2025.

[21] Yujun Mao, Yoon Kim, and Yilun Zhou. CHAMP: A Competition-level Dataset for Fine-Grained Analyses of LLMs' Mathematical Reasoning Capabilities. *arXiv preprint arXiv:2401.06961*, 2024.

[22] OpenAI. An openai model has disproved a central conjecture in discrete geometry. `https://openai.com/index/model-disproves-discrete-geometry-conjecture/`, 2026. Accessed: 2026-08-07.

[23] OpenAI. Ten advances in mathematics and theoretical computer science. `https://openai.com/index/ten-advances-in-mathematics/`, 2026. Accessed: 2026-08-07.

[24] Liam Parker, Francois Lanusse, Siavash Golkar, Leopoldo Sarra, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Geraud Krawezik, Michael McCabe, Rudy Morel, et al. Astroclip: a cross-modal foundation model for galaxies. *Monthly Notices of the Royal Astronomical Society*, 531(4):4990–5011, 2024.

[25] Ivo Petrov, Jasper Dekoninck, and Martin Vechev. BrokenMath: A Benchmark for Sycophancy in Theorem Proving with LLMs. *arXiv preprint arXiv:2510.04721*, 2025.

[26] Antoine Peyronnet, Fabian Gloeckle, and Amaury Hayat. LemmaBench: A Live, Research-Level Benchmark to Evaluate LLM Capabilities in Mathematics. *arXiv preprint arXiv:2602.24173*, 2026.

[27] Inioluwa Deborah Raji, Emily M Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. Ai and the everything in the whole wide world benchmark. *arXiv preprint arXiv:2111.15366*, 2021.

[28] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, et al. Paperbench: Evaluating ai's ability to replicate ai research. In *Forty-second International Conference on Machine Learning*.

[29] Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

[30] Lakshmi Varanasi. Gpt-4 can ace the bar, but it only has a decent chance of passing the cfa exams. here'sa list of difficult exams the chatgpt and gpt-4 have passed. *Business Insider*, 5, 2023.

[31] Ricardo Vinuesa, Steven L Brunton, and Beverley J McKeon. The transformative potential of machine learning for experiments in fluid mechanics. *Nature Reviews Physics*, 5(9):536–545, 2023.

[32] Erik Y Wang, Sumeet Motwani, James V Roggeveen, Eliot Hodges, Dulhan Jayalath, Charles London, Kalyan Ramakrishnan, Flaviu Cipcigan, Philip Torr, and Alessandro Abate. Horizon-math: Measuring ai progress toward mathematical discovery with automatic verification. *arXiv preprint arXiv:2603.15617*, 2026.

[33] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

[34] David P. Woodruff, Vincent Cohen-Addad, Lalit Jain, Jieming Mao, Song Zuo, Mohammad-Hossein Bateni, Simina Branzei, Michael P. Brenner, Lin Chen, Ying Feng, Lance Fortnow, Gang Fu, Ziyi Guan, Zahra Hadizadeh, Mohammad T. Hajiaghayi, Mahdi JafariRaviz, Adel Javanmard, Karthik C. S., Ken ichi Kawarabayashi, Ravi Kumar, Silvio Lattanzi, Euiwoong Lee, Yi Li, Ioannis Panageas, Dimitris Paparas, Benjamin Przybocki, Bernardo Subercaseaux, Ola Svensson, Shayan Taherijam, Xuan Wu, Eylon Yogev, Morteza Zadimoghaddam, Samson Zhou, Yossi Matias, James Manyika, and Vahab Mirrokni. Accelerating scientific research with gemini: Case studies and common techniques, 2026.

[35] Yanbo Zhang, Sumeer A Khan, Adnan Mahmud, Huck Yang, Alexander Lavin, Michael Levin, Jeremy Frey, Jared Dunnmon, James Evans, Alan Bundy, et al. Exploring the role of large language models in the scientific method: from hypothesis to discovery. *npj Artificial Intelligence*, 1(1):14, 2025.

