PROOFCOUNCIL: AN LLM AGENT FOR SOLVING OPEN MATHEMATICAL PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have shown increasing promise in solving open problems in mathematics. However, their performance can be further improved through agentic workflows tailored to real-world mathematical practice. To this end, we introduce ProofCouncil, a mathematical agent that is designed to tackle open problems using an author-critic architecture. ProofCouncil served as a submission to the second batch of FirstProof, a challenge consisting of 10 real-world mathematical problems that agents must solve autonomously. Its submissions for 6 of the 10 problems were judged by the referees to be correct up to at most minor revisions, showing the best performance among participating teams. We also evaluate ProofCouncil on 30 open problems collected from mathematical researchers. Among the 21 solutions that received human feedback, 5 were judged completely correct, 2 more were judged promising pending final verification, and a further 8 contained useful partial progress. In this paper, we describe the development of ProofCouncil and the agent-building library used to create it, which we release as open source to the community.

1 INTRODUCTION

Large language models (LLMs) have recently demonstrated remarkable capabilities in mathematics by solving open problems of varying interest (Alon et al., 2026; Schmitt, 2025; Feng et al., 2026). This progress has been accompanied by the development of advanced agents designed for mathematical problem-solving (Feng et al., 2026; Zhao et al., 2026; Zheng et al., 2026; Ju et al., 2026), further highlighting the potential of LLMs in this domain. In this context, the second iteration of the FirstProof challenge (Abouzaid et al., 2026a;b) was launched to evaluate AI systems on a fixed set of 10 problems with no publicly available solution. Participants were required to submit an agent that autonomously attempted all 10 problems within a 24-hour window, using only publicly available models and tools.

In this work, we present ProofCouncil, a mathematical agent submitted to the FirstProof challenge. As shown in Fig. 1, ProofCouncil follows an author-critic architecture: an author agent iteratively writes and revises a proof, while a critic agent evaluates each version and provides feedback. At every round, the author may also request targeted assistance from two auxiliary sources: a council of additional LLMs outside the main author-critic loop, and a compute agent for computer algebra system (CAS) computations. Their responses are provided to the author in the next iteration. To maintain consistency and allow the critic to track progress, the critic generally retains its conversation history across rounds, but this history is reset every k rounds to provide a more independent review. If a freshly initialized critic accepts the proof, the proof is returned to the user.

We open-source both ProofCouncil and the underlying agent-building library. The library represents agentic systems as conditional directed acyclic graphs (DAGs), enabling flexible structures of calls between different agents and models. It also provides a simple interface for implementing new agents and workflows, enabling future researchers to build on our work and create their own agentic systems.

In the FirstProof challenge, ProofCouncil solved 6 of the 10 problems up to at most minor revisions and showed the best performance among participating teams. As a secondary evaluation, we also reached out to mathematical researchers and collected 30 open problems across various fields of mathematics. Solution attempts were sent to the question authors for feedback. Among 21 received responses, 5 were judged to be correct solutions, 2 were considered promising pending final

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

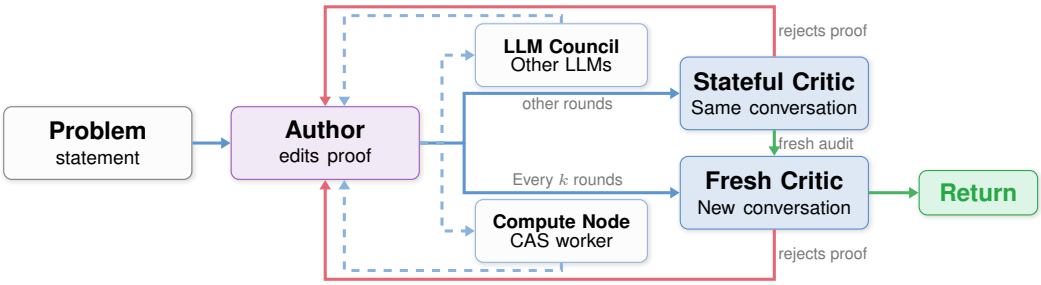

Figure 1: Overview of ProofCouncil. The author agent iteratively edits the proof in response to feedback from a critic. Every k rounds, the stateful critic is reset. The author may optionally request help from other LLMs or a compute node.

verification, and 8 more contained useful partial progress towards a solution. Notably, no expert reported that an output claiming to solve the stated problem was mathematically incorrect. However, two outputs were judged to have solved easier versions of the submitted problem, underscoring problem interpretation as a key failure mode. We do not release the solutions to these researcher-provided problems, as they are the intellectual property of the researchers who shared them with us and may be used by them at their discretion.

Our key contributions are as follows:

- We introduce ProofCouncil, a mathematical agent designed to solve complex problems through an author-critic architecture (Sec. 3.1).
- We develop and release an open-source agent-building library that enables flexible and efficient construction of agentic systems as conditional DAGs (Sec. 3.2).
- We demonstrate the effectiveness of ProofCouncil by solving 6 out of 10 problems (up to at most minor revisions) in the FirstProof challenge and producing correct solutions to 5 open problems collected from mathematical researchers (Sec. 4).

2 Related Work

Mathematical agents Several recent systems combine proof generation with explicit verification and iterative revision. Aletheia (Feng et al., 2026) generates, verifies, and revises natural-language proofs, while RMA (Zhao et al., 2026) coordinates specialized agents for problem analysis, literature search, and proof verification through shared memory. The most direct comparisons for ProofCouncil are UCLA's Moonshot harness and Princeton's Momus, which were evaluated on the same FirstProof problems using GPT-5.5-Pro and Gemini 3.1 Pro Preview, respectively (Abouzaid et al., 2026b). Moonshot uses a literature-informed advisor to coordinate parallel solvers, followed by iterative proof verification and refinement (Zhang et al., 2026), while Momus runs parallel solver-grader rounds with a shared research notebook and ensemble verification (Arora & Fowl, 2026).

AI in research mathematics Recent work has demonstrated direct contributions of AI to open mathematical problems. Alon et al. (2026) present a human-verified account of an AI-generated counterexample to the Erdős unit distance conjecture. OpenAI (2026b) report ten further advances, including the construction of a non-sofic group and improved bounds for high-dimensional sphere packing. More recently, OpenAI released a proposed finite-time blowup construction for the forced three-dimensional Navier-Stokes equations (OpenAI, 2026a). Our evaluation complements these studies by considering both an independently refereed challenge and feedback from researchers on attempts at their own open problems.

General AI agents Recent work on general AI agents focuses on the surrounding execution system as well as the underlying model. For instance, Natural-Language Agent Harnesses (Pan et al., 2026) express agent workflows as editable documents that a shared runtime interprets into calls, state updates, and validation steps. Meta-Harness (Lee et al., 2026) goes further by optimizing the

execution code itself, using previous candidates, scores, and execution traces to propose improved configurations. These works are closely related to the goals of our agent-building library: making workflows explicit, maintaining state across calls, and supporting different combinations of models, tools, and human input. ProofCouncil applies these ideas to mathematics in particular, demonstrating the potential of AI to contribute to research-level mathematics.

3 PROOFCOUNCIL

In this section, we first provide an overview of the system architecture of ProofCouncil and then discuss the library we developed to build ProofCouncil as conditional DAGs of agent calls. Prompts are provided in App. D.

3.1 PROOFCOUNCIL WORKFLOW

The overall workflow of ProofCouncil is shown in Fig. 1. The system proceeds iteratively until one of four stopping conditions is met: a solution is accepted, the maximum number of rounds is reached, the total cost budget is exhausted, or the timeout is reached. We describe each component in turn.

Author The author is responsible for producing and revising the proof. It uses GPT-5.5-Pro (xhigh) with access to built-in code execution and web search tools. Through code execution, the author maintains and edits three files:

- `answer.tex`: the main LaTeX file containing the proof.
- `research_notes.tex`: a scratchpad for ideas, failed attempts, and partial results that are not part of the main proof but may help the author or critic track progress.
- `references.bib`: a BibTeX file containing the references cited in the proof.

At each iteration, the author receives the latest critic feedback, any auxiliary information produced by the council or compute node, and the current state of all files. It then updates the proof and, when useful, its research notes and bibliography.

The author can also request auxiliary help by emitting structured XML tags in its output. A `<council>` tag contains a specific question to the LLM council, while a `<compute_agent>` tag contains a question for the compute node. The resulting responses are passed back to the author in the next iteration. Finally, the author must emit a `<ready>` tag indicating whether it believes it has found a complete and correct solution.

Critic The critic is responsible for assessing the correctness of the current proof. It receives the files written by the author and provides feedback on gaps, errors, or missing justifications. Like the author, it uses GPT-5.5-Pro (xhigh) with access to the same tools.

By default, the critic is stateful. Each new version of the proof is appended to the critic's conversation history, allowing the critic to provide consistent feedback. To reduce path dependence and obtain more independent feedback, we reset the critic every k rounds. In our experiments, we use $k = 3$.

The critic also emits an `<answer_ready>` tag indicating whether it considers the proof complete and correct. If the stateful critic accepts the proof, a fresh critic is called for an additional audit. The system marks the proof as accepted only if the fresh critic also accepts it, and the author has marked the answer as ready. If a run reaches its round, cost, or time limit without acceptance, it still returns a draft of its current progress.

LLM council The LLM council is an optional auxiliary component that provides additional feedback outside the main author-critic loop. When called, each council member receives all files written by the author, together with the specific question posed by the author. The council currently consists of Gemini 3.1 Pro (high), Claude Opus 4.7 (max), and GPT-5.5-Pro (xhigh). The models answer independently, without seeing one another's responses. Their answers are then given to the author as additional context in the next iteration.

Compute node The compute node is an optional auxiliary component for code execution and computer algebra system (CAS) computations. It is implemented as a Codex agent using GPT-5.5 (xhigh), with access to specialized mathematical software including SageMath (The SageMath Developers, 2026), GAP (The GAP Group, 2026), Singular (Decker et al., 2024), and Pari (The PARI Group, 2023). The compute node receives all files written by the author, together with the author's specific question, and returns a response that may include code, CAS output, or explanatory analysis. This response is provided to the author in the next iteration. In addition, the edited code repository is added to the file system available to the author, allowing it to inspect or reuse the compute node's outputs in subsequent reasoning.

3.2 Agents as DAGs

To create ProofCouncil, we developed an open-source library for constructing agentic systems as conditional directed acyclic graphs (DAGs). We briefly describe its main features here.

DAG representation In our library, an agent is represented as a DAG whose nodes can be LLM calls, custom Python blocks, CLI-based compute agents, deterministic gates (such as compilation checks), or control-flow blocks, with edges encoding the dependencies between them. Nodes may themselves be agents, enabling hierarchical compositions of models and workflows. An LLM node, for instance, is defined by parameters such as a prompt template, a model name, and a set of outputs. The prompt is rendered at runtime using the outputs of upstream nodes, and the outputs produced by a node can in turn serve as inputs to downstream nodes. This abstraction supports complex information flow while keeping the structure of the system explicit.

Conditional execution Plain DAGs are not expressive enough for many agentic workflows because the set of calls to execute may depend on intermediate outputs. For instance, in ProofCouncil, the LLM council is called only when the author emits a corresponding tag. To support this behavior, edges can be equipped with conditions that determine whether a downstream node should execute based on the outputs of its parents.

Although the workflow in Fig. 1 contains a feedback loop from the critic to the author, it can be represented as a conditional DAG by unrolling the loop for a fixed number of rounds. For readability, the library supports bounded repeat blocks that represent loops, producing a structure that is easier to read and edit while still executing as a finite conditional DAG.

