IMProofBench: Benchmarking AI on Research-Level Mathematical Proof Generation

Johannes Schmitt[1], Gergely Bérczi[2], Jasper Dekoninck[1], Jeremy Feusi[1], Tim Gehrunger[1]
[1]ETH Zurich [2]Aarhus University

Benchmark Contributors: Raphael Appenzeller, Jim Bryan, Niklas Canova, Timo de Wolff, Filippo Gaia, Michel van Garrel, Baran Hashemi, David Holmes, Aitor Iribar Lopez, Victor Jaeck, Martina Jørgensen, Steven Kelk, Stefan Kuhlmann, Adam Kurpisz, Chiara Meroni, Ingmar Metzler, Martin Möller, Samuel Muñoz-Echániz, Robert Nowak, Georg Oberdieck, Daniel Platt, Dylan Possamaï, Gabriel Ribeiro, Raúl Sánchez Galán, Zheming Sun, Josef Teichmann, Richard P Thomas, Charles Vial

https://improofbench.math.ethz.ch

Abstract

As the mathematical capabilities of large language models (LLMs) improve, it becomes increasingly important to evaluate their performance on research-level tasks at the frontier of mathematical knowledge. However, existing benchmarks are limited because they focus on final-answer questions or high-school competition problems. To address this, we introduce IMProofBench, a private benchmark consisting of 39 peer-reviewed problems developed by expert mathematicians. Each problem requires an LLM to produce a proof, which is then graded by the problem's author. Within an evaluation environment equipped with various tools, the best model, GPT-5, solves 22% of the problems, closely followed by GROK-4 at 19%. Importantly, an analysis of our results indicates that current LLMs can aid research mathematicians on a basic level, but still need significant supervision to avoid simple mistakes. As LLMs continue to improve, IMProofBench will evolve as a dynamic benchmark in collaboration with the mathematical community, ensuring its relevance for evaluating the next generation of LLMs.

1 Introduction

Large language models (LLMs) are making rapid progress on mathematical benchmarks [6, 18]. These improvements suggest that LLMs may soon support mathematical research by collaborating with professional mathematicians on open problems. To determine whether current LLMs are capable of contributing in such settings, benchmarks are needed that test capabilities at the frontier of mathematical research. However, existing benchmarks fall short of this objective: most focus on high-school or undergraduate mathematics [6, 15], and the few benchmarks that target research-level mathematics, like FrontierMath [18] and HLE [23], focus exclusively on final-answer problems.

This work: IMProofBench To fill this gap, we introduce IMProofBench, a private benchmark developed in collaboration with the mathematical research community to evaluate LLMs on research-level proof writing. IMProofBench includes tasks ranging from challenging oral exam questions in a graduate course to open research questions. Currently, IMProofBench consists of 39 problems developed in collaboration with over 23 mathematicians. However, unlike static benchmarks, IMProofBench is designed as a platform for continuous evaluation: problems are added on a rolling basis, ensuring its relevance for evaluating the next generation of frontier LLMs.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MATH-AI.

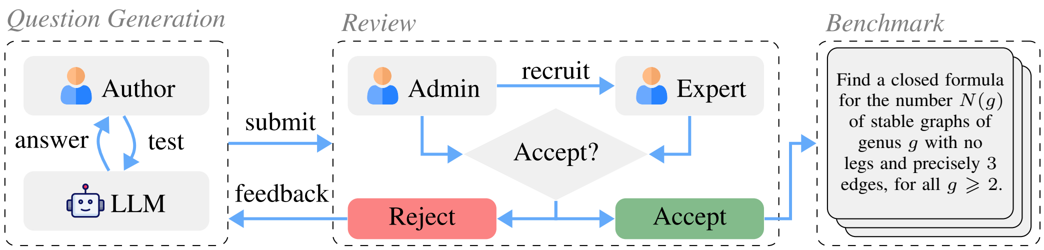

Figure 1: Workflow for question creation with peer review.

Problem creation and evaluation Each problem in IMProofBench is authored by a research mathematician within their area of expertise and undergoes a thorough review process. Alongside the main proof-writing tasks, authors are encouraged to add follow-up subquestions with final answers that can be automatically graded, enabling a comparison between final-answer and proof-based performance. Evaluation is conducted in an agentic framework designed to mirror a research environment with access to tools like Python, SageMath [1], and web search.

Key results Our results show that models can already solve a small but meaningful fraction of research-level problems: the best model, GPT-5, produces complete solutions for 22% of tasks, closely followed by GROK-4 at 19%. Beyond aggregate scores, our analysis reveals that many models are prone to reasoning errors, ranging from logical mistakes to misconceptions that professional mathematicians would not make. At the same time, they also show a wide-ranging familiarity with existing literature and can often provide insights that could meaningfully support mathematicians.

Core contributions The core contributions of this work are:

- IMProofBench, a private and evolving benchmark for research-level problems.
- A systematic analysis of proof generation capabilities across state-of-the-art LLMs.
- A qualitative analysis discussing the difficulties and strengths of current state-of-the-art LLMs.

2 Benchmark Methodology

2.1 Problem Creation Pipeline

Problem creation As shown in Fig. 1, authors draft questions through a web interface and can immediately test them on an instance of GPT-5, allowing quick, optional feedback on both difficulty and potential ambiguities. Where possible, authors are asked to include follow-up subquestions with unique, automatically gradable answers, with the option to assign point weights for the solution of different subquestions to reflect their difficulty or importance. This facilitates broader evaluation of more models by reducing reliance on human grading, while also supporting comparisons between final-answer accuracy and proof-generation capability. To guide contributions, authors receive detailed instructions that include illustrative examples and emphasize that questions should require PhD-level insight. A complete description of the author instructions is provided in App. B.2.

Problem peer-review process Once a question is submitted, an administrator recruits a reviewer whose expertise aligns with the problem's subject area. The review process follows an academic peer-review model, with the administrator and reviewer providing detailed feedback, asking for revisions where necessary. While the reviewer concentrates on verifying mathematical correctness and difficulty, the administrator ensures that the submission adheres to the guidelines. Authors are then invited to revise their problem and respond to comments with clarifications or adjustments. A problem is accepted only after both the administrator and reviewer have no remaining concerns. A full description of the reviewer instructions is given in App. B.3.

2.2 Model Evaluation

Evaluation As shown in Fig. 2, models are evaluated within an agentic framework that approximates real research conditions. We use the Inspect framework [2] and give models access to a diverse set of tools, such as web search, Python, and SageMath [1]. A description of these tools is provided in App. E. To submit an answer, models must use a dedicated submit tool, which ensures a clear distinction between reasoning steps and the final output. Each model is allocated up to 300,000 tokens for main questions, with an additional 100,000 tokens available for each follow-up subquestion.

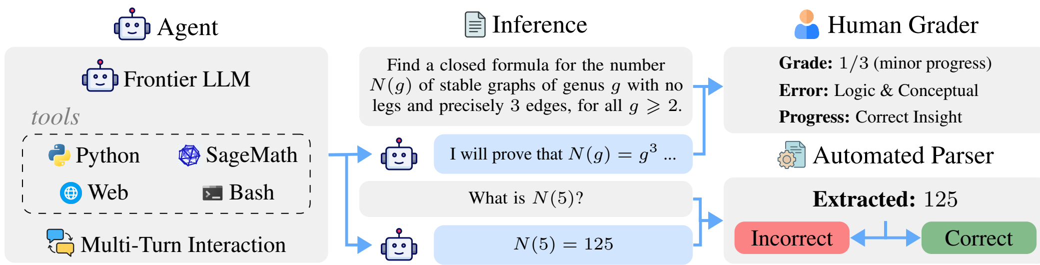

Figure 2: Evaluation workflow in a multi-turn environment with research tools.

Model selection and tiers To ensure scalability, we adopt a tiered evaluation system. Each model is assigned to a tier that reflects its priority for human grading, allowing question authors to focus on the most important submissions when their time is limited. The highest-priority tier includes LLMs that demonstrate strong performance on existing benchmarks: GPT-5 [22], GEMINI-2.5-PRO [14], GROK-4 [26], and CLAUDE-OPUS-4.1 [3]. A complete description of the tiers is provided in App. D.

Grading process Scoring of model answers takes place in two separate stages. First, follow-up subquestions are automatically graded by comparing the model's output with the ground-truth reference. In the second stage, human grading is conducted through our dedicated web interface. The question's author serves as the grader and provides three types of feedback:

- **Error classification**: marking whether the model solution contains incorrect logic, hallucinations, calculation errors, or conceptual misunderstandings.
- **Achievement indicators**: marking whether the model demonstrated understanding, reached correct conclusions, identified key insights, or produced useful reasoning.
- **Overall progress**: assigning a score of no (0/3), minor (1/3), major (2/3), or full (3/3) progress.

2.3 Benchmark Statistics and Future Development

State of the benchmark IMProofBench is under active development and currently consists of 39 questions and 79 follow-up subquestions. Topics range from areas of pure mathematics, such as algebraic geometry, combinatorics, and graph theory, to applied subjects such as stochastic analysis and bioinformatics. Of the 39 benchmark problems, authors characterize 7 as open research questions.

Continuous development To ensure that IMProofBench remains challenging, we are committed to its continuous development along several dimensions. First, we will maintain our problem creation pipeline and accept problems on a rolling basis. Second, to prevent contamination, we will use a problem management system in which authors can revisit their problems once new publications make them significantly easier. Other ideas for future work are given in App. F.

3 Experimental Results

3.1 Main Results

Proof-based evaluation As shown in Fig. 3, GPT-5 achieves the strongest performance, producing a complete solution in 22% of cases. It fails to make any progress on only 17% of the questions, showing that the model can engage meaningfully with most problems in the benchmark. These results highlight both the impressive capabilities of current systems and the difficulty of IMProofBench, as substantial progress remains possible. Importantly, none of the 7 open problems were solved.

Final-answer evaluation In Fig. 4, we show the performance of the models on the follow-up subquestions in IMProofBench, averaging using the author-appointed weights. GROK-4 achieves the highest performance, with an almost 10% margin over the second-ranked model, GPT-5. Importantly, GROK-4 tends to produce very short answers, which reduces its score in proof-based evaluation. Automated grading of final-answer questions does not penalize this, giving GROK-4 the advantage.