[36] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*, 2023.

A Omitted Details

A.1 Full Prompt Details

We present here the prompt formatting for requesting the evaluated model to prove a target statement:

```
   You are an expert mathematician.  Your task is to provide a thorough and
correct proof for a mathematical problem (your Target Problem).  This proof will
be used to benchmark your abilities as a mathematician.

To obtain your proof, you can assume and use any reference (lemma, theorem,
definition, etc) from the context below:

********** BEGIN CONTEXT **********

{CONTEXT}

********** END CONTEXT **********

********** BEGIN EVALUATION CRITERIA **********

Recall that you can assume and use any of the mathematical statements provided
in the context above.

Your proof must be complete and valid.  As a guideline, here is a non-exhaustive
list of criteria to evaluate your proof.  You should use it, together with any
additional criteria you can think of, to evaluate any candidate proof before your
final response.

Evaluation Criteria:

1.  **Logical Rigor and External Assumption Check**:

   - **No Unauthorized Constraints**:  The proof must not introduce arbitrary
     numerical constraints or "safety margins" to simplify the proof (e.g.,
     assuming k >= 2, assuming epsilon is sufficiently small, or assuming sets are
     non-empty) when such constraints are not explicitly stated in the target
     statement or the context.

   - **Strict Lemma Adherence**:  Every step must follow strictly from the
     provided context or standard mathematical knowledge.  Applying a theorem
     without explicitly verifying all its preconditions (as defined in the
     context) is a failure.

   - **Case Integrity**:  If the proof involves case-splitting (e.g., i* <= T vs
     i* = T + 1), the proof must address these specific boundaries.  Skipping a
```

boundary case or "merging" distinct logical paths via hand-waving results in a failure.

2. **Variable Alignment and Property Scope**:

   - **Property Scope Integrity**: Properties must only be applied to the specific variables for which they are defined. If the proof generalizes a property of a subset to a larger set then this constitutes a failure (e.g., applying a u-uniformity property defined for "outer queries" T_2 to the "total queries" T, or applying a property of a specific index j to all indices i without proof).

   - **Index and Set Precision**: The proof must maintain the integrity of sets (e.g., B vs B^*, S_i vs S_{i-1}) and indices. Misidentifying a variable or applying a property to the wrong time step or set is a fatal error.

3. **Semantic Integrity and Jargon Detection**:

   - **No Hallucinated Logic/Word Salad**: If the proof uses repetitive, nonsensical, or overly dense jargon to mask a lack of logical depth, then this is a failure. The proof must be linguistically coherent. If a paragraph consists of technical terms strung together without clear propositional logic (e.g., "mapping limits matching identical parameters constraints mapping"), it is a failure.

   - **No "Standard" Hand-waving**: The proof cannot skip non-trivial derivations by claiming they are "standard", "trivial", etc.

4. **Quantitative and Limit Accuracy**:

   - **Derivation Accuracy**: Any error in arithmetic, algebraic manipulation, or inequality direction will automatically invalidate the proof.

   - **Boundary and Floor/Ceiling Precision**: Bounds must be exactly supported. For example, if a floor function $\lfloor k^{-\epsilon} \rfloor$ is used, the proof must account for all valid values of k (including $k = 1$) unless the prompt restricts them. Failure to do so, invalidates the proof.

5. **Self-Containment**

   - **Citations**: Citing external papers to utilize their lemmas, theorems, or proofs is strictly forbidden to prevent hallucinations; unless both the citation and the referenced lemma, theorem, proof, etc, is explicitly stated in the context.