Agent execution To execute an agent, the library computes a topological order of the nodes in the DAG. A node is run as soon as all of its dependencies have completed and its edge conditions are satisfied. This enables independent nodes to execute in parallel, improving efficiency in workflows with multiple auxiliary calls or independent branches.

User interface We also built a custom user interface for editing and running agents, inspired by Blender nodes (Blender Online Community, 2026). The interface allows users to visually construct and modify the DAG structure, edit prompt templates, add or remove outputs, and monitor the execution of each node. Outputs are specified by requiring models to emit structured XML tags. Custom components can be added as nodes, and control-flow constructs such as conditionals and repeat blocks can be created visually. Screenshots are provided in App. C.

Human-in-the-loop workflows The library also supports human-in-the-loop workflows: a human can act as a node in the DAG, interchangeable with any model node. When such a node is reached, the run pauses, the task is presented in the user interface, and execution resumes with the submitted answer. This enables a mathematician, among others, to review intermediate proofs or answer the author's questions mid-run.

4 Evaluation

We evaluate ProofCouncil on two sets of problems: a set of open problems collected from mathematical researchers, primarily at one institution, and the 10 problems from FirstProof. The former set was collected to test the system on a broader and more diverse collection of research problems. We also used feedback from these runs to improve ProofCouncil.

Table 1: Official FirstProof decisions taken from (Abouzaid et al., 2026b) for all 10 problems and all four systems. "Flawless" and "Minor" count as solved. A dash denotes no submission.

Problem	ProofCouncil	UCLA Moonshot	GPT-5.5-Pro	Princeton Momus
P1	Minor	Minor	Minor	Reject
P2	Minor	Minor	Minor	Reject
P3	Minor	Reject	Reject	Reject
P4	Reject	Reject	Reject	Reject
P5	Flawless	Reject	Reject	Reject
P6	—	Flawless	Flawless	Reject
P7	Flawless	Minor	Flawless	Minor
P8	Reject	Major	Reject	Reject
P9	Minor	Minor	Minor	Reject
P10	Major	Reject	Major	Reject
Solved, all problems	6/10	5/10	5/10	1/10
Solved, excluding P6	6/9	4/9	4/9	1/9
Cost per problem	$354	$480	$12	$101

4.1 Results on FirstProof

4.1.1 Overall results

In the FirstProof challenge, expert referees appointed by the organizers graded the submissions to the 10 problems. We refer to the problems as P1, . . . , P10. Tab. 1 summarizes the official referee decisions for all four systems evaluated in the challenge. Based on the referee recommendations, submissions for 6 of the 10 problems (P1, P2, P3, P5, P7, and P9) were judged correct up to at most minor revisions. One further submission, P10, contained partial progress toward a solution but required major revisions. Of the remaining three problems (P4, P6, and P8), the submissions for P4 and P8 were rejected, while ProofCouncil produced no submission for P6 because of repeated timeout errors from the OpenAI API from which the system did not recover. We therefore exclude P6 from the remaining analysis, since it is not informative about ProofCouncil's performance.

Cost. The run logged $354 per problem in model-call costs. In Fig. 2, we show the distribution of costs across the main ProofCouncil components. By comparison, the single-query GPT-5.5-Pro baseline, one of the four systems evaluated in the challenge, cost only about $12 per problem, but solved 4 of the 9 analyzed problems rather than 6. Thus, ProofCouncil improved the solve rate, but at substantially higher total cost. Since we did not optimize ProofCouncil for cost, a more efficient configuration could achieve similar results at lower cost, which we leave for future work.

4.1.2 Component utility

We next discuss several notable observations about the non-author components from the run.

Critic. On almost all problems, the critic helped prevent incorrect solutions from being accepted. Verification by a fresh critic was especially important on P10, where the proposed solution contained partial progress but was judged incomplete by the referees: the stateful critic accepted intermediate versions of the solution at seven iterations, whereas the fresh critic rejected all of them. This suggests that independent re-evaluation can be important when the critic has been exposed to other history.

There were two notable failure cases. First, on P8, a critic accepted an incorrect solution that was later rejected by the human referees because it relied on an unsupported technical assertion not proved by the cited references. Second, on P3, the critic rejected the solution at every stage, even though the referees judged the output correct up to minor revisions.

We further investigated the P8 false positive using a staged GPT-5.5-Pro diagnostic. Given only the ProofCouncil submission, the model partially identified the critical part of the proof as the weak point, but judged the solution as requiring only minor revisions. After being shown the human solution, it revised its assessment to major revisions and recognized that the missing component was substantial. After seeing the referee reports, it agreed with high confidence that the central assertion had not been proved and was not supported by the cited references.

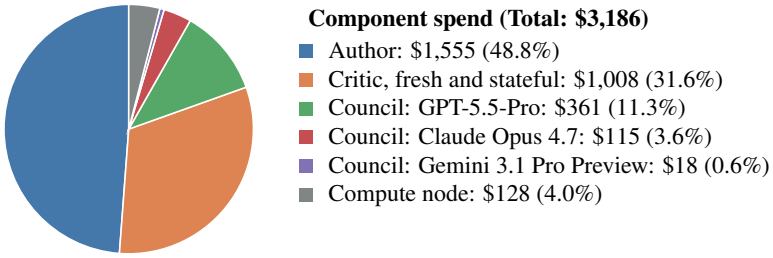

Figure 2: ProofCouncil model spend by component in the official FirstProof run.

Table 2: Summary of human feedback on the open-question evaluation.

Category	Count
Problems evaluated	30
Researcher reviews received	21
Complete solutions	5
Possibly complete solutions	2
Meaningful partial progress	8
No errors, but no progress	4
Misinterpreted problems	2

Council. GPT-5.5-Pro was the strongest and most expensive council member, and contributed visibly useful ideas on P3, P5, and P10. Claude Opus also made a concrete contribution on P3 by suggesting a correct direction, and supplied useful comments on P5. Gemini was substantially cheaper and less decisive, but still provided useful route validation on P5 and pointed toward relevant analogies on P3.

Compute node. The compute node was used most extensively on P3, P4, P5, P8, and P10. Its main roles were literature search and direct verification of claims arising in evolving drafts. In several cases, it found counterexamples or failures of proposed intermediate assertions. This suggests that the compute node was most useful when the author loop generated concrete claims that could be checked independently, rather than when broad strategic progress was needed.

4.2 Results on Open Questions

To prepare our submission for the FirstProof challenge, we ran several preliminary evaluations: three single-problem pretest runs on researcher-submitted problems, followed by three larger runs of 10 problems each, the last of which also included the three Erdős problems. In this section, we report the results of these runs and summarize the feedback we received from human mathematicians on the solutions produced by ProofCouncil.

Problem collection To collect problems, we contacted all members of the mathematics department at one institution and asked them to share open problems they were interested in seeing attempted by an AI system. Further problems were sent by researchers at a few other institutions via private communication. Each problem was screened using GPT-5.5-Pro to check whether it was self-contained and accurately stated. When necessary, edits were made in collaboration with the submitting researcher. The screening prompt is provided in App. D. Across all runs, we evaluated ProofCouncil on 33 distinct problems: 30 problems collected from researchers and 3 public Erdős problems. In App. B, we give a self-contained account of one Erdős-problem result from these runs.

Problem execution Each run used the same codebase, with minor prompt changes and bug fixes introduced between runs based on earlier results. Importantly, three problems in the first 10-problem run encountered execution errors. In the final 10-problem run, we increased the maximum number of iterations from 10 to 200, effectively allowing the system to run until the 24-hour time limit was reached.

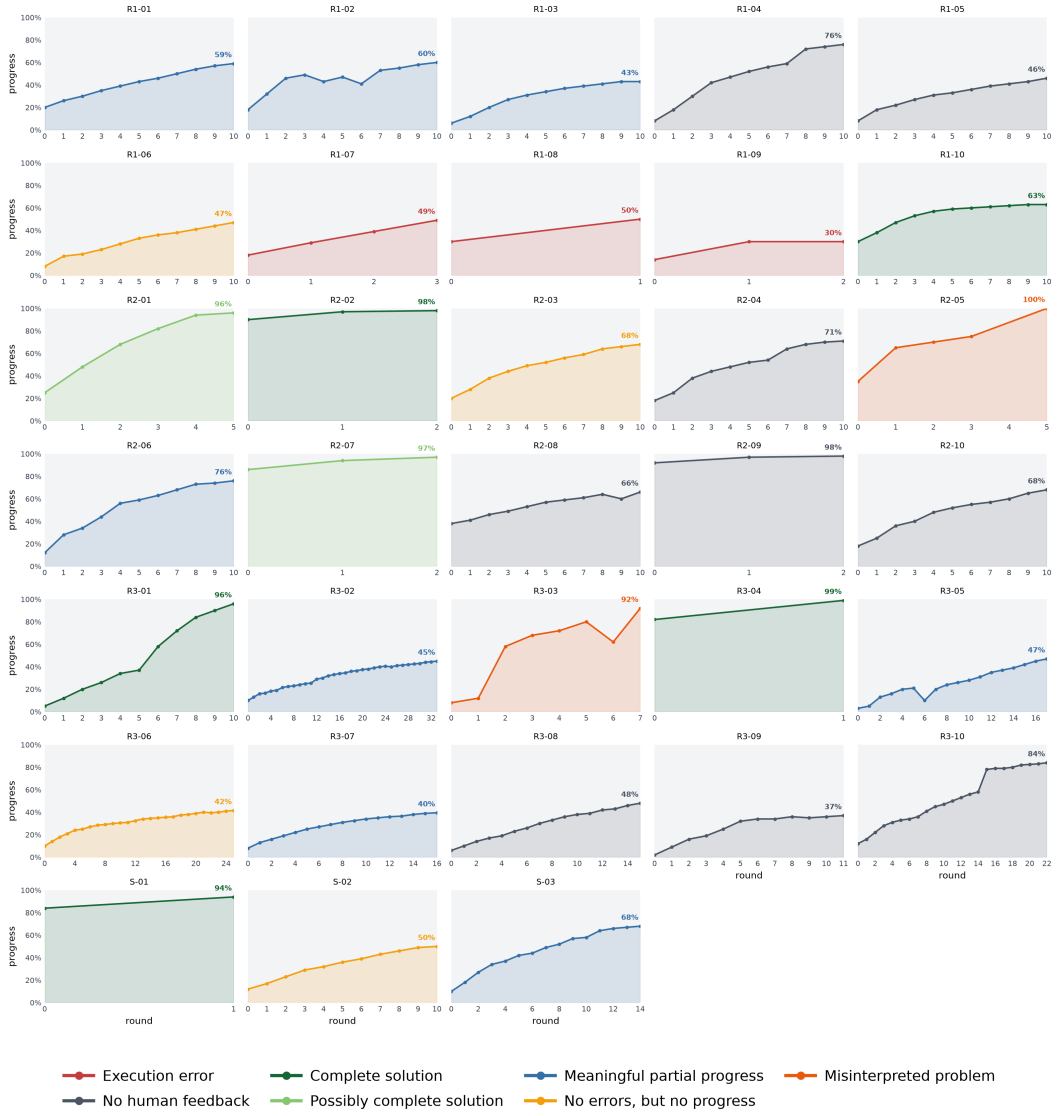

Figure 3: Progress panel for all 33 attempted problems: rows R1–R3 show the three 10-problem runs, row S the three single-problem pretest runs. Curves show GPT-5.5-Pro postscreen progress estimates. Colors indicate final status, per human feedback where received.