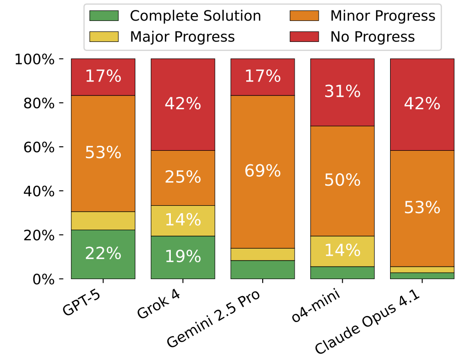

Figure 3: Proof-based results on IMProofBench.

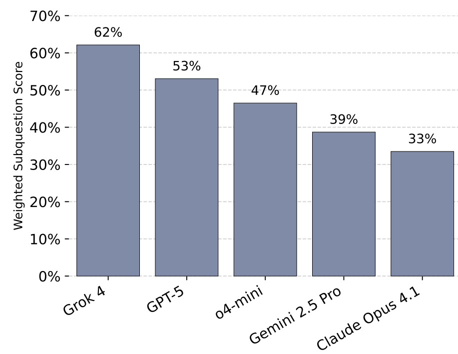

Figure 4: Final-answer results on IMProofBench.

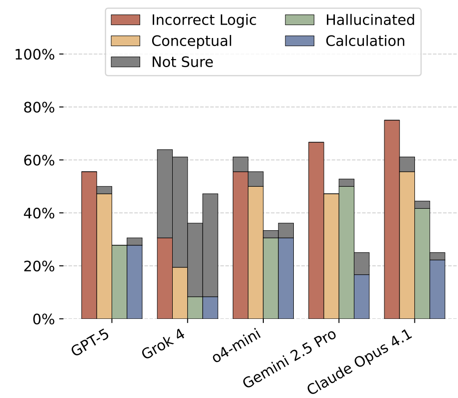

Figure 5: Error classification

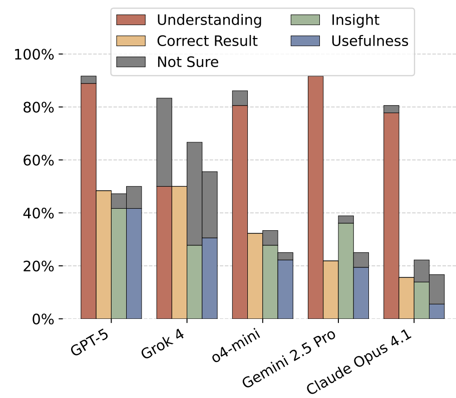

Figure 6: Achievement indicators

Error classification As shown in Fig. 5, models make a wide variety of errors. Importantly, the common conceptual errors are described in the grader guidelines as fundamental misunderstandings of mathematical concepts, showing that models do not fully understand some advanced mathematical concepts. Furthermore, hallucinations are surprisingly frequent, with GEMINI-2.5-PRO hallucinating results in 50% of its answers. A notable outlier among all models is GROK-4: it often produces extremely short answers that only contain a final answer attempt without supporting arguments. This leads graders to be unsure about the precise mistakes or achievements in its reasoning.

Achievement indicators As shown in Fig. 6, models demonstrate familiarity with the required background knowledge, which is an impressive achievement given that many of these questions use highly specialized mathematical concepts. Creative ideas are rarer, but GPT-5 still displays non-trivial creativity in almost half its solutions. This indicates that models can already make remarkable progress on difficult problems. Finally, in some cases, models provide insights that could be helpful to mathematicians, with GPT-5 offering meaningful contributions in almost half its attempts.

3.2 Qualitative Analysis

Mistakes are often hidden Models often add just a single incorrect simplifying assumption or claim that makes the problem significantly easier, but leads to incorrect conclusions. Importantly, they are usually presented with confidence and framed rhetorically, for example, by stating that a "well-known result" implies a key step. Sometimes, different models even independently converge on the *same* wrong shortcut. Importantly, models also rarely abstain from answering a question, preferring to provide an incorrect but convincing proof instead.

User testimonials For many contributors, this benchmark was their first hands-on experience with state-of-the-art LLMs in an agentic setup. Participants at outreach events expressed surprise at the level of performance (*"Quite impressive, especially the case of degree 3 where one has to argue a little bit..."*). Furthermore, during grading, we found that some models applied new approaches to known problems, surprising the expert graders (*"Interestingly, I was not familiar with the correct solution from the models, even though it is relatively fundamental."*).

4 Conclusion

In this paper, we introduced IMProofBench, a benchmark designed to evaluate research-level proof-writing capabilities in LLMs. Each problem is authored and peer-reviewed by professional mathematicians, and evaluation takes place in an agentic framework that mirrors a real research environment. Our experiments with state-of-the-art LLMs show that models can already solve a meaningful subset of research-level problems, with GPT-5 solving 22% of the questions.

Acknowledgments

Acknowledgments We would like to thank Honglu Fan for his help and support throughout the project, including advice and feedback at the early planning stages. We are also deeply grateful to the IT Support group of the D-MATH at ETH, and in particular to Michele Marcionelli, who provided competent, patient and prompt support in setting up our benchmarking website.

We also thank the organizers of the 2025 Helvetic Algebraic Geometry Seminar, Oberwolfach Workshop ”Recent Trends in Algebraic Geometry” and the ITS workshop ”Computations in Algebraic Geometry: Complex, Real, and Tropical” for providing outreach platforms to advertise the benchmark in the form of satellite events and during poster sessions.

Financial acknowledgments (Project) We acknowledge that our project received support in the form of free credits for both the xAI and the Gemini APIs, for which we thank the teams at the respective companies.

Financial acknowledgments (Contributors) Various contributors where supported by grants:

- David Holmes is supported by ERC grant EAGL.
- Baran Hashemi is supported by the Excellence Cluster ORIGINS, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy-EXC2094-390783311.
- Chiara Meroni is supported by Dr. Max Rössler, the Walter Haefner Foundation, and the ETH Zürich Foundation.
- Johannes Schmitt was supported by SNSF grant 200020-219369 and SwissMAP.
- Charles Vial is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 491392403 – TRR 358.
- Gergely Bérczi was supported by DFF Grant 40296.

References

[1] Sagemath — open-source mathematical software system. `https://www.sagemath.org/`, 2025. Accessed: 2025-09-15.

[2] UK AI Security Institute. Inspect AI: Framework for Large Language Model Evaluations, 2024. URL `https://github.com/UKGovernmentBEIS/inspect_ai`.

[3] Anthropic. Claude 4.1 system card (addendum). Technical report, Anthropic, August 2025. Last updated September 15, 2025. Available at `https://assets.anthropic.com/m/4c024b86c698d3d4/original/Claude-4-1-System-Card.pdf`.

[4] Benjamin Assarf, Ewgenij Gawrilow, Katrin Herr, Michael Joswig, Benjamin Lorenz, Andreas Paffenholz, and Thomas Rehn. Computing convex hulls and counting integer points with polymake. *Math. Program. Comput.*, 9(1):1–38, 2017. doi: 10.1007/s12532-016-0104-z. URL `http://dx.doi.org/10.1007/s12532-016-0104-z`.

[5] V. Baldoni, N. Berline, J. A. De Loera, B. Dutra, M. Köppe, S. Moreinis, G. Pinto, M. Vergne, and J. Wu. *A User's Guide for LattE integrale v1.7.2*, 2013. URL `https://www.math.ucdavis.edu/~latte/`.

[6] Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanovic, and Martin T. Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *CoRR*, abs/2505.23281, 2025. doi: 10.48550/ARXIV.2505.23281. URL `https://doi.org/10.48550/arXiv.2505.23281`.

[7] Peter Belcak, Ard Kastrati, Flavio Schenker, and Roger Wattenhofer. Fact: Learning governing abstractions behind integer sequences. *Advances in Neural Information Processing Systems*, 35: 17968–17980, 2022.

[8] Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Christoph Graczyk, Katrin Halbig, Ivo Hedtke, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Julian Manns, Gioni Mexi, Erik Mühmer, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Dieter Weninger, and Lixing Xu. The SCIP Optimization Suite 9.0. ZIB-Report 24-02-29, Zuse Institute Berlin, February 2024. URL `https://nbn-resolving.org/urn:nbn:de:0297-zib-95528`.

[9] Nathan Bowler and Johannes Carmesin. Proofbench: a benchmark suite for mathematical proof verification and generation. in preparation.

[10] W. Bruns, B. Ichim, C. Söger, and U. von der Ohe. Normaliz. algorithms for rational cones and affine monoids. Available at `https://www.normaliz.uni-osnabrueck.de`.

[11] Stéphane D'Ascoli, Pierre-Alexandre Kamienny, Guillaume Lample, and Francois Charton. Deep symbolic regression for recurrence prediction. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 4520–4536. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/d-ascoli22a.html`.

[12] Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer, 2021. doi: 10.1007/978-3-030-79876-5_37. URL `https://doi.org/10.1007/978-3-030-79876-5_37`.

[13] 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.

[14] Google DeepMind. Gemini 2.5 pro model card. Technical report, Google DeepMind, June 2025. URL https://storage.googleapis.com/model-cards/documents/gemini-2.5-pro.pdf. Last updated: June 27, 2025; Accessed: 2025-09-21.

[15] Simon Frieder, Luca Pinchetti, Alexis Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. Mathematical capabilities of chatgpt. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/58168e8a92994655d6da3939e7cc0918-Abstract-Datasets_and_Benchmarks.html.

[16] *GAP – Groups, Algorithms, and Programming, Version 4.14.0*. The GAP Group, 2024. URL https://www.gap-system.org.

[17] Thibault Gauthier and Josef Urban. Learning program synthesis for integer sequences from scratch. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7670–7677, 2023.

[18] Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, Natalie Stewart, Bogdan Grechuk, Tetiana Grechuk, Shreepranav Varma Enugandla, and Mark Wildon. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in AI. *arXiv*, 2024.

[19] Maxima. Maxima, a computer algebra system. version 5.48.0, 2025. URL https://maxima.sourceforge.io/.

[20] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014. ISSN 0747-7171. doi: 10.1016/j.jsc.2013.09.003. URL https://www.sciencedirect.com/science/article/pii/S0747717113001193.

[21] OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences, 2025. Published electronically at http://oeis.org.