6. **Additional Criteria**

   - **Maximum scrutiny**: You must think of any additional criteria that the proof must meet to be correct and ensure your proof passes them. This is for your own benefit, to maximize the chances that your proof is correct so

```
    that you can pass the benchmark.  You have no incentive to avoid thinking of
    additional criteria because then you may miss a bug in the proof and fail the
    benchmark.

********** END EVALUATION CRITERIA **********

********** BEGIN FINAL RESPONSE FORMAT **********

Once you have completed your reasoning, have obtained a proof that passes all
evaluation criteria, and are ready to submit your final answer, your final output
should be the proof in latex format.

   - DO NOT include any other text, reasoning, or formatting in your final
     submission turn.

   - If you have the ability to store your answer in a file, DO NOT use it.  The
     proof should be in your final response.

********** END FINAL RESPONSE FORMAT **********

With this in mind, you are tasked to prove the following target statement.

Target Problem (your task):

{TARGET STATEMENT}
```

The following is the prompt used to query our verifier model to check a proofs correctness against a ground-truth result.

```
   # Instructions
   Your task is to evaluate whether a provided 'student_answer' correctly and
rigorously proves the 'Target Problem' using only the definitions and lemmas
found in the 'solve_promp'.  You must compare the student's logic, variable
tracking, and quantitative derivations against the 'ground_truth_proof'.

   Evaluation Criteria:

   1.  **Logical Rigor and External Assumption Check**:

   - **No Unauthorized Constraints**:  Mark as "0" if the student introduces
     arbitrary numerical constraints or "safety margins" to simplify the proof
     (e.g., assuming k >= 2, assuming ε is sufficiently small, or assuming sets
     are non-empty) when such constraints are not explicitly stated in the
     'solve_prompt'.

   - **Strict Lemma Adherence**:  Every step must follow strictly from the
```

provided lemmas. Applying a theorem without explicitly verifying all its preconditions (as defined in the ‘solve_prompt‘) is a failure.

- **Case Integrity**: If the ‘ground_truth_proof‘ relies on specific case-splitting (e.g., $i^* \leq T$ vs $i^* = T + 1$), the student must address these specific boundaries. Skipping a boundary case or "merging" distinct logical paths via hand-waving results in a "0".

2. **Variable Alignment and Property Scope**:

- **Property Scope Integrity**: Properties must only be applied to the specific variables for which they are defined. Mark as "0" if the student generalizes a property of a subset to a larger set (e.g., applying a u-uniformity property defined for "outer queries" T_2 to the "total queries" T, or applying a property of a specific index j to all indices i without proof).

- **Index and Set Precision**: The proof must maintain the integrity of sets (e.g., B vs B^*, S_i vs S_{i-1}) and indices. Misidentifying a variable or applying a property to the wrong time step or set is a fatal error.

3. **Semantic Integrity and Jargon Detection**:

- **No Hallucinated Logic/Word Salad**: Mark as "0" if the student uses repetitive, nonsensical, or overly dense jargon to mask a lack of logical depth. The proof must be linguistically coherent. If a paragraph consists of technical terms strung together without clear propositional logic (e.g., "mapping limits matching identical parameters constraints mapping"), it is a failure.

- **No "Standard" Hand-waving**: The student cannot skip non-trivial derivations by claiming they are "standard," "trivial," or "harmonious" if those steps rely on specific lemma interactions shown in the ground truth.

4. **Quantitative and Limit Accuracy**:

- **Derivation Accuracy**: Mark as "0" for any error in arithmetic, algebraic manipulation, or inequality direction.

- **Boundary and Floor/Ceiling Precision**: Bounds must be exactly supported. For example, if a floor function $\lfloor k^{-\epsilon} \rfloor$ is used, the student must account for all valid values of k (including $k = 1$) unless the prompt restricts them.

Output Format:

- If the ‘student_answer‘ is a valid, rigorous, and self-contained proof that correctly applies the provided materials, respects variable scopes, avoids external assumptions, and matches the logical depth/cases of the ground truth, output exactly "1".

- If the ‘student_answer‘ contains logical gaps, jargon-masking, unauthorized assumptions (like $k > 1$), scope errors (query type inflation), or calculation errors, output exactly "0".
- Do not provide any explanation, feedback, or additional text. Your response must be only "1" or "0".