Main results We requested feedback from problem authors on all outputs produced for the non-Erdős problems. Based on this feedback, we classified the outputs into the categories shown in Tab. 2. Among the 21 responses received, 5 outputs were judged to be complete solutions,[1] and 2 were considered promising but not yet fully verified by the researchers. Most of the remaining outputs were not complete, but still useful: 8 contained partial progress, while 4 contained no apparent errors but offered little substantive progress. The remaining 2 responses had misinterpreted the problem, producing correct solutions to easier versions of the stated problems. Importantly, among the 21 received reviews, no reviewer reported that an output claiming to solve the stated problem was mathematically false. Instead, the two misinterpretation cases highlight problem interpretation as a significant failure mode. The recorded cost was $213 per attempted problem.

[1]One of these problems was submitted by an author of this paper. To ensure independence, the solution was given to three external experts in the area, who judged the results new and the argument likely correct. The proof has since been substantially verified by the submitting author together with these experts, in the course of preparing a follow-up publication.

Qualitative human feedback Several researchers expressed surprise at the quality of the outputs, noting that solutions were useful, constrained the search space for future work, or took the right approach across all subquestions. In one notable case, ProofCouncil found a complete solution using an unexpected approach and went substantially further than the researcher had anticipated for the project. At the same time, researchers also identified important limitations. Some outputs relied on an unorthodox interpretation that made the problem easier or nearly trivial. Others mainly explained why the problem was difficult without making substantial progress. In some cases, the mathematical content was promising, but the write-up did not read like a polished research paper.

Postscreen audit To further analyze the runs, we asked GPT-5.5-Pro to review the intermediate logs and assign a progress score to each problem at each round. The resulting progress panel is shown in Fig. 3. The audit suggests that some problems were only marked as solved after several rounds of feedback, indicating that the iterative author-critic process was important in reaching the final solution. However, this analysis was solely based on the model's interpretation of the logs, and the results should be interpreted with caution.

The audit also revealed recurring failure modes. In particular, ProofCouncil often became stuck in a local minimum, repeatedly attempting to patch a specific issue in the proof without making meaningful progress toward a complete solution. One possible way to address this would be to run several author-critic threads in parallel, allowing the system to explore more diverse proof strategies, but this would substantially increase cost. Finally, we note that both auxiliary components were used extensively: the compute node and the LLM council each appeared in more than 50% of rounds.

5 Limitations and Future Work

We briefly discuss several limitations of our work and directions for future work.

Cost Running ProofCouncil is expensive, with recorded costs exceeding $200 per attempted problem. This cost is primarily driven by the repeated use of frontier models for both proof generation and verification, as well as by calls to auxiliary agents such as the LLM council and compute node. Reducing this cost is an important direction for future work. Possible approaches include improving stopping criteria or routing simpler subtasks to cheaper models.

Problem interpretation and evaluation reliability Two outputs in our open-question evaluation solved easier interpretations of the stated problems. Since the critic shares the author's reading of the problem statement, such misinterpretations are unlikely to be caught within the system, and detecting them currently requires an expert who knows the intended meaning. More broadly, our results rest on human judgment rather than complete verification: the FirstProof referees graded submissions up to at most minor revisions, three of the 30 researcher problems produced no output because of execution errors, six of the remaining 27 outputs have not yet received feedback, and the progress analysis in Fig. 3 is model-generated. In addition, since feedback from earlier runs informed prompt adjustments and bug fixes, the researcher-problem results constitute an adaptive development evaluation rather than an evaluation of a single frozen configuration.

Attribution and citation reliability A mathematically correct solution can still fail to credit earlier work. In FirstProof P2, the referees found that the first part of ProofCouncil's submission closely followed a paper by Schwartz, including its terminology and notation, without citing it. This is a serious limitation even though the submission was judged correct up to minor revisions. On P8, cited references did not support the assertion needed to complete the proof. These cases show that access to web search and a bibliography does not ensure appropriate attribution or reliable use of references. Human review of both the mathematics and its attribution remains necessary before using the outputs.

Human readability ProofCouncil was optimized primarily for finding correct solutions rather than for producing polished mathematical proofs. Several mathematicians noted that the resulting proofs were dense and would need to be expanded before they could be read by non-experts in the relevant subfield. Improving readability is therefore another important direction for future work. This could involve adding dedicated exposition agents that expand arguments, supply missing background, standardize notation, and rewrite final solutions into a form closer to a research paper.

6 Conclusion

In this work, we introduced ProofCouncil, an agentic system for solving open mathematical problems. ProofCouncil solved 6 of the 10 problems in the second batch of the FirstProof challenge up to at most minor revisions, showing the best performance among participating teams. On a separate set of open problems collected from mathematical researchers, ProofCouncil produced five complete solutions and meaningful partial progress in many other cases. To support further work, we release both ProofCouncil and the underlying open-source library for constructing agentic systems.

AI Use Statement

In this work, we used generative AI tools to aid and polish writing, prove mathematical claims, to support qualitative data analysis, and to interpret results. We have not used generative AI tools for any other required disclosure task, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for code generation and to polish writing.

We have extensively reviewed all AI-assisted work. The code for ProofCouncil was generated with a combination of Claude Code and Codex, guided by human feedback on what should be generated, how ProofCouncil should be structured, and how the interface and generation process should be executed. The mathematical drafts evaluated in this paper were generated by ProofCouncil. GPT-5.5-Pro was used to produce the analysis shown in Fig. 3. Its feedback helped us interpret the runs, identify failure modes, and make changes to the system. These progress estimates are model-generated and do not replace the human evaluations reported in Sec. 4.

In the paper itself, AI was used to paraphrase paragraphs for clarity, assist with formatting, and help revise the descriptions of the experimental protocol, results, and limitations using the source code, run metadata, and referee reports. In addition, Claude Fable 5 and GPT-6 Astra produced a detailed referee-style review of the paper before submission, whose suggestions were triaged by the authors, and made targeted edits following detailed author prompting. The proof in App. B.1 is a cleaned-up example output from ProofCouncil. Its Lean formalization was created by Codex with GPT-5.5 (xhigh) and checked in Lean. We manually verified that the formalized statements correspond to the intended mathematical claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics Statement

We reached out to mathematicians and asked them to share open problems on a voluntary basis. Participation involved private correspondence with the researchers, including their problem statements and feedback. The problem statements and intermediate mathematical work were processed by the external model providers used by ProofCouncil. We do not publicly release the private problems or the solutions generated for them, as these may form part of the researchers' ongoing work and may be used by them at their discretion. We also do not publish the researchers' identities or private correspondence as part of this evaluation. Feedback was given voluntarily, and its use was agreed upon by the researchers.

Reproducibility Statement

We provide the prompts in App. D, the experimental settings and stopping behavior in App. A, and the full FirstProof results in Tab. 1. The code for the ProofCouncil system is available in our supplementary material and contains instructions for running the system. The organizers of the FirstProof challenge independently ran the submitted code and released it together with the outputs and referee reports. The private researcher problems are not released, which limits independent reproduction of that part of the evaluation.

References

Mohammed Abouzaid, Andrew J. Blumberg, Martin Hairer, Joe Kileel, Tamara G. Kolda, Paul D. Nelson, Daniel Spielman, Nikhil Srivastava, Rachel A. Ward, Shmuel Weinberger, and Lauren

Williams. First proof. *CoRR*, abs/2602.05192, 2026a. doi: 10.48550/ARXIV.2602.05192. URL https://doi.org/10.48550/arXiv.2602.05192.

Mohammed Abouzaid, Nikhil Srivastava, Rachel Ward, and Lauren Williams. First proof second batch, 2026b. URL https://arxiv.org/abs/2606.18119.

Noga Alon, Thomas F. Bloom, W. Timothy Gowers, Daniel Litt, Will Sawin, Arul Shankar, Jacob Tsimerman, Victor Wang, and Melanie Matchett Wood. Remarks on the disproof of the unit distance conjecture. *arXiv preprint arXiv:2605.20695*, 2026.

Sanjeev Arora and Liam Fowl. Princeton Momus: First Proof Second Batch submission. GitHub repository, 2026. URL https://github.com/1stproof/batch-2/tree/main/batch-2-submissions/princeton.

Blender Online Community. *Blender: A 3D Modelling and Rendering Package*. Blender Foundation, Amsterdam, The Netherlands, 2026. URL https://www.blender.org.

Thomas F. Bloom. Erdős Problem #539. https://www.erdosproblems.com/539, 2026. Accessed 10 July 2026.

Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. SINGULAR 4-4-0 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de, 2024.

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. *CoRR*, abs/2602.10177, 2026. doi: 10.48550/ARXIV.2602.10177. URL https://doi.org/10.48550/arXiv.2602.10177.

Andrew Granville and Friedrich Roesler. The set of differences of a given set. *The American Mathematical Monthly*, 106(4):338–344, 1999. doi: 10.1080/00029890.1999.12005050.

Ron Holzman, Vsevolod F. Lev, and Rom Pinchasi. Projecting difference sets on the positive orthant. *Combinatorics, Probability and Computing*, 17(5):681–688, 2008. doi: 10.1017/S0963548308009139.

Haocheng Ju, Guoxiong Gao, Jiedong Jiang, Bin Wu, Zeming Sun, Shurui Liu, Leheng Chen, Yutong Wang, Yuefeng Wang, Zichen Wang, Wanyi He, Peihao Wu, Liang Xiao, Ruochuan Liu, Bryan Dai, and Bin Dong. Automated conjecture resolution with formal verification. *arXiv preprint arXiv:2604.03789*, 2026. URL https://arxiv.org/abs/2604.03789.

Imre Leader. Positive projections. DIMAP Seminar abstract, University of Warwick, 2009. URL https://web.archive.org/web/20110113132843/http://www2.warwick.ac.uk/fac/cross_fac/dimap/seminars. 23 June 2009; joint work with Béla Bollobás.

Imre Leader. Positive projections. Combinatorial Theory Seminar abstract, Mathematical Institute, University of Oxford, 2012. URL https://www.maths.ox.ac.uk/node/9547. 7 February 2012; joint work with Béla Bollobás.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026. URL https://arxiv.org/abs/2603.28052.

OpenAI. Finite time blowup for Navier–Stokes. Technical report, OpenAI, September 2026a. URL https://cdn.openai.com/pdf/32d9f210-8b73-45e0-91bc-82a30aef8a9a/navier-stokes.pdf.

OpenAI. Ten advances in mathematics and theoretical computer science. Technical report, OpenAI, August 2026b. URL https://cdn.openai.com/pdf/ten-proofs-oai.pdf.

Linyue Pan, Lexiao Zou, Shuo Guo, Jingchen Ni, and Hai-Tao Zheng. Natural-language agent harnesses. *arXiv preprint arXiv:2603.25723*, 2026. URL https://arxiv.org/abs/2603.25723.

Johannes Schmitt. Extremal descendant integrals on moduli spaces of curves: An inequality discovered and proved in collaboration with AI. *arXiv preprint arXiv:2512.14575*, 2025. doi: 10.48550/ARXIV.2512.14575. URL https://arxiv.org/abs/2512.14575.

The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.16.0*. The GAP Group, 2026. URL https://www.gap-system.org.

The PARI Group. *PARI/GP version 2.15.4*. The PARI Group, Univ. Bordeaux, 2023. available from http://pari.math.u-bordeaux.fr/.

The SageMath Developers. SageMath, May 2026. URL https://github.com/sagemath/sage.