[22] OpenAI. Gpt-5 system card. Technical report, OpenAI, August 2025. URL https://cdn.openai.com/gpt-5-system-card.pdf. Accessed: 2025-09-21.

[23] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, Adam Khoja, Ryan Kim, Jason Hausenloy, Oliver Zhang, et al. Humanity's last exam. *arXiv*, 2025.

[24] Johannes Schmitt and Jason van Zelm. Intersections of loci of admissible covers with tautological classes. *Selecta Math. (N.S.)*, 26(5):Paper No. 79, 69, 2020. ISSN 1022-1824. doi: 10.1007/s00029-020-00603-4. URL https://doi.org/10.1007/s00029-020-00603-4.

[25] *PARI/GP version 2.17.2*. The PARI Group, Univ. Bordeaux, 2024. available from http://pari.math.u-bordeaux.fr/.

[26] xAI. Grok 4 model card. Technical report, xAI, August 2025. Last updated August 20, 2025. Available at https://data.x.ai/2025-08-20-grok-4-model-card.pdf.

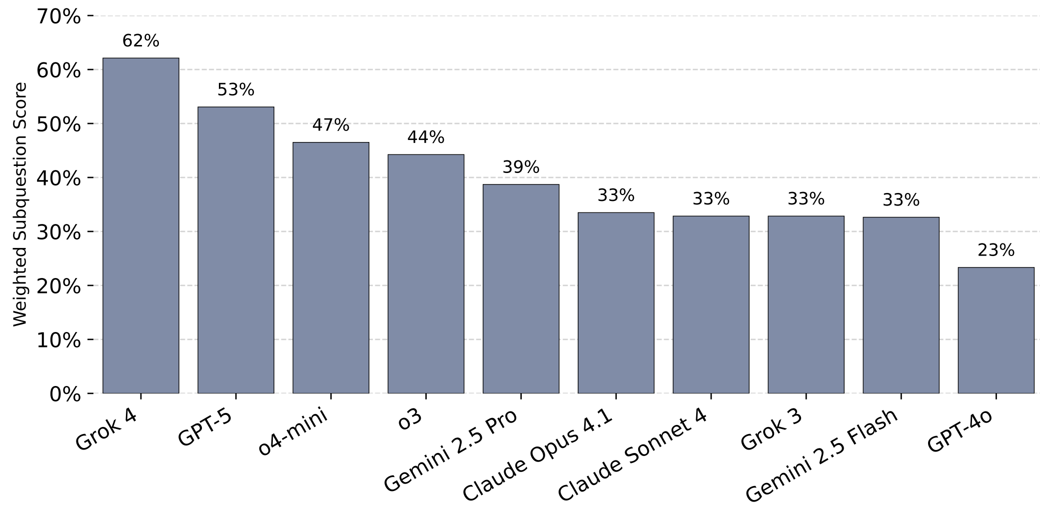

Figure 7: Average percentage of points for subquestion evaluation. Here performance on any individual question is weighted by the point rewards determined by the problem author.

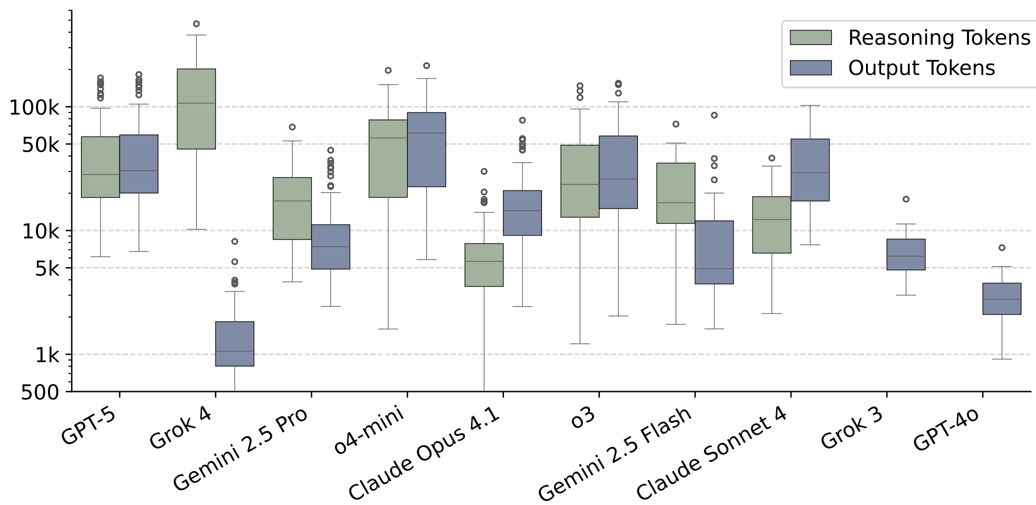

Figure 8: Token usage distribution for problem evaluation (main question and subquestions) for all tested models.

A Additional results

Performance on final-answer subquestions In Fig. 7, we present the average scores obtained by all 10 evaluated models on the final-answer subquestions, using the author-appointed weights that reflect importance or difficulty. Conclusion are similar to those presented in §3.

Token usage In Fig. 8, we show the distribution of reasoning and output tokens across the evaluated questions. GROK-4 produces the longest reasoning traces but the shortest outputs among all models in the benchmark, consistent with the trend described in §3.1. In contrast, the OpenAI models show a more balanced ratio of reasoning to output tokens. The Gemini models use slightly more reasoning tokens, while the Claude models generate more verbose outputs. With respect to token limits, which allow 300k tokens for the main question and 100k tokens for each subquestion, models almost always remain well below these thresholds.

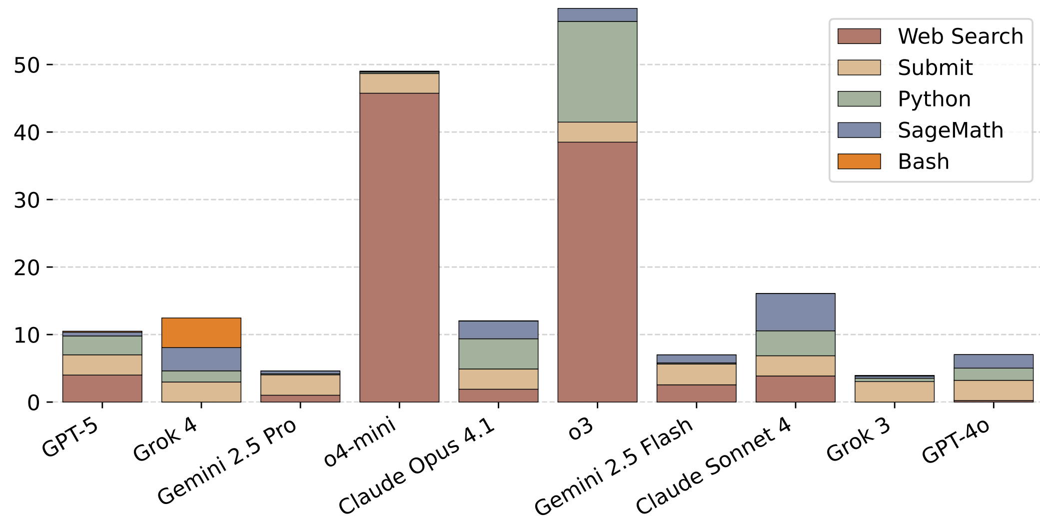

Figure 9: Average tool use per question for all tested models.

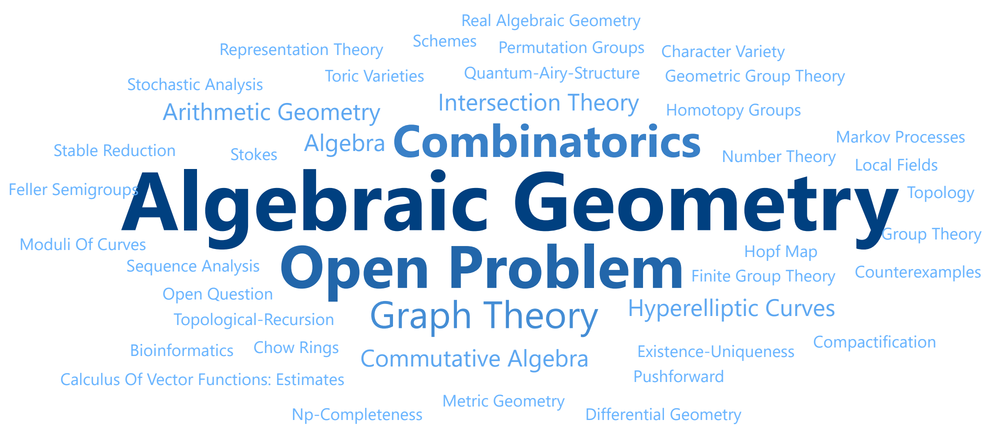

Figure 10: Word cloud of tags assigned to IMProofBench problems.

Tool usage In Fig. 9, we show the average tool usage across models. The patterns differ substantially. O4-MINI and O3 each make around 50 tool calls per problem, relying more heavily on the web search tool than any other model in the leaderboard. Further, GROK-4 is the only model that makes frequent use of the bash tool. Examining its logs reveals regular use of the command line to download research papers from arXiv (via `wget` or `curl`) and to process them with utilities such as `strings` or `gs`. This sometimes provides an advantage over models that depend solely on internal search tools. Other models display broadly similar usage patterns, distributing their calls among web search, Python, and SageMath.

Topics in IMProofBench In Fig. 10, we display the distribution of problem tags in IMProofBench. The topic of ”Algebraic Geometry” currently dominates, reflecting the research focus of the benchmark organizers. These organizers both contributed problems themselves and solicited input primarily from colleagues in their own academic networks. Future development of the benchmark will aim to broaden its coverage to include a wider range of topics in pure and applied mathematics, as outlined in App. F.

B Human Interface and Instructions

In this appendix, we discuss how contributors and benchmark administrators interact with IMProof-Bench, including the instructions and interface for different steps of the submission process (question generation, review and grading). In App. B.1, we give a brief overview of the main pages on the web interface. Then, in App. B.2, we provide details on how questions are created and edited. In App. B.3, we explain the review process. Finally, in App. B.4, we discuss the grading interface.