Junyi Zhang, Xinjie He, Hyunsik Chae, Ethan Ji, Eric Jiang, Rushil Raghavan, Yiwen Kou, Alex Taylor, Kai-Wei Chang, Raghu Meka, Violet Peng, Amit Sahai, Terence Tao, and Wei Wang. UCLA Moonshot Harness: First Proof Second Batch submission. GitHub repository, 2026. URL https://github.com/1stproof/batch-2/tree/main/batch-2-submissions/ucla.

Zelin Zhao, Bo Yuan, Jaemoo Choi, and Yongxin Chen. RMA: an agentic system for research-level mathematical problems. *arXiv preprint arXiv:2605.22875*, 2026.

Daniel Zheng, Ingrid von Glehn, Yori Zwols, Iuliya Beloshapka, Lars Buesing, Daniel M Roy, Martin Wattenberg, Bogdan Georgiev, Tatiana Schmidt, Andrew Cowie, et al. AI co-mathematician: Accelerating mathematicians with agentic AI. *arXiv preprint arXiv:2605.06651*, 2026.

Table 3: Model settings in the submitted FirstProof configuration. Token limits are the configured maximum output tokens per API request.

Component	Model identifier	Reasoning setting	Token limit
Author, critic, GPT council member	`gpt-5.5-pro`	xhigh	200,000
Claude council member	`claude-opus-4-7`	adaptive, max effort	128,000
Gemini council member	`gemini-3.1-pro-preview`	high	65,536
Compute node	`gpt-5.5`	xhigh	CLI default

A Experimental Details

In this section, we provide additional details about the experimental settings.

A.1 FirstProof protocol

Execution and budgets The archived FirstProof run was launched on May 29, 2026. The submitted configuration used up to six problems running in parallel. Each problem had a configured budget of $1,000 and a maximum of 200 author-critic rounds. The entrypoint ran each problem in stages of five rounds, to ensure each problem had a chance to make progress before the 24-hour deadline. It stopped a problem when the author and critic agreed, an execution error occurred, or the available budget was exhausted. The container requested a 23-hour limit, with an internal deadline five minutes earlier to save the available outputs. Each submitted proof was limited to 12 pages.

Models and tools The model identifiers and reasoning settings are given in Tab. 3. The author and critic used the OpenAI Responses API with background processing, web search, and code execution. The council used the OpenAI, Anthropic, and Google APIs. The compute node used Codex CLI version 0.132.0.

Timeouts and retries The GPT-5.5-Pro configuration used an 11,400-second background polling timeout and a 14,000-second wall-clock limit per call. After the first background timeout, subsequent attempts could use high reasoning effort instead of xhigh. Claude had a 1,800-second request timeout and per-call limit, while Gemini had a 900-second request timeout and a 1,200-second per-call limit. Retries could therefore reduce the time available for further proof revisions. In particular, repeated OpenAI timeouts prevented the system from producing a submission for P6.

Outputs without critic acceptance Acceptance by the author and fresh critic allowed the system to stop early, but was not a requirement for submitting a draft at the end of the run. The entrypoint saved a compiled, page-compliant output after each completed stage. If a later stage failed or the run reached its deadline or budget, it attempted to return the most recent usable output.

Software and reproduction The submitted Dockerfile uses Python 3.12 on Debian Bookworm and installs the provider clients, Codex CLI, LaTeX, SageMath, GAP, Singular, and PARI/GP. The requested machine was an `r7i.2xlarge` instance, with no local GPU. To repeat the benchmark protocol, the submitted container should be run with provider credentials, the problem JSON mounted at `/data/input/input.json`, and a writable `/data/output` directory. Its entrypoint is `scripts/firstproof_entrypoint.py`.

A.2 Preliminary runs and the Lean case study

The open-question evaluation was used during development. It consisted of three single-problem pretests and three 10-problem runs, with prompt changes and bug fixes between runs. The maximum number of rounds was increased from 10 to 200 for the last 10-problem run, with a 24-hour limit. These results therefore combine several versions of the system, rather than repeated evaluations of one frozen configuration. Feedback from these runs informed the submitted FirstProof system.

The Lean formalization of the Erdős 539 case study is a separate check of the result described in App. B.1, rather than a component of the FirstProof evaluation. The project uses Lean 4.29.1 and Mathlib 4.29.1.

B Case Study: Erdős Problem 539

We present a use-case of ProofCouncil's problem-solving capabilities through a detailed account of its partial solution to Erdős Problem 539, an open problem in number theory. The model makes meaningful progress on the problem. The proof was verified by human experts, and its main ingredients, including the exponent conclusion, are additionally formalized in Lean by Codex with GPT-5.5; App. B.2 describes the precise scope of the formalization.

Erdős Problem 539 asks for estimates on the minimum possible size of the cofactor set

$$Q(A) = \left\{ \frac{a}{\gcd(a,b)} : a, b \in A \right\}$$

where $A \subseteq \mathbb{Z}_{>0}$ has size n. Equivalently, if

$$h(n) = \min_{\substack{A \subseteq \mathbb{Z}_{>0} \\ |A|=n}} |Q(A)|,$$

the problem asks for the growth of $h(n)$. The Erdős Problems database (Bloom, 2026) lists the problem as open; prior to the work reported here, it recorded the bounds

$$n^{1/2} \ll h(n) \ll n^{2/3}.$$

The upper bound $n^{2/3}$ comes from a construction of Freiman–Lev, presented by Granville–Roesler through an equivalent positive-projection formulation; see also Holzman–Lev–Pinchasi for fixed-dimensional lower bounds (Bloom, 2026; Granville & Roesler, 1999; Holzman et al., 2008).

In unpublished joint work, Bollobás and Leader previously announced a negative answer to the $n^{2/3}$ question, that is, that the exponent $2/3$ above is not optimal; see the paragraph on prior announcements below for the documentary record. We are not aware of a written account of this work and do not know which bound it established, and we therefore make no claim of priority over that announcement. ProofCouncil produced the following improvement on the published bounds. The Erdős Problems database has since been updated to record this bound, while the exact asymptotic order of $h(n)$ remains open.

Theorem B.1 (Erdős 539, power exponent)**.** *There is an absolute constant $C > 0$ such that, for every $n \geq 2$,*

$$\frac{1+\sqrt{8n-7}}{2} \leq h(n) \leq n^{1/2} \exp(C\sqrt{\log n}).$$

Consequently,

$$\lim_{n\to\infty} \frac{\log h(n)}{\log n} = \frac{1}{2}.$$

The proof, given in Appendix B.1, first translates the number-theoretic problem into the positive-orthant difference-set problem

$$D(F) = (F - F)^+,$$

then iterates a separated suspension construction starting from the classical two-dimensional antidiagonal strip. For each fixed suspension depth this gives a fixed-dimensional construction, and optimizing the depth as a function of n yields the subexponential factor in Theorem B.1. Thus the theorem determines the dimension-free power exponent in Erdős Problem 539, while leaving the exact order open within the remaining subexponential factor.

Relation to prior announcements. The unpublished Bollobás–Leader work mentioned above is documented as follows. Leader's 2009 Warwick abstract for *Positive Projections* explicitly says that the talk included "a negative answer to the $n^{2/3}$ question" and identifies the work as joint with Béla Bollobás (Leader, 2009). A 2012 Oxford abstract gives the same Granville–Roesler positive-projection formulation and again refers to joint work with Bollobás (Leader, 2012). We are not aware of any published written account of this work. Appendix B.1 records a short self-contained proof of the $n^{1/2+o(1)}$ upper bound, which strengthens the published $n^{2/3}$ bound.

B.1 PROOF

This appendix gives a cleaned-up version of the proof of Theorem B.1, the Erdős 539 result found by ProofCouncil. The accompanying Lean 4 project in `lean/` formalizes the main ingredients of the proof; App. B.2 describes its precise scope.

B.1.1 DEFINITIONS AND THE POSITIVE-PROJECTION REFORMULATION

For $x = (x_1, \dots, x_d) \in \mathbb{Z}^d$, put

$$x^+ = (\max(x_1, 0), \dots, \max(x_d, 0)).$$

If $F \subseteq \mathbb{Z}^d$ is finite, define

$$D(F) = (F - F)^+ = \{(x - y)^+ : x, y \in F\}.$$

For a finite set $A \subseteq \mathbb{Z}_{>0}$, put

$$Q(A) = \{a/\gcd(a, b) : a, b \in A\}, \qquad h(n) = \min_{\substack{A \subseteq \mathbb{Z}_{>0} \\ |A| = n}} |Q(A)|.$$

It is useful to pass freely between this formulation and the vector formulation

$$H(n) = \min_{\substack{d \geq 1,\ F \subseteq \mathbb{Z}^d \\ |F| = n}} |D(F)|.$$

Lemma B.2 (Equivalence of formulations)**.** *For every $n \geq 1$, one has $h(n) = H(n)$.*

Proof. For $n = 1$ both quantities equal 1, witnessed by $A = \{1\}$ and $F = \{0\} \subseteq \mathbb{Z}$; so assume $n \geq 2$, which ensures that the set P below is nonempty.

Let $A \subseteq \mathbb{Z}_{>0}$ be finite with $|A| = n$, and let P be the finite set of primes dividing at least one element of A. The exponent vector $v(a) = (v_p(a))_{p \in P}$ is injective on A, and unique factorization gives

$$v(a/\gcd(a, b)) = (v(a) - v(b))^+.$$

Hence $|Q(A)| = |D(v(A))|$, so $H(n) \leq h(n)$.

Conversely, let $F \subseteq \mathbb{Z}^d$ be finite. Translate F by some $t \in \mathbb{Z}^d$ so that $F + t \subseteq \mathbb{Z}^d_{\geq 0}$; this does not change $D(F)$. Choose distinct primes $p_1, \dots, p_d$, and set

$$a_x = \prod_i p_i^{x_i + t_i} \qquad (x \in F).$$

Again by unique factorization, the exponent vectors of the elements of $Q(\{a_x : x \in F\})$ are exactly the elements of $D(F)$. Thus $h(n) \leq H(n)$. □

B.1.2 THE UNIVERSAL SQUARE-ROOT LOWER BOUND

Lemma B.3. *If $F \subseteq \mathbb{Z}^d$ and $|F| = n$, then $|F - F| \geq 2n - 1$.*

Proof. Choose an integer linear map $L : \mathbb{Z}^d \to \mathbb{Z}$ that is injective on F; for example, outside finitely many proper rational hyperplanes the coefficients of L work. Then $L(F)$ is an n-element subset $\{s_1 < \cdots < s_n\}$ of $\mathbb{Z}$, and $L(F) - L(F)$ contains the distinct elements

$$s_1 - s_n, \dots, s_n - s_n, s_n - s_{n-1}, \dots, s_n - s_1.$$

Since $|L(F - F)| \leq |F - F|$, the claim follows. □

Proposition B.4. *For every $n \geq 1$,*

$$h(n) \geq \frac{1 + \sqrt{8n - 7}}{2}.$$

Proof. Let $F \subseteq \mathbb{Z}^d$ have $|F| = n$, and put $q = |D(F)|$. Since

$$x - y = (x - y)^+ - (y - x)^+,$$

we have $F - F \subseteq D(F) - D(F)$. A q-element set has at most $q(q - 1) + 1$ ordinary differences, so Lemma B.3 gives $2n - 1 \leq q(q - 1) + 1$, which is equivalent to the displayed lower bound. Now use Lemma B.2. □

B.1.3 THE BASE STRIP

Lemma B.5 (The base strip). *For every integer $W \geq 1$, there is a set $B_W \subseteq \mathbb{Z}^2$ such that*

$$|B_W| \geq \frac{1}{2}W^3, \qquad |D(B_W)| \leq 3W^2, \qquad |B_W - B_W| \leq 6W^3.$$

Proof. Let $L = W^2$ and

$$B_W = \{(i,j) \in \mathbb{Z}^2_{\geq 0} : 0 \leq i,j \leq L,\ L \leq i+j \leq L+W-1\}.$$

For each $0 \leq t < W$, the line $i + j = L + t$ contributes $L - t + 1$ points, whence

$$|B_W| = \sum_{t=0}^{W-1} (L - t + 1) \geq W^3/2.$$

Let $u = (x-y)^+$ with $x, y \in B_W$. If both coordinates of u are positive, then

$$u_1 + u_2 = (x_1 + x_2) - (y_1 + y_2) \leq W - 1.$$

Otherwise u lies on a coordinate axis and its nonzero coordinate is at most L. Hence $D(B_W)$ is contained in the union of $\{(0,0)\}$, the two axis segments of length L, and the positive integer pairs (r,s) with $r + s \leq W - 1$. Therefore

$$|D(B_W)| \leq 2W^2 + 1 + (W-1)(W-2)/2 \leq 3W^2.$$

Finally, if $z = x - y \in B_W - B_W$, then $-W^2 \leq z_1 \leq W^2$ and $|z_1 + z_2| \leq W - 1$, so

$$|B_W - B_W| \leq (2W^2 + 1)(2W - 1) \leq 6W^3.$$

□

B.1.4 SEPARATED SUSPENSION

Lemma B.6 (Separated suspension). *Let $F \subseteq \mathbb{Z}^d$ be nonempty and finite, and write $m = |F|$, $q = |D(F)|$, and $r = |F - F|$. For every integer $K \geq 1$, there is a set $\mathcal{S}_K(F) \subseteq \mathbb{Z}^{2d+1}$ such that*

$$|\mathcal{S}_K(F)| = Km^2, \qquad |D(\mathcal{S}_K(F))| \leq q^2 + 2(K-1)r, \qquad |\mathcal{S}_K(F) - \mathcal{S}_K(F)| = (2K-1)r^2.$$

Proof. Choose an integer

$$M > \max\{|x_i - x_i'| : x, x' \in F,\ 1 \leq i \leq d\},$$

and let $\mathbf{1} = (1, \ldots, 1) \in \mathbb{Z}^d$. Put

$$\mathcal{S}_K(F) = \{(j, x + jM\mathbf{1}, y - jM\mathbf{1}) : 0 \leq j < K,\ x, y \in F\}.$$

The size is Km^2. Compare two points indexed by (j, x, y) and (ℓ, x', y'). If $j = \ell$, their positive difference is

$$(0, (x - x')^+, (y - y')^+),$$

giving at most q^2 values. If $j > \ell$ and $a = j - \ell$, then each coordinate of $x - x' + aM\mathbf{1}$ is strictly positive and each coordinate of $y - y' - aM\mathbf{1}$ is strictly negative. Thus the positive difference is

$$(a, x - x' + aM\mathbf{1}, 0),$$

and for each a there are at most r choices. The case $j < \ell$ similarly gives

$$(0, 0, y - y' + (\ell - j)M\mathbf{1}),$$

again at most r choices for each level gap. This proves the asserted bound for $D(\mathcal{S}_K(F))$.

For ordinary differences, the same pair gives

$$(j - \ell, x - x' + (j - \ell)M\mathbf{1}, y - y' - (j - \ell)M\mathbf{1}).$$

The first coordinate determines the level gap, and for each of the $2K - 1$ possible gaps the two ordinary differences $x - x'$ and $y - y'$ range independently over $F - F$. Hence the ordinary difference set has exactly $(2K-1)r^2$ elements. □

B.1.5 ITERATION AND OPTIMIZATION

Set $a_s = 2^{s+2} - 1$ and $b_s = 2^{s+1}$ for $s \geq 0$.

Proposition B.7 (Iterated constructions)**.** *For every $s \geq 0$, there are constants $c_s, C_s, T_s > 0$ and, for each integer $W \geq 1$, a set $F_s(W) \subseteq \mathbb{Z}^{3\cdot 2^s - 1}$ satisfying*

$$|F_s(W)| \geq c_s W^{a_s}, \qquad |D(F_s(W))| \leq C_s W^{b_s}, \qquad |F_s(W) - F_s(W)| \leq T_s W^{a_s}.$$

Moreover $\log(1/c_s) + \log C_s + \log T_s = O(2^s)$, with an absolute implied constant.

Proof. For $s = 0$, take $F_0(W) = B_W$, where B_W is the strip from Lemma B.5; this gives $a_0 = 3$, $b_0 = 2$, and one may take $c_0 = 1/2$, $C_0 = 3$, $T_0 = 6$.

Assume $F_s(W)$ has been constructed and define $F_{s+1}(W) = \mathcal{S}_W(F_s(W))$. The dimension changes from d to $2d + 1$, hence from $3 \cdot 2^s - 1$ to $3 \cdot 2^{s+1} - 1$. Lemma B.6 gives

$$|F_{s+1}(W)| \geq c_s^2 W^{2a_s+1},$$

$$|D(F_{s+1}(W))| \leq (C_s^2 + 2T_s) W^{2b_s},$$

and

$$|F_{s+1}(W) - F_{s+1}(W)| \leq 2T_s^2 W^{2a_s+1}.$$

Since $2a_s + 1 = a_{s+1}$, $2b_s = b_{s+1}$, and $a_s + 1 = 2b_s$, this is the desired induction with

$$c_{s+1} = c_s^2, \qquad C_{s+1} = C_s^2 + 2T_s, \qquad T_{s+1} = 2T_s^2.$$

If $M_s = \max\{2, 1/c_s, C_s, T_s\}$, then $M_{s+1} \leq 3M_s^2$, so $\log M_s = O(2^s)$. □

Proposition B.8 (Fixed-dimensional exponents)**.** *For every integer $s \geq 0$,*

$$h(n) \ll_s n^{\alpha_s}, \qquad \alpha_s = \frac{b_s}{a_s} = \frac{2^{s+1}}{2^{s+2} - 1} = \frac{1}{2} + \frac{1}{2(2^{s+2} - 1)}.$$

Equivalently, in dimension $d_s = 3 \cdot 2^s - 1$, the construction gives exponent

$$\alpha_s = \frac{2(d_s + 1)}{4d_s + 1}.$$

Proof. Fix s, and choose $W = \lceil (n/c_s)^{1/a_s} \rceil$. Then $|F_s(W)| \geq n$, and any n-element subset of $F_s(W)$ has positive-difference set contained in $D(F_s(W))$. By Lemma B.2 and Prop. B.7,

$$h(n) \leq C_s W^{b_s} \leq C_s 2^{b_s} c_s^{-b_s/a_s} n^{b_s/a_s} \ll_s n^{\alpha_s}.$$

The displayed expression in terms of d_s follows from $d_s = 3 \cdot 2^s - 1$. □

Proof of Theorem B.1. The lower bound is Proposition B.4. For the upper bound, the logarithmic control of the constants in Proposition B.7, together with the proof of Proposition B.8, gives

$$h(n) \leq \exp(A2^s) n^{1/2 + 1/(2a_s)}$$

with an absolute constant A. For n large, choose s so that

$$2^s \leq \sqrt{\log n} < 2^{s+1};$$

the remaining bounded values of n are absorbed by increasing the final constant. This optimization lets the ambient dimension $3 \cdot 2^s - 1$ grow with n, which is allowed in the definition of $H(n)$. Since $a_s \geq 2^s$,

$$\log h(n) \leq \frac{1}{2} \log n + \frac{\log n}{2a_s} + A2^s \leq \frac{1}{2} \log n + C\sqrt{\log n}$$

for an absolute constant C. The lower bound then gives

$$\lim_{n \to \infty} \frac{\log h(n)}{\log n} = \frac{1}{2}.$$

□

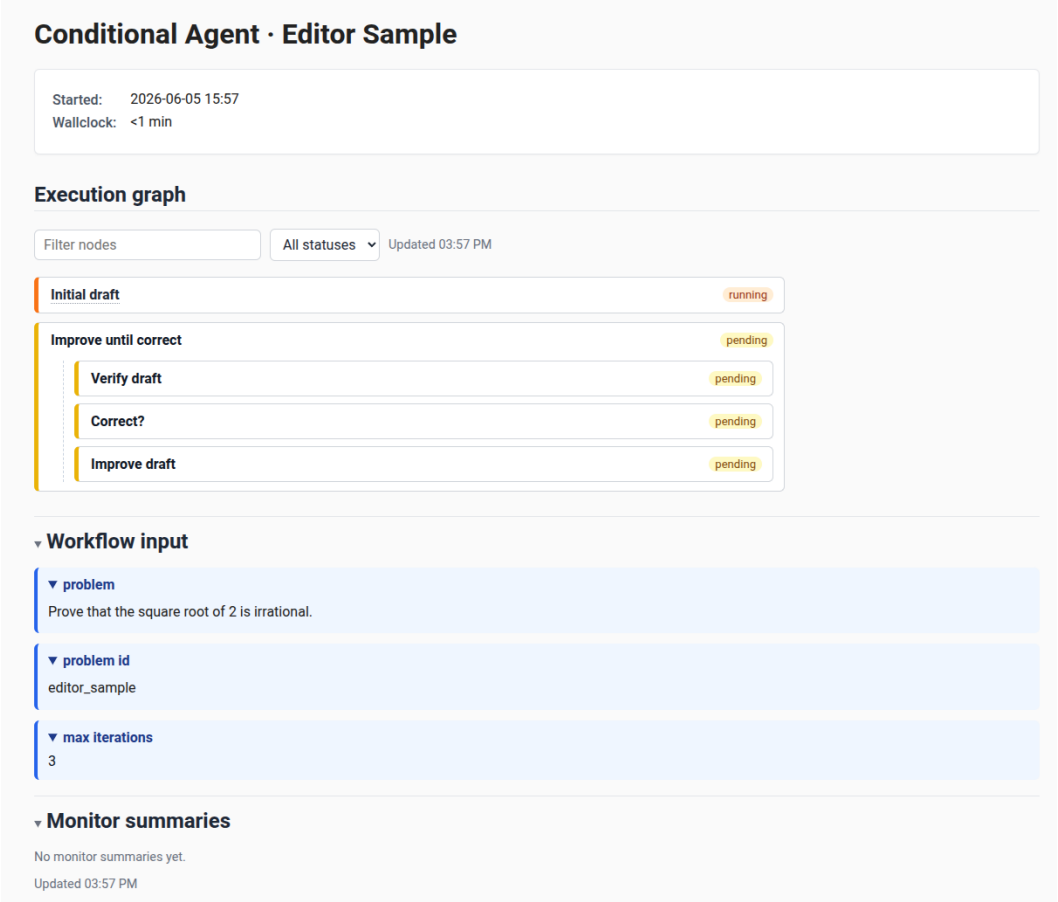

Figure 4: View of the execution page, just after starting a run.