B.1 Submission website

Contributors submit problems via a secure website designed for submitting and reviewing questions, and grading AI answers (see Fig. 11). Features include:

- **User accounts and permissions**: Contributors can create an account tied to a (verified) email, which allows them to author questions and use website features like the free AI solution previews for these questions. Benchmark administrators have additional access to manage model evaluations, review requests and access a live view of benchmark results.
- **Community features**: The website shows a list of contributors (ordered by numbers of accepted questions or similar parameters) to encourage active participation, and links to a project Zulip with further news and opportunity to provide feedback.
- **Benchmark dashboard**: Total numbers of contributors and questions in different stages of the submission process are displayed to show project progress. An overview page with both live results and archived snapshots of the benchmark state will be added in the future.
- **About the project**: Information about the IMProofBench is provided. This information contains the initial whitepaper, overview of core team members, timeline of planned steps, and a page with frequently asked questions. A privacy policy detailing our handling of user data is linked in the footer of the page.

B.2 Question creation and editing

Benchmark problems are created through a structured interface that guides contributors through the submission requirements. The system provides comprehensive guidelines (see Figure 12) emphasizing the key characteristics of suitable benchmark problems.

Problem guidelines. Effective benchmark problems must meet several criteria:

- **PhD-level difficulty:** Problems should be suitable for oral exams of graduate courses, research papers, or advanced seminars, representing mathematics close to or at research-level.
- **Genuine mathematical insight:** Solutions must require non-routine approaches that cannot be solved through pattern matching or standard algorithm application.
- **Clear proof-based main question:** The primary answer should consist of a complete mathematical argument rather than merely a numerical result.
- **Auto-gradable subquestions:** Each problem requires 2–3 subquestions with unique answers (e.g., "Is the statement true for $n = 5$?" or "What is the rank of this group?"), enabling automated evaluation.

Contributors should avoid problems solvable by lucky guessing, standard textbook exercises (even from graduate texts), or purely computational problems that mathematical software can solve directly.

Question editing interface. The question creation and editing window (see Figure 13) provides a comprehensive authoring environment with the following components:

- **Main question editor:** A text area supporting Markdown with LaTeX mathematics, featuring a live preview pane that renders the formatted content in real-time. Contributors can use standard LaTeX delimiters (`$...$` for inline and `$$...$$` for display mathematics).
- **Problem metadata:** A tags field allows contributors to categorize problems by area (e.g., "group theory", "representation theory", or "permutation groups") and special characteristics (e.g., "open problem" for questions where the author seeks but does not know the answer).

- **AI solution preview:** Contributors can test their questions against a frontier AI model (currently GPT-5 with high reasoning effort) using up to 20 free attempts per day. This feature helps authors evaluate whether their problem has appropriate difficulty and clarity.
- **Sample solution:** A dedicated editor for the complete solution, which serves as the reference for reviewers and graders. The solution should demonstrate the expected level of rigor and detail to allow expert review to verify correctness and serve as a reference for grading model answers.
- **Subquestions management:** A dynamic form system for adding multiple subquestions, where each subquestion consists of:
  - Question text (supporting Markdown and LaTeX)
  - Expected answer field for the unique answer
  - Evaluation method selector (e.g., exact match)
  - Optional points value (defaulting to 1) for weighting subquestions by difficulty or importance
  - Rationale field for explaining the correct answer

Question detail view. Once submitted, questions are displayed in a detail view (see Figure 14) that presents all components in their rendered form. This view shows:

- The question status in the submission pipeline (Draft → Under Review → Approved → Active)
- Rendered main question and sample solution with properly formatted mathematics
- List of subquestions with their expected answers
- AI solution attempt preview when available
- Review comments from expert reviewers (when in review stage)
- Response interface allowing authors to address reviewer feedback and revise their submission

The detail view serves as the central hub for tracking a question's progress through the review process and facilitating communication between authors and reviewers.

B.3 Review process and instructions

Each question is reviewed by at least one expert before being included in the benchmark. These experts are invited to submit a review via email. An example of such an email is included below.

Reviewer invitation email

```
Dear [invited_user],

My name is [inviting_user] and I am part of a small team of mathematicians studying the question of
how good today's AI models are at solving research-level math questions. As part of this
IMProofBench project, we are building a collection of challenging mathematical problems to use for
testing the AI performance.

We would like to ask for your help in verifying the mathematical correctness of one such question.
If you are interested to learn more about the project, further information is available at https://
improofbench.math.ethz.ch/faq/

The following question was submitted for inclusion in the IMProofBench dataset:

Title: Permutation representation
Author: Example Participant

Would you be willing to review this question and:
- Verify that the phrasing is well-defined and unambiguous
- Confirm the provided solution is mathematically correct
- Make any suggestions for improvements (e.g., additional unique-answer subquestions)

We estimate that for most problems this should take between 10 and 30 minutes.

You can view the full submitted problem and write a review at:
[ACCEPT_URL]

There you will also have the option to decline this review request after viewing the question.
Alternatively, you can decline immediately by clicking:
[DECLINE_URL]
```

```
If you provide a review, the question's author will be notified and have the chance to revise the
question and compose a response. After seeing the response, you have the option to submit a further
review or recommend the question for acceptance in the benchmark.

Thank you for considering this request!

Best regards,
[inviting_user]

Note: To track your review and allow you to see the author's replies, accepting the review request
will create a user account for you on our website. You can optionally set a password after
submitting your review to log back in and e.g. contribute a question to the benchmark yourself.
```

When the reviewer accepts the review invitation by clicking on the link, they are forwarded to a webpage displaying the problem to be reviewed, along with a form for review submission and further information (see Figure 15). The reviewer may also view the full review guidelines displayed in Figure 16. The review consists of a short comment by the reviewer indicating improvements and/or mistakes in the question statement. Before submitting the review, the reviewer decides on a recommended action among the following: "Recommended for acceptance", "Needs revision" and "Not suitable". The site admins are notified when a review is complete and can take action accordingly. If the reviewer selects "Not suitable", the question is automatically reset to the "draft" status. Independently of the outcome, the author is permitted to submit an answer to the reviewer's comments and change the question if necessary. The reviewer may then either submit a new review taking into account the changes, or a new reviewer may be invited.

B.4 Grading interfaces

The grading system provides a structured interface for human evaluation of model-generated proofs through a dedicated web page.

Human grading interface. The main grading interface (see Figure 17) employs a three-column layout designed to facilitate easy access to relevant information and the feedback form:

- **Left column:** Displays the question statement and sample solution for reference
- **Center column:** Shows the model's complete response with mathematical rendering
- **Right column:** Contains the interactive grading panel with scoring controls

To prevent bias, model identities are concealed behind randomized aliases (Answer A, B, C, etc.) that remain hidden until all answers for a question have been graded. The system maintains independent grading sessions for each evaluator, with aliases shuffled differently to ensure blind evaluation.

Grading categories. The scoring form consists of three main components providing multifaceted evaluation, with relevant information available via concise tooltips:

AI Mistake Indicators: Four binary categories identifying common failure modes:

1. **Incorrect Logic:** Flawed logical steps or invalid reasoning
2. **Hallucinated:** References to non-existent theorems, papers, or results
3. **Calculation:** Arithmetic or algebraic errors
4. **Conceptual:** Fundamental misunderstanding of mathematical concepts

AI Achievement Indicators: Four binary categories recognizing positive aspects:

5. **Understanding:** Correctly identifies what needs to be proven or calculated
6. **Correct Result:** Arrives at the correct final answer (with N/A option for open-ended problems or when the correct answer is unknown)
7. **Insight:** Shows creative problem-solving or novel approaches
8. **Usefulness:** Solution would be helpful to someone learning this topic

Each binary category offers three response options: “True”, “False”, or “Not Sure”, allowing graders to indicate uncertainty when evaluation is ambiguous.

Overall Progress: A four-point scale (0–3) rating overall solution progress:

- **0/3:** No progress toward solution
- **1/3:** Minor progress with limited advancement
- **2/3:** Major progress with substantial work completed
- **3/3:** Complete solution achieved

This overall progress score serves as the primary metric for model ranking and comparison.

Additional grading features. The interface includes several supporting elements to ensure grading consistency and quality:

- **Grading notes:** A persistent text area where graders record their evaluation criteria and decision patterns across all answers (e.g., “Matrix errors count as Calculation, Theory errors as Logic”). These notes help maintain consistency when grading multiple model responses and facilitate reproducibility in future grading sessions.
- **Comments field:** Answer-specific observations about edge cases or explanations for grading decisions.
- **Auto-save functionality:** Grading selections are automatically preserved with a 2-second debounce to prevent data loss.
- **Focus mode:** An optional distraction-free interface that maximizes screen space by hiding navigation elements and allowing collapsible panels, enabling graders to concentrate on detailed evaluation.
- **Flag for organizers:** Option to mark responses requiring special attention due to serious issues or technical problems.

The grading workflow supports iterative evaluation, allowing graders to mark answers as complete, incomplete, or given up (for responses that cannot be meaningfully evaluated). Once all model answers for a question are marked complete, the system reveals the true model identities, enabling post-hoc analysis of performance patterns.

C Sample Problem

Below we present an example of a problem from the benchmark and discuss model performance and solution strategies from our evaluation.

Background for reader (not included in benchmark question) A *stable graph* is a connected graph $\widehat{\Gamma}$, multi-edges and loops allowed, together with a vertex-labeling by non-negative integers $(g_v)_{v \in V(\widehat{\Gamma})}$ satisfying that each vertex v with $g_v = 0$ has valence at least 3. These combinatorial objects appear in algebraic geometry in the study of moduli spaces of stable curves, see e.g. [24, Section 2]. The *genus* of $\widehat{\Gamma}$ is defined as $g = b_1(\widehat{\Gamma}) + \sum_{v \in V(\widehat{\Gamma})} g_v$, with b_1 the first Betti number (or cyclomatic number) of $\widehat{\Gamma}$.

Question Given an integer $g \geq 2$, let N_g be the number of isomorphism classes of stable graphs of genus g with precisely 3 edges. Give a closed formula for N_g valid for all $g \geq 2$.

Solution To compute N_g, we note that each stable graph $\widehat{\Gamma}$ has an undecorated underlying graph Γ, which is one of the 10 connected multi-graphs with precisely 3 edges. Then N_g can be calculated by summing over those graphs Γ and counting the number of assignments g_v to the vertices of Γ, avoiding double-counting by taking into account symmetries of Γ.