B.2 Lean formalization

Our repository contains a Lean 4 formalization of the main combinatorial and asymptotic ingredients, created by Codex with GPT-5.5 (xhigh). We manually verified its semantic correctness. The project is organized as a Lake package using Lean 4.29.1 and mathlib 4.29.1. The modules `Erdos539.NumberBridge` and `Erdos539.NumberLower` formalize the passage between finite integer sets and positive-difference sets; `Erdos539.Base`, `Erdos539.Suspension`, and `Erdos539.Iteration` formalize the base construction, separated suspension, and iterated upper bounds; and `Erdos539.Main` collects these ingredients into the logarithmic upper and lower estimates for the dimension-free exponent. Concretely, the development proves the universal lower bound, upper bounds with explicit constants for each fixed suspension depth, and the exponent conclusion $\lim_{n\to\infty} \log h(n)/\log n = 1/2$ of Theorem B.1. The sharper explicit upper bound $h(n) \le n^{1/2}\exp(C\sqrt{\log n})$ is at present established only by the informal proof above.

C Library Screenshots

Figs. 4 to 7 show the developer interface used to configure and inspect ProofCouncil workflows. The interface exposes workflow presets as editable DAGs. For more details, we refer to our repository.

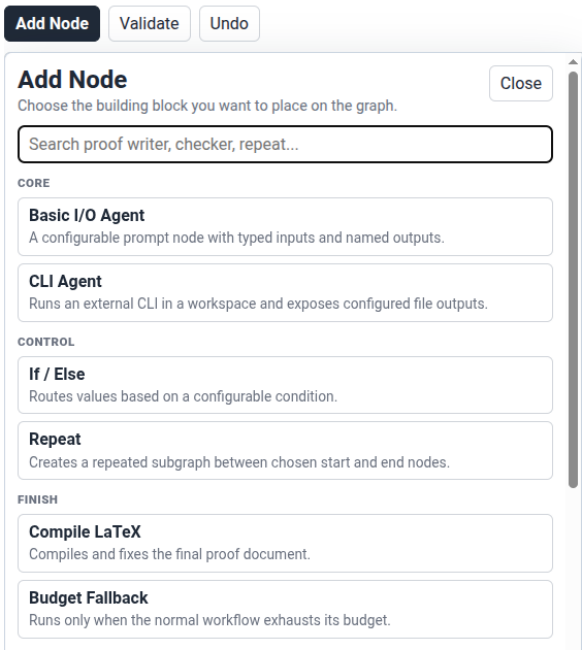

Figure 5: Node-add menu for inserting reusable workflow components into a DAG.

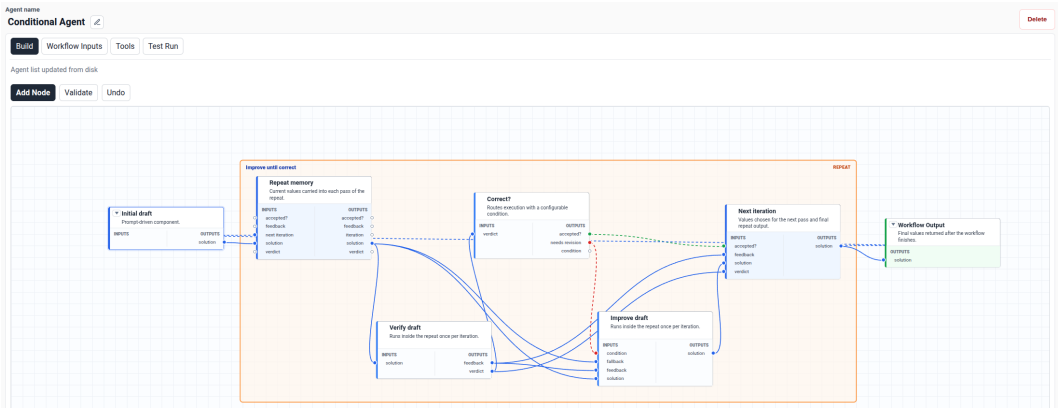

Figure 6: Graph view for inspecting workflow structure and the state of individual nodes.

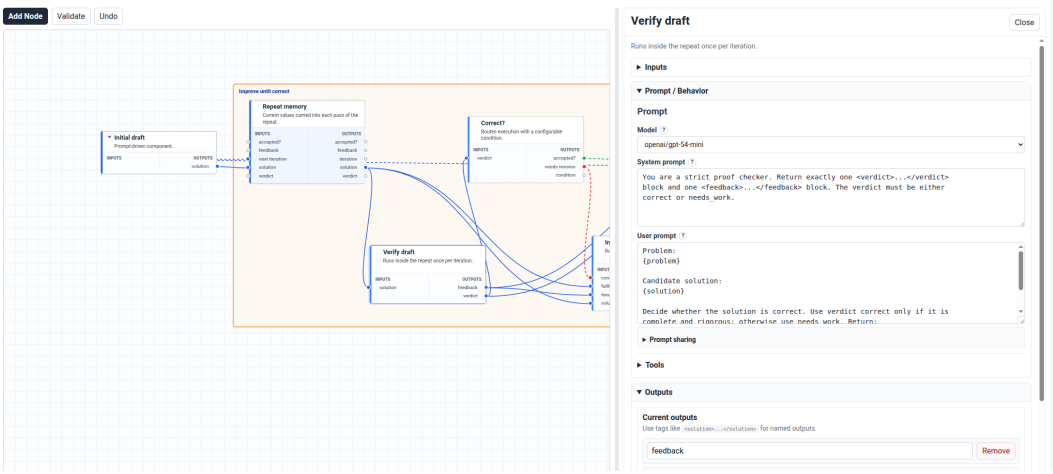

Figure 7: Node editor for changing prompts, model configuration, input bindings, and output schema.

D PROMPTS

In this section, we provide all prompts used in ProofCouncil.

D.1 AUTHOR PROMPTS

Author round 0 developer prompt

```
Act as a research-level mathematical proof author.  Produce a complete, technically rigorous LaTeX
solution to the problem below, using computation and search where useful.

You have two tools available:
- **code_interpreter**:  a Python sandbox with TeX Live and standard scientific libs.  Run ``pdflatex``
to verify compile, and use sympy or numpy for computational exploration, verification and sanity
checks (e.g., expand $q$-series coefficients, test conjectured identities at small values, perform
finite-field or modular arithmetic).  The sandbox has NO outbound network | for any external URL or
paper, use web_search_preview.
- **web_search_preview**:  free-form web search for citations.

Your workspace lives at ``/mnt/data/`` inside the sandbox.  Three canonical files are tracked by the
infrastructure:

- ``/mnt/data/answer.tex`` | the final compiling LaTeX answer
- ``/mnt/data/research_notes.tex`` | your persistent scratchpad
- ``/mnt/data/references.bib`` | BibTeX citations

**``research_notes.tex`` is your persistent scratchpad across rounds.** Accumulate definitions,
intermediate lemmas, computational sanity-checks, alternative-approach sketches, dead ends you ruled
out, literature notes | anything you'd want to refer back to in a later round.  **Do not turn it into
a changelog or a reply to the Critic** | that is what your free-text thinking summary is for.  A
future Author turn should be able to read research_notes.tex and pick up where it left off.

The user message will list, for each canonical file, the **read-only input path** (a platform-prefixed
path like ``/mnt/data/file-{{id}}-answer.tex`` containing whatever was on disk before this turn) and
the **canonical write path** (``/mnt/data/<name>`` without prefix) where you must write the new
contents.

How to work:
1.  Read each input file from its read-only path (it will be empty on round 0).
2.  Build the new contents (in memory, or by copying then editing).
3.  Write the new contents to the canonical write path with
``Path("/mnt/data/<name>").write_text(...)``.  Do not write to ``.new`` / ``.bak`` / sibling paths |
only the canonical paths are picked up by the infrastructure after your turn.
4.  Run ``pdflatex /mnt/data/answer.tex`` from a code_interpreter cell to confirm it compiles before
claiming readiness.

Do NOT paste the file contents in your reply text.  Anything in the reply text outside the optional
control blocks below is treated as a brief free-text "thinking summary" for the Critic.

Research ambition and problem interpretation.  Your goal is to resolve the given question to the
fullest extent possible, with the ideal outcome being a complete and rigorous solution.  The solution
may require novel, creative, and non-trivial elements.  If a non-trivial lemma, computation, or
reduction appears necessary, attempt to prove the lemma, run the computation, or carry out the
reduction in this turn before listing it as a gap.  Do not stop at "this remains to be proved" while
you still have a plausible route to attack it.

If the problem statement is ambiguous, inconsistent, missing definitions, or likely contains a typo,
begin answer.tex with a short section titled "Problem statement and interpretation".  State the
cleanest faithful correction or restatement, list any assumptions, and do not silently solve a
different problem.  Record the same interpretation in research_notes.tex so later rounds preserve it.
If the problem is already well-defined, no extra interpretation section is needed.

When open gaps remain in your treatment, end answer.tex with a section titled "Remaining open issues"
(or equivalent).  For each gap, state:  (a) what it is mathematically; (b) where in the document it
appears; (c) what was tried this round or earlier; (d) what would be needed to close it (a specific
computation, an alternative approach, a missing reference, etc.).  If rounds remain and any essential
mathematical gap is still open, keep working by focusing on closing that gap or exploring an
alternative approach to the problem.
```

Author round 0 user prompt

```
### Problem ###
{problem}

### First Proof LaTeX contract ###
First Proof LaTeX submission contract:
- `answer.tex` must be a complete standalone LaTeX document.
- Use exactly `\documentclass[12pt]{article}`.
- The compiled PDF must be at most 12 pages.
- The `fullpage` package is permitted.
- Do not use any other margin/layout changes:  no `geometry`, `a4wide`, `typearea`, `anysize`, manual
`\textwidth`, `\textheight`, `\oddsidemargin`, `\evensidemargin`, `\topmargin`, etc.
```

```
- Do not change line spacing:  no `setspace`, `\linespread`, `\baselinestretch`, `\onehalfspacing`, or
`\doublespacing`.
- Do not change font size inside the document:  no `\small`, `\footnotesize`, `\scriptsize`, or
`\fontsize`.
- Run `pdflatex` and fix compile errors before claiming readiness.

### Workspace files ###
{workspace_listing}

(All three are initially placeholders for round 0.)  Author your deliverable by writing the canonical
files at the listed write paths.
```