The final answer is that for $g = 2$ we have $N_2 = 2$ and for $g \geq 3$, we have

$$N_g = \begin{cases} \frac{1}{9}g^3 + \frac{7}{8}g^2 + \frac{5}{12}g - 2 & \text{if } g \equiv 0 \pmod 6 \\ \frac{1}{9}g^3 + \frac{7}{8}g^2 + \frac{1}{6}g - \frac{155}{72} & \text{if } g \equiv 1 \pmod 6 \\ \frac{1}{9}g^3 + \frac{7}{8}g^2 + \frac{5}{12}g - \frac{20}{9} & \text{if } g \equiv 2 \pmod 6 \\ \frac{1}{9}g^3 + \frac{7}{8}g^2 + \frac{1}{6}g - \frac{19}{8} & \text{if } g \equiv 3 \pmod 6 \\ \frac{1}{9}g^3 + \frac{7}{8}g^2 + \frac{5}{12}g - \frac{16}{9} & \text{if } g \equiv 4 \pmod 6 \\ \frac{1}{9}g^3 + \frac{7}{8}g^2 + \frac{1}{6}g - \frac{187}{72} & \text{if } g \equiv 5 \pmod 6 \end{cases}$$

Subquestions What is N_3? (Answer: 9) What is N_8? (Answer: 114) What is N_{10000}? (Answer: 111198615276)

Model approaches and performance

- GPT-5 instantly identifies the solution strategy in its first reasoning step, writing *"Essentially, I'm computing N_g as a sum of connected multigraph types limited by 3 edges and considering partitions of genera"*. It performs a Python calculation to obtain first experimental data. From theoretical considerations, it correctly identifies the shape of the final answer, writing *"Ultimately, I want a final closed formula for N_g as a degree-3 quasi-polynomial with a period of 6."*. After a few attempts it calculates this polynomial via Lagrange interpolation on datapoints with fixed residue modulo 6, discovering that the case $g = 2$ needs separate treatment. This not only represents a perfect solution to the given problem, but also mirrors precisely the approach of the human question author to solving the problem.
- GROK-4 obtains an expression for N_g in a single reasoning step, though no further details are available as the GROK-4 API does not expose reasoning summaries. The model then uses a Python tool to calculate the first values and the SageMath tool to look up the resulting integer sequence in the OEIS database [21]. This being unsuccessful it submits a very concise sketch of its answer, which is slightly less simple than the formula for N_g above, as it still features a summation over $g - 2$ terms.
In a second evaluation, GROK-4 uses the bash tool to download textbooks on algebraic graph theory and moduli spaces of curves and convert them to text. Lacking the software tools for the latter, it tries and fails to install new packages on the sandboxed docker container, receiving an error for attempting to use sudo rights. Finally it abandons these attempts and just submits a solution which is even mostly correct, but has some small error in one of the terms.
- CLAUDE-OPUS-4.1 also tries to combine combinatorial arguments with computer calculations in SageMath, but fails to find even the contribution from 2-vertex graphs, forgetting some topological possibilities for Γ. One noteworthy pattern is that the model includes very verbose reasoning in form of comments and static print statements within the SageMath code.
- GEMINI-2.5-PRO starts with a correct calculation of N_2, N_3, N_4. However, then it makes the completely unfounded claim that *"This implies that N_g is a quadratic polynomial in g."*, whereas in reality it is a *cubic quasi*-polynomial. It then submits an answer based on that wrong assumption. It does get partial credit in the subquestions for calculating $N_3 = 9$ correctly.

D Model Tiers

We evaluate models across four tiers based on their capabilities and release timeline. Tier 1 comprises current frontier models with state-of-the-art mathematical reasoning capabilities. Tier 3 includes previous generation models that have demonstrated strong mathematical performance. Tier 4 contains legacy models included for historical comparison and baseline establishment. Currently, only models in Tiers 1–3 are included in human grading to focus evaluation resources on the most relevant comparisons.[1]

All models are evaluated using the Inspect framework with standardized prompting and tool access, including Python execution, web search, and SageMath for advanced mathematical computation (see App. E). The `reasoning_effort` parameter, when specified as `"high"`, enables enhanced reasoning

[1]Tier 2 is reserved for testing Command Line Interface models such as Claude Code, but implementation has been deferred to a future version of the benchmark.

Table 1: Models evaluated in IMProofBench, organized by tier

Tier	Model	API Endpoint	Parameters
1	CLAUDE-OPUS-4.1	claude-opus-4-1-20250805	cache_prompt="auto" max_tokens=32000 reasoning_tokens=31000
	GPT-5	gpt-5	reasoning_effort="high" reasoning_summary="auto"
	GEMINI-2.5-PRO	gemini-2.5-pro	reasoning_tokens=32768
	GROK-4	grok-4-0709	—
3	O4-MINI	o4-mini-2025-04-16	reasoning_effort="high" reasoning_summary="auto"
4	CLAUDE SONNET 4	claude-sonnet-4-20250514	cache_prompt="auto" max_tokens=64000 reasoning_tokens=63000
	GPT-4O	gpt-4o-2024-11-20	—
	GEMINI-2.5-FLASH	gemini-2.5-flash	reasoning_tokens=24576
	GROK-3	grok-3	—
	O3	o3-2025-04-16	reasoning_effort="high" reasoning_history="auto" reasoning_summary="auto" reasoning_tokens=100000

capabilities for models that support it. The `reasoning_tokens` parameter controls the maximum length of the model's internal reasoning process, while `max_tokens` limits the total response length including both reasoning and final answer.

E Detailed Tool Descriptions

The evaluation environment for IMProofBench was designed to emulate the computational resources available to research mathematicians when solving complex problems. Rather than restricting models to basic arithmetic operations, we provide access to the same sophisticated mathematical software that researchers routinely use in their work. This approach reflects the reality that modern mathematical research frequently involves computational exploration, symbolic manipulation, and verification of conjectures through extensive calculation.

E.1 Technical Specifications

All tools operate within the following constraints to balance computational power with practical limitations:

- **Timeout**: 15 minutes per tool invocation
- **Memory limit**: 8 GB RAM per execution
- **Environment**: Isolated Docker container running Arch Linux
- **Execution model**: Independent tool calls (no variables persist between calls), but files written to the filesystem remain accessible throughout the evaluation session

E.2 Core Computational Tools

E.2.1 Python Environment

The Python tool provides access to a comprehensive scientific computing environment (Python 3.13.7). This language was chosen for its prevalence in scientific computing and the extensive familiarity that language models demonstrate with its syntax and libraries. The environment includes standard numerical and symbolic computation packages:

- **Numerical computing**: NumPy, SciPy, pandas
- **Symbolic mathematics**: SymPy, SymEngine
- **Visualization**: Matplotlib (though output is text-based)
- **Graph theory**: NetworkX, igraph, graph-tool
- **Optimization**: CVXPY with multiple backend solvers (GLPK, ECOS, OSQP, SCS, CSDP)
- **Machine learning**: Basic scikit-learn functionality

Each Python execution runs independently with no variables or imports preserved between invocations, though files written to disk remain accessible for subsequent tool calls.

E.2.2 Bash Shell Access

The bash tool provides command-line access to the evaluation environment, enabling models to leverage specialized mathematical software that operates through command-line interfaces. This tool serves as the gateway to domain-specific mathematical systems detailed in Section E.3.

E.2.3 SageMath

SageMath [1] (version 10.6) serves as the primary computer algebra system, providing a unified Python-based interface to numerous mathematical software packages. Its significance in the research community stems from its comprehensive coverage of mathematical domains and its philosophy of combining the best open-source mathematics software into a coherent system.

Key features available through the `sage_computation` tool include:

- Natural mathematical syntax through automatic preparsing (e.g., `x^2` for exponentiation, `K.<a>` for field extensions)
- Extensive algebraic capabilities: polynomial rings, number fields, elliptic curves, modular forms
- Combinatorial structures: graphs, matroids, posets, designs
- Specialized packages: `admcycles` for moduli spaces of curves, `ore_algebra` for D-finite functions and recurrence operators, `pari_jupyter` for enhanced PARI/GP integration
- Integration with external systems: automatic interfacing with GAP, Maxima, PARI/GP, Singular

E.3 Specialized Mathematical Software

The evaluation environment includes a comprehensive suite of specialized mathematical software, accessible through the bash tool:

E.3.1 Computer Algebra Systems

- **GAP** (Groups, Algorithms, Programming): Specialized system for computational discrete algebra, particularly group theory and combinatorics [16]
- **Maxima**: General-purpose computer algebra system for symbolic computation, descended from MIT's Macsyma [19]
- **PARI/GP** (version 2.17.2): High-performance system focused on number theory computations [25]
- **Singular**: Specialized system for polynomial computations, commutative algebra, and algebraic geometry [13]
- **Polymake** (version 4.14): System for research in polyhedral geometry and related areas [4]

E.3.2 Algebraic and Geometric Computation

- **Normaliz**: Computation of normalizations of affine semigroups and rational cones [10]
- **LattE integrale**: Lattice point enumeration and integration over convex polytopes [5]

- **Gfan**: Gröbner fans and tropical varieties computation
- **4ti2**: Algebraic, geometric, and combinatorial problems on linear spaces
- **msolve**: Polynomial system solving over finite fields and rational numbers

E.3.3 Graph Theory and Combinatorics

- **nauty and Traces**: Graph automorphism and canonical labeling [20]
- **bliss**: Another efficient graph automorphism tool
- **igraph**: Network analysis and graph algorithms library

E.3.4 Optimization Solvers

- **Linear Programming**: GLPK (GNU Linear Programming Kit), Gurobi-compatible interfaces
- **Mixed-Integer Programming**: SCIP (Solving Constraint Integer Programs) [8]
- **Semidefinite Programming**: CSDP, DSDP for SDP problems
- **SAT Solvers**: glucose, kissat, cryptominisat for Boolean satisfiability

E.3.5 Proof Assistants and Verification

- **Lean** [12]: Interactive theorem prover and functional programming language
- **Mathics**: Open-source alternative to Mathematica for symbolic computation

E.3.6 Numerical and Scientific Computing

- **Julia**: High-performance language for numerical computing
- **SciLab**: Numerical computational package similar to MATLAB
- **FLINT**: Fast Library for Number Theory
- **NTL**: High-performance number theory library

E.4 Data Resources

The environment includes numerous mathematical databases accessible through SageMath:

- Stein-Watkins database of elliptic curves
- Jones database of number fields
- Kohel database for elliptic curves and modular polynomials
- Cunningham tables for factorizations
- OEIS (Online Encyclopedia of Integer Sequences) integration
- Various polytope databases and mutation class data

E.5 Web Search Capabilities

The `web_search` tool provides access to current mathematical literature and online resources. The implementation follows a provider-based architecture:

- **Internal providers**: Models from OpenAI, Anthropic, and Grok utilize their respective built-in web search capabilities, requiring no additional API keys
- **External provider**: Tavily is configured as a fallback for models without internal search capabilities (e.g., Gemini), providing AI-optimized search results

Some models, notably GROK-4, combine web search capabilities with the `wget` bash command to download full research papers for detailed analysis.

E.6 Example tool uses from benchmark evaluation

Below we list some example tool applications that occurred during our model evaluations. In each case, the full log file of the multi-turn evaluation reveals that the respective calculation played a decisive role in allowing the model to find the correct answer. To preserve benchmark privacy, we describe the relevant tool uses in general terms while leaving out the details of the specific benchmark problem.

- **Generating functions** (Model: GROK-4, Tool: SageMath)
  Solved combinatorics problem by calculating a generating function $F(x)$ and forming the exponential $G(x) = \exp(F(x))$ to extract a specific coefficient from G
- **Modular forms** (Model: GROK-4, Tool: SageMath)
  Compute q-expansion of the weight 12 cusp form Δ
- **Group theory** (Model: GPT-5, Tool: [16] via Bash Shell)
  Accessed entries of the character table of a sporadic group
- **Literature access** (Model: GROK-4, Tool: Bash Shell)
  Model uses `curl` to download pdf of paper from arXiv, installs the `PyPDR2` package via `pip` and converts the pdf to text to obtain relevant information for the benchmark problem. Note: after an initial failed attempt at installing the `PyPDR2` package, the model uses the `pip` argument `--break-system-packages` to force a user installation in the externally managed Python environment of our sandboxed evaluation environment.

F Plans for future development

Below we give further details on our plans for the continuous development of IMProofBench.

- **Scale and outreach**: We aim to expand the benchmark to 150–300 problems, e.g. through strategic partnerships with leading mathematical institutions (e.g., MFO Oberwolfach, IAS, Fields Institute) and by recruiting domain-specific ambassadors who can promote participation at conferences and within their research networks.
- **Quality assurance and grading**: To strengthen the scientific validity of our evaluations, we will study inter-rater reliability by comparing expert gradings on the same problems. We will support graders via AI-assisted pre-screening of model answers and refine our error classification system to localize specific mistakes within solution texts rather than applying only global categories.
- **Dynamic problem management**: As mathematical knowledge evolves, problems may become easier due to new publications or techniques. We will implement a generous retirement policy allowing authors to withdraw problems affected by recent research, while regularly adding fresh problems to maintain benchmark difficulty. We also plan to release small sets of sample problems to provide the community with concrete reference points for gauging AI progress.
- **Technical innovation**: We plan to develop automated difficulty classifiers to predict which problems challenge current AI systems, explore alternative evaluation formats (such as formula reconstruction tasks and interactive problem-solving sessions), and implement bring-your-own-agent interfaces to enable companies to test internal models against the benchmark.
- **Model coverage**: Beyond proprietary frontier models, we will evaluate leading open-source reasoning systems like DeepSeek-V3.1-Terminus and Qwen3-235B-Think, promoting the strongest to Tier 1 status for human grading, ensuring long-term comparison baselines even as commercial models are deprecated. *Note*: One reason why these models were not included in this initial version of the benchmark is the ongoing challenges with enabling tool use for these models - a requirement to put them on equal footing with other models within the inspect evaluation framework of IMProofBench.
- **Evaluation modalities**: Building on the existing IMProofBench platform and contributor network, we plan to explore further problem types and evaluation methodology. This includes:
  - combinations of informal and formalized questions and solutions (e.g. in collaboration with the ProofBench project [9]),
  - specialized task formats with wide importance to mathematical research, such as formula reconstruction for sequence data of natural/rational numbers, polynomials, . . . (see e.g. [17, 7, 11]),

– interactive or collaborative proof attempts, including provision of hints or feedback to model during evaluation time, more closely mimicking the setting of a researcher using commercially available AI systems.

G Use of Large Language Models

We report our use of LLMs throughout this research project. The authors take full responsibility for all content in this paper, including any LLM-assisted portions.

G.1 Writing and Presentation

Claude Opus 4.1 was used to generate an initial draft of Sections 3 and Appendices B, D, E and G, and provided feedback and suggestions for our Reproducibility and Ethics Statements. Additionally, Claude Opus 4.1 provided proofreading assistance and offered stylistic and structural suggestions. GPT-5 was used to collect bibliography entries of software packages in Appendix D. All LLM-generated content was thoroughly reviewed, fact-checked, and edited by the authors.

G.2 Literature Discovery and Related Work

During the ideation phase and preparation of the benchmark whitepaper, we used ChatGPT o3 and Claude Opus Research to conduct comprehensive searches of the benchmarking literature and identify related projects. These tools helped surface relevant prior work and ensure thorough coverage of the existing landscape, though all citations were independently verified by the authors.

G.3 Research Implementation and Development

LLMs played a substantial role in implementing the benchmark infrastructure. Claude Code, supplemented by ChatGPT's Codex CLI tool, assisted with:

- Development of the benchmark website and database architecture
- Adaptation of the Inspect framework for model evaluation
- Extraction and visualization of quantitative results

Additionally, Claude Opus provided support for organizational tasks including meeting summaries and creation of promotional materials. The background image for our benchmark poster was generated using GPT-5's multimodal capabilities.

All code and implementations were tested, validated, and debugged by the authors to ensure correctness and functionality.

H Evaluation Prompts

Main Question Prompt

```
# Background
The IMProofBench project is a mathematical reasoning benchmark for AI systems, testing their ability
 to solve research level math problems. Each such problem consists of one **main question**, where
the expected answer is a longform mathematical proof, and several related **subquestions** which
have short, unique answers (e.g. a natural number). The main answer will be graded by both human
expert mathematicians (often the author of the question) and AI evaluators, whereas subquestion
answers are checked automatically using a Python script.

# Structure of the evaluation
In the following we would like to evaluate your mathematical reasoning abilities on one such problem.
 The overall structure of the conversation below is that we iterate through the questions in order (
main question, sub-question 1, sub-question 2, ...) and in each step, you can:

- Read the current question
- Think about it in a multi-turn environment with tool use (see below)
- Submit the answer to the current question

At each point in the conversation, you have the context of the entire previous conversation
including your outputs in the thinking steps and the record of any tool uses. Note that you will *
```

```
not necessarily* have access to records of your internal reasoning traces and internal tool uses, so
 any helpful information from these should be documented in your (external) thinking outputs.

# Multi-turn reasoning environment
To help you solve the problem, you will have access to a multi-turn conversation environment with
optional tool use, based on the Inspect AI framework. At each step, you can:

- Think out loud to analyze the problem, devise a solution approach, think through the steps of
mathematical arguments, etc.
- Use the `python` tool to run self-contained experiments in a standard python environment
- Use the `bash` tool to execute commands inside a docker container (running ArchLinux with some
open-source mathematical software installed)
- Use the `web_search` tool to search for current information, mathematical definitions, theorems,
or recent research
- Use the `sage_computation` tool for conducting an experiment in a self-contained SageMath terminal
 session
- Use the `submit` tool to provide your final answer to the current question (main or sub-question)

All tools have a timeout of 15 minutes, maximal memory usage (RAM) of 8 GB and run on standard 2025
hardware.

# Token constraints
You have {main_question_token_limit:,} tokens to solve the main question, and {
subquestion_token_limit:,} tokens for each of the following sub-questions. This counts both your
output tokens (including in tool calls) and your reasoning tokens. You are informed about your
current usage after each conversation turn.

# Answer format for main question
Below you will see the text of the main question. Once you finished reasoning about it, you can
register your answer using the `submit` tool. The answer for the main question should be a detailed
mathematical argument, formatted in Markdown with LaTeX formulas using $...$ for inline mathematical
 expressions and $$...$$ for equations. Use Markdown [link formatting](https://www.markdownguide.org
/basic-syntax/#links) for including online references, *not* any internal web-referencing system.

**Problem:**

{question_text}

Please work through this problem step by step. When you have your final answer, use the submit()
tool to submit it.
```

Subquestion Prompt

```
---

**Great work on the previous part!**

You have successfully completed the previous question. Now please solve the following subquestion
while keeping the context of your previous work:

**Subquestion {subquestion_order}:**
{subquestion_text}

**Instructions:**
- You can reference your work from previous parts
- Use the same mathematical tools available to you
- When you have your final answer, use the submit() tool to submit it
- Be precise and specific in your answer format

Please proceed with solving this subquestion.
```

Conversation Status Update

```
Please continue working on the current question. To formally register your answer, use the `submit`
tool as per the original instructions above. Note: the conversation will only proceed to the next
stage once you use the `submit` tool.

Token usage: {current_tokens:,} of {token_limit:,} tokens used for this stage.
```

Python tool description

```
Use the python function to execute Python code.
```

```
The Python tool executes single-run Python scripts. Important notes:
1. Each execution is independent - no state is preserved between runs
2. You must explicitly use print() statements to see any output
3. Simply writing expressions (like in notebooks) will not display results
4. The script cannot accept interactive input during execution
5. Return statements alone won't produce visible output
6. All variables and imports are cleared between executions
7. Standard output (via print()) is the only way to see results
8. This tool has a timeout of 15 minutes and maximal memory usage (RAM) of 8 GB
```

Bash tool description

```
Use this function to execute bash commands. Underlying system is ArchLinux with many standard open-
source computer algebra systems (like GAP) pre-installed.
This tool has a timeout of 15 minutes and maximal memory usage (RAM) of 8 GB.
```