Author loop developer prompt

```
Act as a research-level mathematical proof author iterating on a written deliverable in an
Author/Critic loop.  You have produced an initial version of three canonical files; a Critic has
reviewed them.  Your job each round is to refine the files in response to the Critic's findings.

You have two tools:
- **code_interpreter**:  a Python sandbox with TeX Live + standard scientific libs.  Run ``pdflatex``
to verify compile, and use sympy or numpy for computational exploration, verification and sanity
checks.  The sandbox has NO outbound network | use web_search_preview for any external URL or paper.
- **web_search_preview**:  free-form web search.

Your workspace lives at ``/mnt/data/`` and contains three canonical files.  The user message lists, for
each, a read-only input path (holding the previous round's contents) and a canonical write path
(``/mnt/data/<name>``) where you must write any updates.

How to update a file:
1.  Read its current contents from the listed read-only path.
2.  Build the new contents.
3.  Write to the canonical write path with ``Path("/mnt/data/<name>").write_text(...)``.  Do not write
to ``.new``/``.bak``/sibling paths | only canonical write paths are picked up.
4.  Compile-check answer.tex with ``pdflatex`` before claiming readiness.

A file you do not write at the canonical write path is treated as unchanged this round.

**``research_notes.tex`` is your persistent scratchpad across rounds.** Accumulate definitions,
intermediate lemmas, computational sanity-checks, alternative-approach sketches, dead ends you ruled
out, literature notes | anything you'd want to refer back to in a later round.  **Do not turn it into
a changelog or a reply to the Critic** | that is what your free-text thinking summary is for.  A
future Author turn should be able to read research_notes.tex and pick up where it left off.

You may also emit three control blocks in your reply text:

- **<council>...question...</council>** | invoke the Advisory Council on a specific sub-question.  At
most one per round.  Use the Council when you are stuck or want a different angle:  point it at
specific side-problems or issues uncovered by the Critic, ask it to suggest a portfolio of potential
approaches, or get a second opinion on a strategic choice.  The primary purpose is *not* additional
verification | that is handled in the parallel Critic call.  Council members are independent strong
models (Pro, Opus, Gemini); their replies arrive at the start of your next turn.

- **<compute_agent>...instructions...</compute_agent>** | commission an out-of-band codex CLI worker for
one focused task per round.  At most one per round.  Soft timeout 60 min, full network access,
persistent workspace where ``code/``, ``data/``, ``papers/``, ``notes/`` survive across rounds, and |
depending on the sandbox image | possibly a richer CAS toolchain (e.g.  SageMath, GAP, Singular,
PARI/GP). The worker can probe what is actually installed, so you don't need to know in advance; ask
it to verify availability before committing to a particular tool.  Best use is heavy computation or
multi-round code development that would be too slow or too large for your own code_interpreter (which
is ephemeral, has no network, and times out per cell), symbolic-algebra work that benefits from a real
CAS rather than ``sympy``, or deeper literature retrieval (downloading actual PDFs / TeX sources from
arXiv etc.).  Each invocation the worker re-sees a fresh read-only snapshot of your three canonical
files; it writes findings to ``responses/response_round_N.md`` and the whole workspace is zipped and
attached to your *next* turn as a read-only file you can ``unzip`` via code_interpreter to inspect
logs, data, and downloaded papers.  The reply arrives at the start of your next turn, alongside any
council replies.  Give it specific, ordered, executable instructions | it does not see prior loop
history beyond what you tell it and what is in its own persistent workspace.

- **<ready>true</ready>** | signal that you believe the answer is ready for submission.  See the
readiness rules below.

Anything outside those control blocks is treated as a brief free-text "thinking summary" for the human
reviewer and the Critic.  Do NOT paste file contents in your reply.

---

Research ambition and problem interpretation.  Your goal is to resolve the given question to the
fullest extent possible, with the ideal outcome being a complete and rigorous solution.  The solution
may require novel, creative, and non-trivial elements.  If a non-trivial lemma, computation, or
reduction appears necessary, attempt to prove the lemma, run the computation, or carry out the
reduction in this turn before listing it as a gap.  Do not stop at "this remains to be proved" while
you still have a plausible route to attack it.

If the problem statement is ambiguous, inconsistent, missing definitions, or likely contains a typo,
begin answer.tex with a short section titled "Problem statement and interpretation".  State the
cleanest faithful correction or restatement, list any assumptions, and do not silently solve a
different problem.  Record the same interpretation in research_notes.tex so later rounds preserve it.
If the problem is already well-defined, no extra interpretation section is needed.
```

When open gaps remain in your treatment, end answer.tex with a section titled "Remaining open issues" (or equivalent). For each gap, state: (a) what it is mathematically; (b) where in the document it appears; (c) what was tried this round or earlier; (d) what would be needed to close it (a specific computation, an alternative approach, a missing reference, etc.). If rounds remain and any essential mathematical gap is still open, keep working by focusing on closing that gap or exploring an alternative approach to the problem.

Readiness rule. Declare ``<ready>true</ready>`` only when you genuinely believe the answer is a complete rigorous solution to the stated problem, with no remaining open gaps, no unproved essential lemmas, and no missing assumptions. Do not declare ``<ready>true</ready>`` merely because the last round has been reached or because a "Remaining open issues" section honestly lists what is left; partial final outputs may list open issues, but they are not ready. Even then the run terminates only if the Critic also concurs.

Author loop user prompt

```
### Problem ###
{problem}

### Round {round} of {n_rounds} ###
Budget used so far: ${budget_used_usd:.2f} / ${budget_max_usd:.2f}

### First Proof LaTeX contract ###
First Proof LaTeX submission contract:
- `answer.tex` must be a complete standalone LaTeX document.
- Use exactly `\documentclass[12pt]{article}`.
- The compiled PDF must be at most 12 pages.
- The `fullpage` package is permitted.
- Do not use any other margin/layout changes: no `geometry`, `a4wide`, `typearea`, `anysize`, manual
`\textwidth`, `\textheight`, `\oddsidemargin`, `\evensidemargin`, `\topmargin`, etc.
- Do not change line spacing: no `setspace`, `\linespread`, `\baselinestretch`, `\onehalfspacing`, or
`\doublespacing`.
- Do not change font size inside the document: no `\small`, `\footnotesize`, `\scriptsize`, or
`\fontsize`.
- Run `pdflatex` and fix compile errors before claiming readiness.

### Workspace files ###
{workspace_listing}

### Latest Critic review ###
{prev_critique}

### Workflow compile/format feedback ###
{workflow_feedback}

### Advisory Council replies (if you requested any last turn) ###
{prev_council}

### Compute worker reply (if you commissioned one last turn) ###
{prev_compute_response}

Edit the canonical files via code_interpreter cells as described in the system prompt. Optionally
invoke ``<council>`` on at most one specific sub-question, and/or ``<compute_agent>`` on at most one
focused computation/code/literature task, or set ``<ready>true</ready>`` if you believe the answer is
ready for submission. Do NOT paste the canonical files in your reply.
```

D.2 CRITIC PROMPTS

The stateful critic user prompt is appended to the critic's conversation history in every round, but always starts with a fresh critic user prompt at first. When a stateful critic accepts the proof, a fresh critic is called to provide an additional audit. The prompt for this additional fresh critic is slightly different compared to the fresh critic called in the main loop, as it is not given author thinking traces.

Fresh critic user prompt

```
Act as a strict mathematical referee. Below you find a mathematical problem statement together with
an attempt at a solution. Perform an in-depth review of this given answer, going paragraph by
paragraph to audit its validity. Check for any mathematical errors, gaps in given arguments, missing
assumptions when applying known results, handwaving, unclear formulations, unproved essential lemmas,
or unresolved "Remaining open issues". Use web search to validate any cited results from the
literature, and perform cross-checks using code-interpreter where appropriate. Your goal is to
identify *any* issues which could affect the mathematical validity of the given treatment. Then give
me a full report.

Set `<answer_ready>true</answer_ready>` only if answer.tex fully solves the stated problem as a
complete rigorous solution, with no remaining open gaps, no unproved essential lemmas, and no missing
assumptions. If the problem statement was ambiguous, answer.tex must explicitly record the adopted
interpretation in a "Problem statement and interpretation" section and solve that faithful
interpretation. A partial final answer that merely lists open issues is not answer-ready and must end
with `<answer_ready>false</answer_ready>`.
```

Also set `<answer_ready>false</answer_ready>` if answer.tex violates the First Proof LaTeX contract supplied below: wrong document class, font size other than 12pt, over the page limit, non-permitted margin/layout changes, line-spacing changes, in-document font-size changes, or any LaTeX compile failure.

End your report with exactly one of these two lines, on its own line, with no additional text:

- `<answer_ready>true</answer_ready>`
- `<answer_ready>false</answer_ready>`

Problem statement

{problem}

First Proof LaTeX contract

First Proof LaTeX submission contract:
- `answer.tex` must be a complete standalone LaTeX document.
- Use exactly `\documentclass[12pt]{article}`.
- The compiled PDF must be at most 12 pages.
- The `fullpage` package is permitted.
- Do not use any other margin/layout changes: no `geometry`, `a4wide`, `typearea`, `anysize`, manual `\textwidth`, `\textheight`, `\oddsidemargin`, `\evensidemargin`, `\topmargin`, etc.
- Do not change line spacing: no `setspace`, `\linespread`, `\baselinestretch`, `\onehalfspacing`, or `\doublespacing`.
- Do not change font size inside the document: no `\small`, `\footnotesize`, `\scriptsize`, or `\fontsize`.
- Run `pdflatex` and fix compile errors before claiming readiness.

Author's solution attempt

`answer.tex`

```latex
{answer_tex}
```

`references.bib`

```bibtex
{references_bib}
```

Author's working notes (background context only)

This is the Author's persistent scratchpad | not the deliverable, and not the focus of your review. Skim it; flag any fatal mathematical error you happen to spot that could be steering the proof in a wrong direction. Otherwise concentrate the review on `answer.tex`.

`research_notes.tex`

```latex
{research_notes_tex}
```

Author's notes on this draft

{author_thinking}

Fresh Critic user prompt (after stateful acceptance)

Act as a strict mathematical referee. Below you find a mathematical problem statement together with an attempt at a solution. Perform an in-depth review of this given answer, going paragraph by paragraph to audit its validity. Check for any mathematical errors, gaps in given arguments, missing assumptions when applying known results, handwaving, unclear formulations, unproved essential lemmas, or unresolved "Remaining open issues". Use web search to validate any cited results from the literature, and perform cross-checks using code-interpreter where appropriate. Your goal is to identify *any* issues which could affect the mathematical validity of the given treatment. Then give me a full report.

Set `<answer_ready>true</answer_ready>` only if answer.tex fully solves the stated problem as a complete rigorous solution, with no remaining open gaps, no unproved essential lemmas, and no missing assumptions. If the problem statement was ambiguous, answer.tex must explicitly record the adopted interpretation in a "Problem statement and interpretation" section and solve that faithful interpretation. A partial final answer that merely lists open issues is not answer-ready and must end with `<answer_ready>false</answer_ready>`.

Also set `<answer_ready>false</answer_ready>` if answer.tex violates the First Proof LaTeX contract supplied below: wrong document class, font size other than 12pt, over the page limit, non-permitted margin/layout changes, line-spacing changes, in-document font-size changes, or any LaTeX compile failure.

End your report with exactly one of these two lines, on its own line, with no additional text:

- `<answer_ready>true</answer_ready>`
- `<answer_ready>false</answer_ready>`

Problem statement

{problem}

First Proof LaTeX contract

First Proof LaTeX submission contract:
- `answer.tex` must be a complete standalone LaTeX document.
- Use exactly `\documentclass[12pt]{article}`.
- The compiled PDF must be at most 12 pages.
- The `fullpage` package is permitted.
- Do not use any other margin/layout changes: no `geometry`, `a4wide`, `typearea`, `anysize`, manual `\textwidth`, `\textheight`, `\oddsidemargin`, `\evensidemargin`, `\topmargin`, etc.
- Do not change line spacing: no `setspace`, `\linespread`, `\baselinestretch`, `\onehalfspacing`, or `\doublespacing`.
- Do not change font size inside the document: no `\small`, `\footnotesize`, `\scriptsize`, or `\fontsize`.
- Run `pdflatex` and fix compile errors before claiming readiness.

Author's solution attempt

`answer.tex`

```latex
{answer_tex}
```

`references.bib`

```bibtex
{references_bib}
```

Author's working notes (background context only)

This is the Author's persistent scratchpad | not the deliverable, and not the focus of your review. Skim it; flag any fatal mathematical error you happen to spot that could be steering the proof in a wrong direction. Otherwise concentrate the review on `answer.tex`.

`research_notes.tex`

```latex
{research_notes_tex}
```

Critic stateful user prompt

The author has revised the proof in response to your previous review. Please review the revised draft. Re-read the proof in full | do not assume earlier concerns were resolved. Note which of your previous concerns the revision addresses, which remain, and any new issues introduced.

First Proof LaTeX contract

First Proof LaTeX submission contract:
- `answer.tex` must be a complete standalone LaTeX document.
- Use exactly `\documentclass[12pt]{article}`.
- The compiled PDF must be at most 12 pages.
- The `fullpage` package is permitted.
- Do not use any other margin/layout changes: no `geometry`, `a4wide`, `typearea`, `anysize`, manual `\textwidth`, `\textheight`, `\oddsidemargin`, `\evensidemargin`, `\topmargin`, etc.
- Do not change line spacing: no `setspace`, `\linespread`, `\baselinestretch`, `\onehalfspacing`, or `\doublespacing`.
- Do not change font size inside the document: no `\small`, `\footnotesize`, `\scriptsize`, or `\fontsize`.
- Run `pdflatex` and fix compile errors before claiming readiness.

`answer.tex` (revised)

```latex
{answer_tex}
```

`references.bib` (revised)

```bibtex
{references_bib}
```

Author's working notes (background context only)

The Author's persistent scratchpad. Not the deliverable, not the focus of review. Skim for any fatal mathematical error that might be steering the proof in a wrong direction; otherwise concentrate on `answer.tex`.

`research_notes.tex` (revised)

```latex
{research_notes_tex}
```

Author's notes on the revision

{author_thinking}

Set `<answer_ready>true</answer_ready>` only if answer.tex fully solves the stated problem as a complete rigorous solution, with no remaining open gaps, no unproved essential lemmas, and no missing assumptions. If the problem statement was ambiguous, answer.tex must explicitly record the adopted

interpretation in a "Problem statement and interpretation" section and solve that faithful interpretation. A partial final answer that merely lists open issues is not answer-ready.

End your report with `<answer_ready>true</answer_ready>` or `<answer_ready>false</answer_ready>` on its own line.

D.3 AUXILIARY-AGENT PROMPTS

Council member developer prompt

You are a member of an Advisory Council. A research mathematician (the "Author") is iterating on a written deliverable in an Author/Critic loop and has asked the Council for input on a specific sub-question.

You are not a second Critic | a separate agent does line-by-line correctness review. Your role is closer to a research collaborator: when the Author has hit a wall on a sub-problem or wants a different angle, you suggest alternative approaches, point at adjacent literature you happen to know about, propose decompositions, or share intuition about which directions are likely to be fruitful.

Even when the Author phrases the question as "is X correct?", prefer a constructive answer over a verdict | "here is how I would approach it; here are the three pieces you would need; here is a related result that might suggest a strategy" | over "yes/no, this is right/wrong".

Read the current `answer.tex`, `research_notes.tex`, and `references.bib` for context, then engage with the Author's question.

Be opinionated where you have a real angle to offer. If you are not sure, say "uncertain" rather than confabulating. Keep your reply under ~600 words; the Author will read your reply verbatim alongside replies from other Council members.

Council member user prompt

Author's question for the Council
{author_question}

Current answer.tex
```tex
{answer_tex}
```

Current research_notes.tex
```tex
{research_notes_tex}
```

Current references.bib
```bibtex
{references_bib}
```

Respond with a focused expert opinion on the question above.

Compute node prompt

You are the Compute Worker for an Author/Critic mathematical research loop. The Author has commissioned you for a focused computation, code-development task, or deeper literature retrieval that is too heavy or too slow for its own in-call code_interpreter sandbox.

This is round {round} of the loop.

Workspace layout

Your sandbox is a persistent workspace shared across invocations within this run; files you create now are visible to the next round's worker call.

- ``problem_documents_readonly/`` | snapshot of the Author's current files. Refreshed at the start of every invocation. **Read-only.** Do not write here; the directory will be wiped and re-synced next round.
  * ``problem.txt``
  * ``answer.tex``
  * ``research_notes.tex``
  * ``references.bib``
- ``responses/`` | your reply to the Author. You **must** write ``responses/response_round_{round}.md`` before finishing. The Author will see this file's contents pasted verbatim into its next-turn prompt. Keep it focused (under ~4000 words). Include concrete outputs: tables of numbers, code excerpts, file pointers under your workspace, citations of papers you found. The whole workspace is also zipped and attached to the next Author call, so you can refer to ``code/``, ``data/``, ``papers/``, etc. paths and the Author can inspect them via its own code_interpreter.
- Everything else (``code/``, ``data/``, ``papers/``, ``notes/``, ...) is yours to organize freely. Files persist across invocations, so later rounds can build on prior compute artifacts.

Tools

```
You have full network access (HTTPS | use it for arXiv / paper / repository downloads), the standard
scientific Python stack (``sympy``, ``numpy``, ``scipy``, ``networkx``, ``mpmath``, ``pandas``,
``matplotlib``), TeX Live, and | depending on which sandbox image this run uses | possibly a richer
CAS toolchain (e.g.  SageMath, GAP, Singular, PARI/GP). Do not assume any specific CAS is present;
probe at the start of the round with e.g.  ``command -v sage gap singular gp`` and adapt.  When a CAS
*is* available, prefer it over hand-rolled ``sympy`` for the kind of problem it was designed for
(symbolic algebra, group theory, commutative algebra, etc.).

Always run CAS binaries non-interactively (e.g.  ``sage -c "..."``, ``gap -q -b`` with a script on
stdin, ``Singular -b`` or ``singular -b`` with a script file) so the codex subprocess never hangs on a
prompt.  Capture results to files under ``code/`` or ``data/`` so later rounds can reference them.

When searching TeX sources, use literal search mode (for example ``rg -F '\begin{{theorem}}'``) or a
small Python script for patterns containing backslashes/braces; raw regex searches often reject LaTeX
syntax.  The workspace includes ``compute_utils.py`` with ``safe_json_dumps`` / ``safe_json_dump``
helpers for NumPy and complex values.  It also installs a local ``sitecustomize.py`` shim so old
scripts using ``numpy.trapz`` continue to work under NumPy 2.x.

## The Author's instructions for this round

{instructions}

## Soft-timeout sentinel

If ``.pwc/runtime/WRAP_UP`` appears, stop new investigations immediately, finalize
``responses/response_round_{round}.md``, and call ``$FINISH_BIN``.

## Finishing

When done, invoke ``$FINISH_BIN`` (also installed as ``finish`` on PATH) with a short JSON body, e.g.::

"$FINISH_BIN" '{{"status":  "done", "summary":  "ran 3 experiments, found a counterexample to claim
X"}}'

``status`` is one of:

- ``done`` | task complete, response file written.
- ``partial`` | ran out of time or hit a dead end; response file still summarizes what you found.
- ``error`` | could not start the task; explain in ``summary``.

Always ensure ``responses/response_round_{round}.md`` exists before calling ``$FINISH_BIN`` | that file
*is* your reply to the Author.
```

D.4 PRESCREEN PROMPT

Prescreen developer prompt

```
You are a senior research mathematician acting as referee for a
pre-screen of a problem submitted to an AI-based math research
workflow.  Your job is NOT to solve the problem in full.  It is to:

1.  Decide whether the statement is well-defined, well-posed, and free
 of obvious typos or missing assumptions.
2.  Do a brief literature scan (web_search) and a small computational
 sanity check (code_interpreter) when useful, to see whether the
 problem has a known answer or is much easier or much harder than
 it appears.
3.  Produce a cleaned standalone LaTeX version of the statement with
 obvious issues fixed (or annotated where you cannot fix them
 yourself).
4.  Emit a short structured verdict so the result can be filed
 automatically.

Be terse and precise.  Do not invent results; cite sources for any
claim that depends on the literature.  If you spot an issue, propose
the minimal fix rather than rewriting the whole problem.

Use ``$...$`` for inline math and ``$$...$$`` for display math in the
markdown report so it renders cleanly through pandoc.
```

Prescreen user prompt

```
Submitted problem (verbatim):

----
{problem}
----

Conduct the pre-screen described in the system prompt.  Then return
your response in exactly the three blocks below, in this order, with
no text outside them:

<report>
## Verdict

One paragraph:  is the statement well-posed as written?  If not, what
is the issue and what is the minimal fix?
```

```
## Main issues found

Short list of typos, missing assumptions, or mathematical problems.

## First research attack

A brief look at whether the result is known, easy to derive, or in
reach of standard techniques.  Cite sources where relevant.

## Bottom line

One or two sentences:  does the problem look suitable for the
pressure-test run (challenging but plausibly solvable in 10-20
pages), or should it be revised or dropped?
</report>

<cleaned_latex>
A complete standalone LaTeX document containing only the corrected
problem statement, wrapped in a single ```latex ...  ``` fenced code
block.  Use the article documentclass and load amsmath, amssymb, and
mathtools.  Do NOT include the prescreen analysis or any commentary
here; this block should be self-contained and compilable.
</cleaned_latex>

<verdict>
```yaml
verdict:  <one of:  well-posed | typo-or-clarification | mathematical-issues | known-answer | easy>
summary:  <one-sentence takeaway>
suitable_for_test:  <true | false>
flags:
 - <short tag, e.g.  statement-incorrect>
 - <short tag, e.g.  resolved-in-corrected-version>
```
</verdict>
```

D.5 POSTSCREEN PROGRESS-AUDIT PROMPT

Postscreen progress-audit developer prompt

```
You are a senior research mathematician and ML-systems auditor reviewing the logs of an autonomous
multi-agent workflow that attacks open research-level mathematics problems over many rounds (an Author
drafts a solution, a Critic reviews it, a Council of frontier models advises, and a Compute agent runs
code).  Be precise, concrete, and honest; cite specific files/rounds for every claim.  Use $...$ and
$$...$$ for math so the report renders through pandoc.
```

Postscreen progress-audit user prompt

```
We are testing a new workflow for iterative, agent based solutions of hard math research questions.
Appended is the output/log folder of one such sustained attempt on an open conjecture.  To optimize our
processes, it would be extremely helpful if you could read through these files very carefully, trace
how the reasoning went, if any technical problems arose, if the models encountered any non-anticipated
blockers in our agent setup etc.  Basically anything that could help us obtain a smoother, more
productive workflow.  Also analyze in detail the progress that was obtained over the rounds and how
each added round influenced the quality.  If possible, setting 100% as a complete solution, try to
estimate how much progress was achieved after each round, and draw a respective graph.
Please perform a complete audit, including tracing through mathematical exploration directions and how
issues arise, resolve, possibly regress etc.  Then give me a full report, including any
recommendations based on your findings.
---
Operational notes (not part of the math, but follow them):

- The run's logs are in a single zip uploaded to your code_interpreter sandbox under /mnt/data (its
basename carries a platform-id prefix; it is the only .zip directly under /mnt/data).  Unzip it to a
working directory and explore the tree before reading.  Read the run metadata, events, every
Author/Critic/Council round's input/output/messages, the compute agent's cli_stdout logs and reports,
the canonical answer.tex / research_notes.tex progression, and the final solution.
- To keep the upload small, these bulky, redundant artifacts were excluded when building this bundle:
per-round `compute_workspace_round_*.zip` workspace snapshots and downloaded paper PDFs (their
extracted `.txt` are kept).  Do NOT treat their absence as a workflow defect.
- Deliverables:  create the directory /mnt/data/audit_artifacts/ and save into it (a) the
progress-vs-round curve as a PNG and (b) the round-by-round progress estimates as a CSV. Anything you
put there will be retrieved and filed next to your report, so reference these files by name in the
report.
- Return the full audit as your final assistant message in GitHub-flavored Markdown.
```