Web search tool description

```
Use this function to search the web for current information, mathematical definitions, theorems, or
recent research.

This tool gives you access to up-to-date information that can help with:
- Looking up mathematical definitions and theorems
- Finding recent research papers or results
- Verifying computational results against known databases
- Checking current mathematical conventions or notation
- Finding examples of similar problems or techniques

The search results will include titles, URLs, and relevant excerpts from web pages.
Use this tool when you need information that might not be in your training data or when you want to
verify facts.
```

Sage tool description

```
Use the sage_computation function to run calculations in the open-source mathematics
software system SageMath.

The sage_computation tool executes single-run SageMath scripts. Important notes:
1. Each execution is independent - no state is preserved between runs
2. You must explicitly use print() statements to see any output
3. Simply writing expressions (like in notebooks) will not display results
4. The script cannot accept interactive input during execution
5. Return statements alone won't produce visible output
6. All variables and imports are cleared between executions
7. Standard output (via print()) is the only way to see results
8. This tool has a timeout of 15 minutes and maximal memory usage (RAM) of 8 GB

All standard SageMath functions are pre-imported and available.
The SageMath preparser is applied, so you can use natural mathematical syntax.

Key Features:
- Natural syntax: Use x^2 for powers, K.<a> for field extensions
- All mathematical objects pre-imported: Matrix, EllipticCurve, PolynomialRing, etc.
- Advanced packages available: admcycles for moduli spaces, and many more

Examples:
  # Factor a polynomial
  factor(x^100 - 1)

  # Define a number field
  K.<a> = NumberField(x^3 - 2)

  # Work with elliptic curves
  E = EllipticCurve([0, 1])
  print(E.rank())

  # Use specialized packages (example with admcycles)
  from admcycles import *
  G = StableGraph([1,1],[[1,3],[2,4]],[(1,2),(3,4)])
  print(f"Automorphisms^2: {G.automorphism_number()^2}")

IMPORTANT: Like the python() tool, you must use print() to see any output.
Nothing is returned automatically - always print your results!
```

Submit tool description

```
Submit your final answer for the current question or subquestion. Use Markdown + LaTeX formatting.
The answer for the main question should be a detailed mathematical argument.

Your answer should be formatted as natural Markdown text with LaTeX formulas.
Use $ for inline math and $$ for display math, or \begin{equation} environments.
Use standard [Markdown link syntax](https://www.markdownguide.org/basic-syntax/#links) for online
references.

RECOMMENDED: Use raw strings (r''' or r"") to write LaTeX naturally without escaping.

Important formatting notes:
- Write your answer exactly as you would in a math document
- Use raw triple quotes r''' for multiline answers with LaTeX
- This lets you write \frac, \sqrt, \int naturally (no escaping needed)
- Include full mathematical reasoning with the final answer clearly stated
- Do not use custom macros (e.g., \Z, \Q, \RR, etc.). Only use valid standard LaTeX commands
```

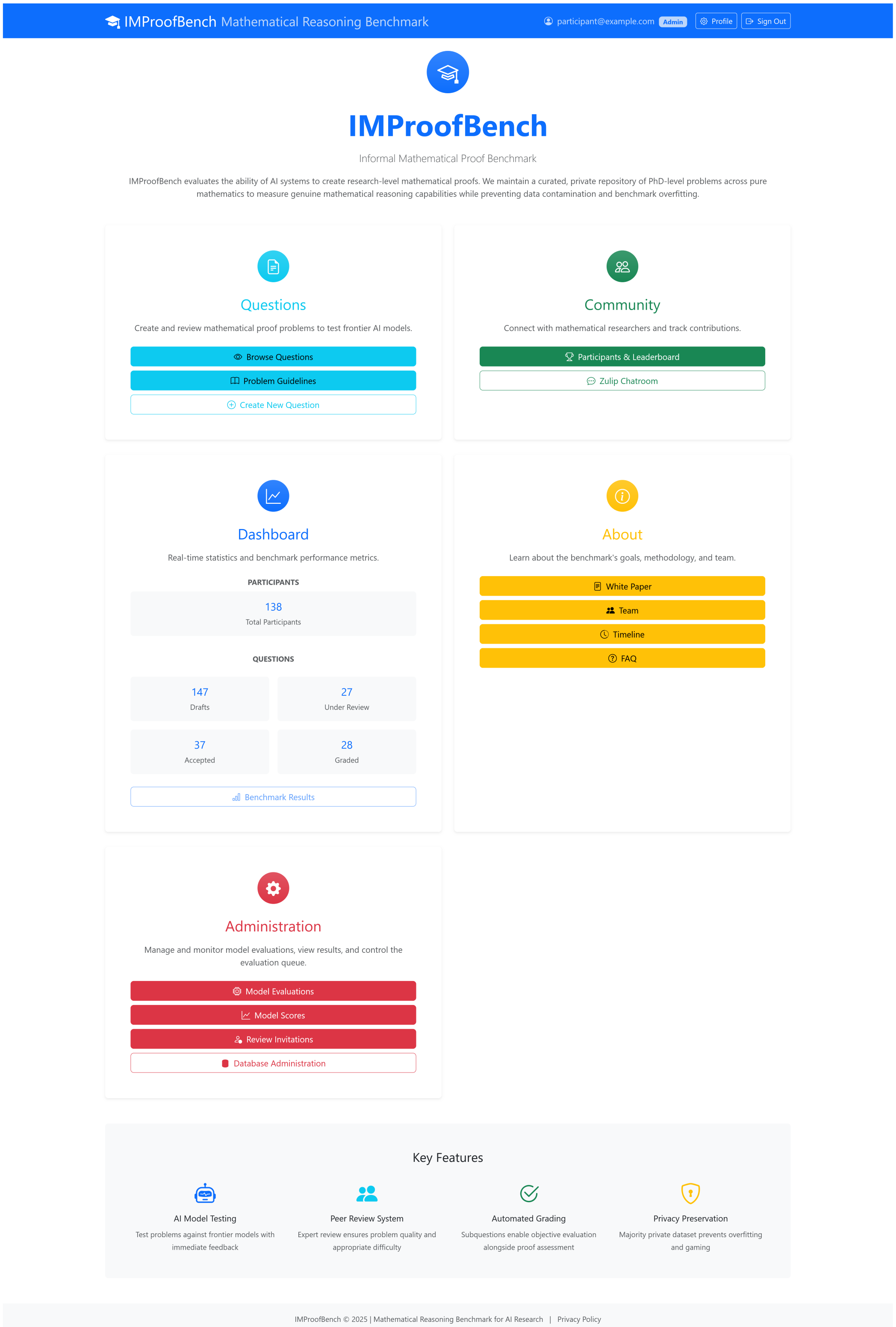

Figure 11: Landing and overview page of IMProofBench website.

IMProofBench Mathematical Reasoning Benchmark   participant@example.com Admin | Profile | Sign Out

Problem Guidelines

Creating high-quality benchmark problems for mathematical AI evaluation

Quick Start

Effective benchmark problems require **PhD-level difficulty**, **genuine mathematical insight**, and **2-3 auto-gradable subquestions**. Think about recent calculations from your research that required a clever insight or non-obvious proof techniques.

Required Characteristics

- **PhD-level difficulty:** Suitable for qualifying exams, research papers, or advanced seminars
- **Requires genuine insight:** Not solvable by routine application of known algorithms
- **Clear proof-based main question:** Answer should be a complete mathematical argument, not just a number
- **2-3 unique-answer subquestions:** Enable automated evaluation (e.g., "Is the statement true for n=5?", "What is the rank of this group?")

What to Avoid

- Problems solvable by pattern matching or lucky guessing
- Standard textbook exercises (even from graduate texts)
- Purely computational problems that Mathematica/SageMath can solve directly
- Problems without clear subquestions for automated evaluation

Problem Templates

Intersection Theory

Main: Let X be [variety]. Compute the class of [specific cycle] in the Chow ring $A^*(X)$.
Subquestions: What is the degree of this class? Does it vanish in $A^2(X)$?

Main: For the moduli space M of [objects], compute a closed formula for the intersection number $\int_M \alpha_1 \cup \alpha_2 \cup \ldots \cup \alpha_n$.
Subquestions: What is this number for specific parameter values?

Classification Problems

Main: Classify all [objects] with [property]. Give explicit representatives for each isomorphism class.
Subquestions: How many classes are there? Which have additional property P?

Main: What is the rank of the cohomology group $H^k(M)$ for [variety/moduli space M]?
Subquestions: What is $\dim H^0(M)$? Is $H^n(M) = 0$ for $n > d$?

Example Problems

Example 1: Stable Graphs

Main question: Find a closed formula for the number $N(g)$ of stable graphs of genus g with no legs and precisely 3 edges, for all $g \geq 2$.

Subquestions:

- What is $N(3)$?
- What is $N(8)$?
- What is $N(1000)$?

Example 2: Permutation Representations

Main question: Let G be a finite group. Is the functor $\mathrm{Perm} : G\text{-sets} \to \mathrm{Rep}_{\mathbb{C}}(G)$ sending X to its permutation representation fully faithful? Prove or provide a counterexample.

Subquestions:

- Is the statement true for all finite groups?
- Is the statement true for all finite cyclic groups?
- Is the statement true for all finite abelian groups?

Brainstorming Tips

→ A tricky calculation from your recent work that required a clever insight
→ An "obvious" statement that actually needs a non-trivial proof
→ A self-contained lemma that came up in a research project
→ An oral exam question for an advanced course

Ready to Contribute?

Start creating your problem using our editor with LaTeX support and AI testing.

Create New Problem

IMProofBench © 2025 | Mathematical Reasoning Benchmark for AI Research | Privacy Policy

Figure 12: Guidelines for authoring benchmark problems.

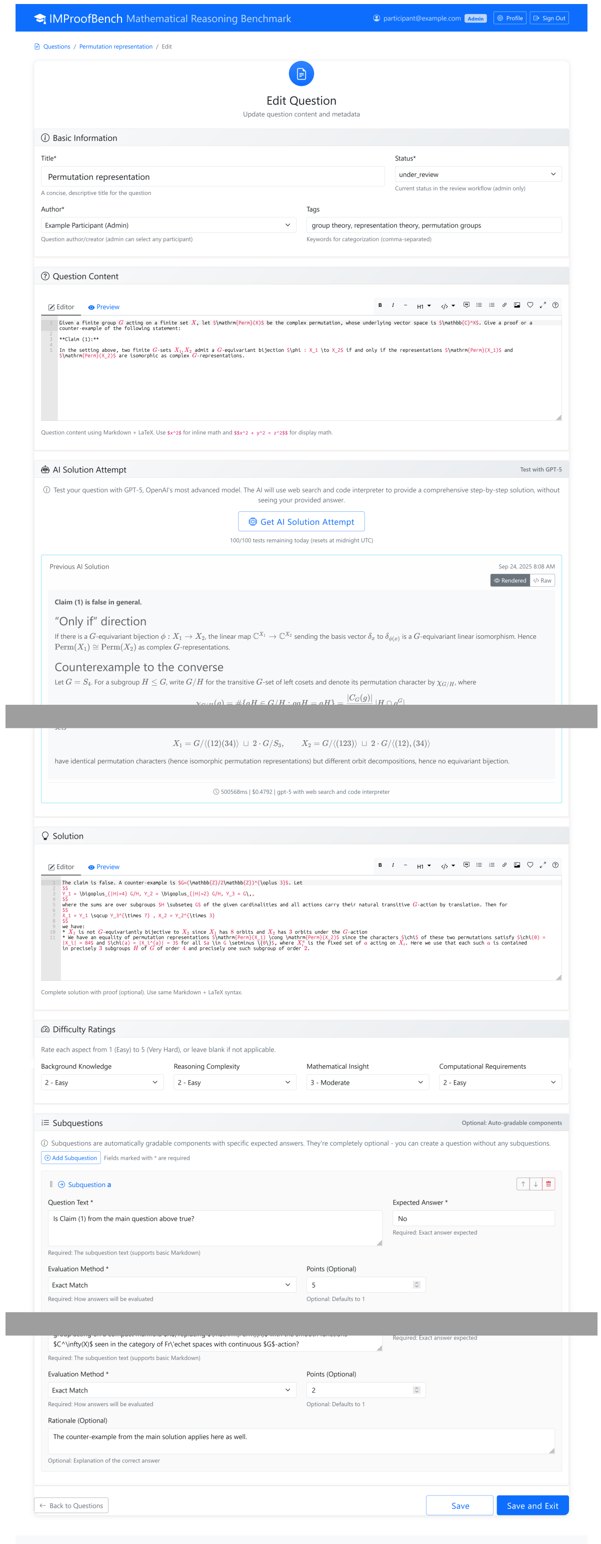

Figure 13: Window for editing questions, solutions, and their associated subquestions; via the blue button, the user can request up to 20 free AI solution previews per day to check suitability of the question.

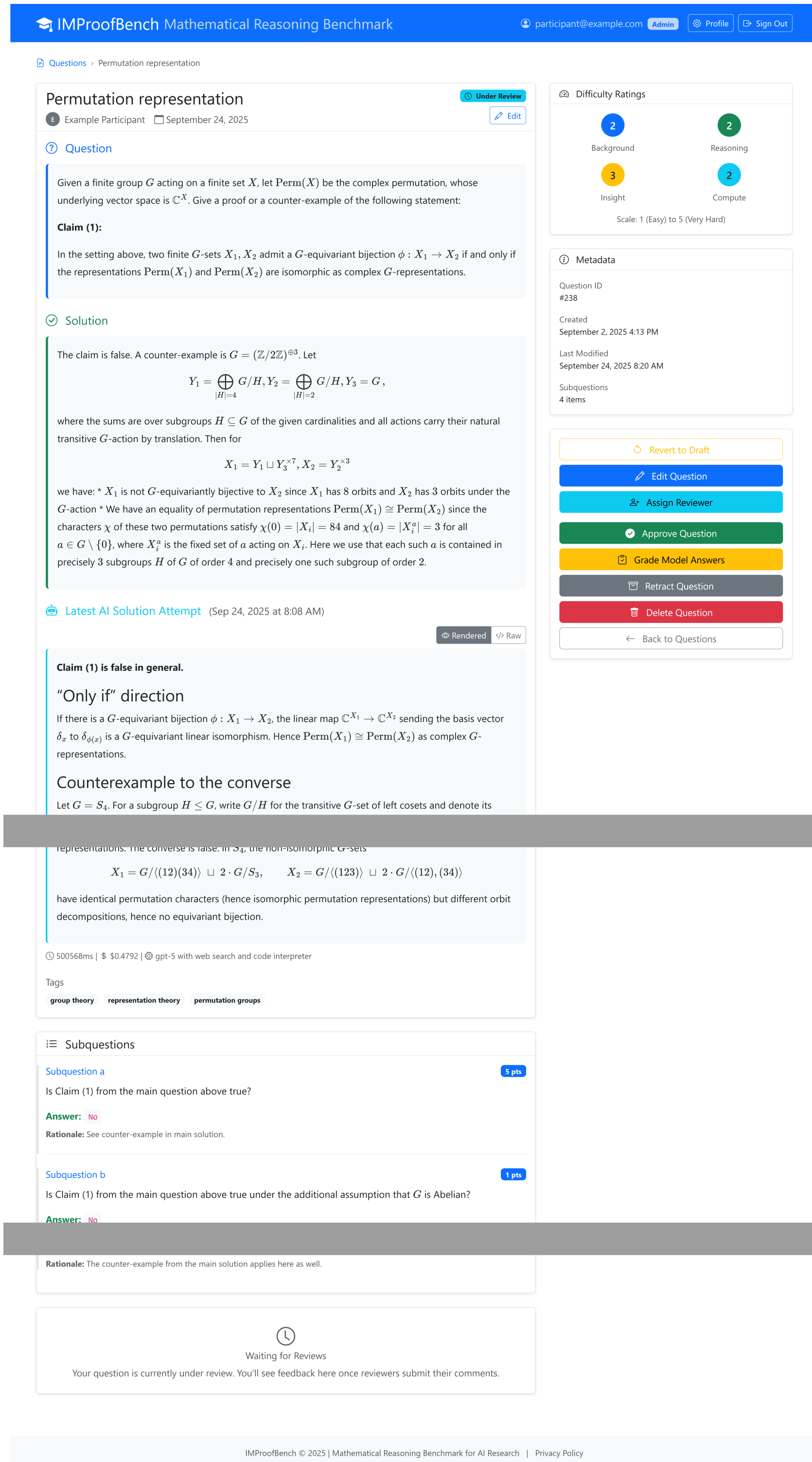

Figure 14: Overview page of question data (with main question, sample solution, AI answer preview and subquestions).

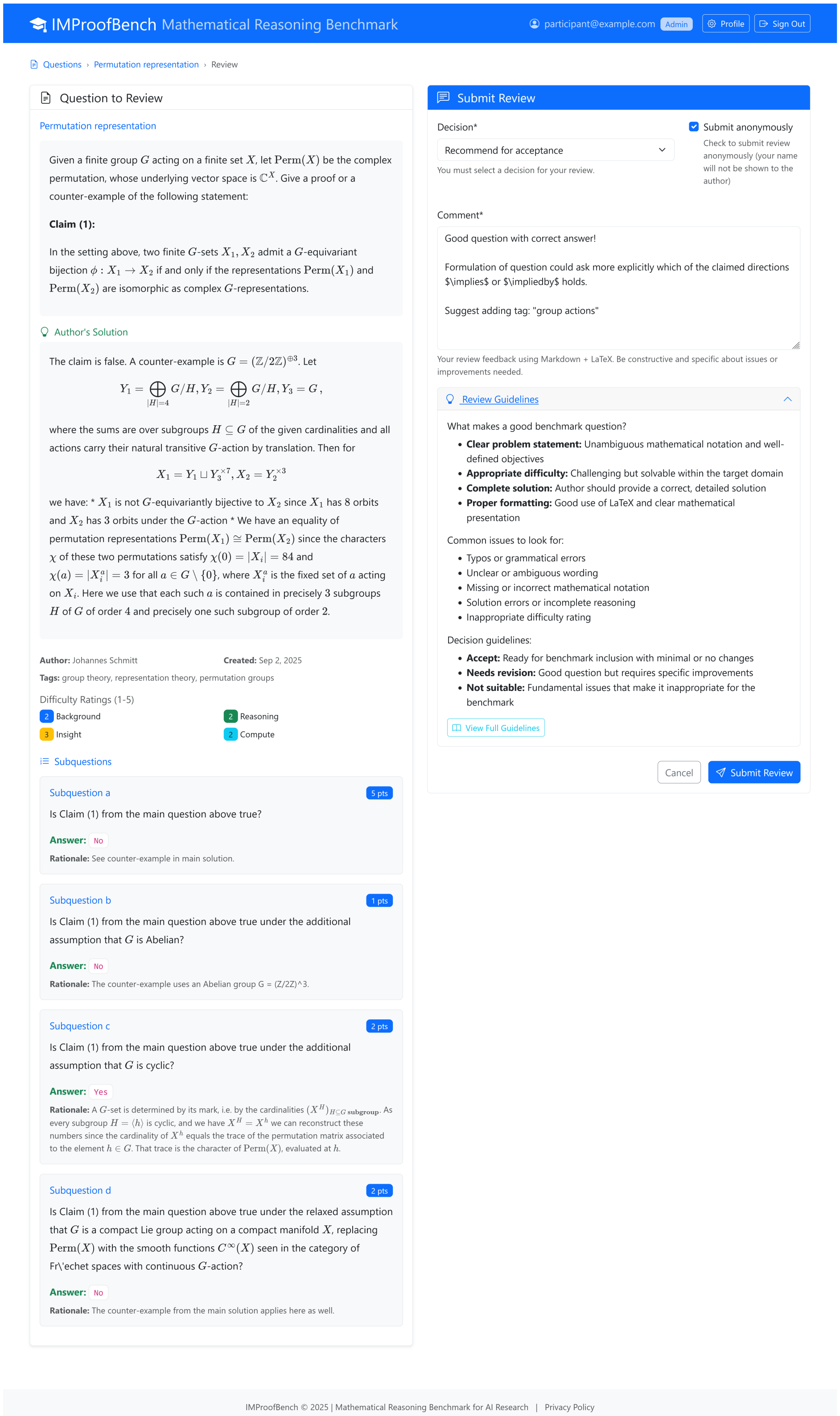

Figure 15: Question review window showing text box for feedback and review instruction summary.

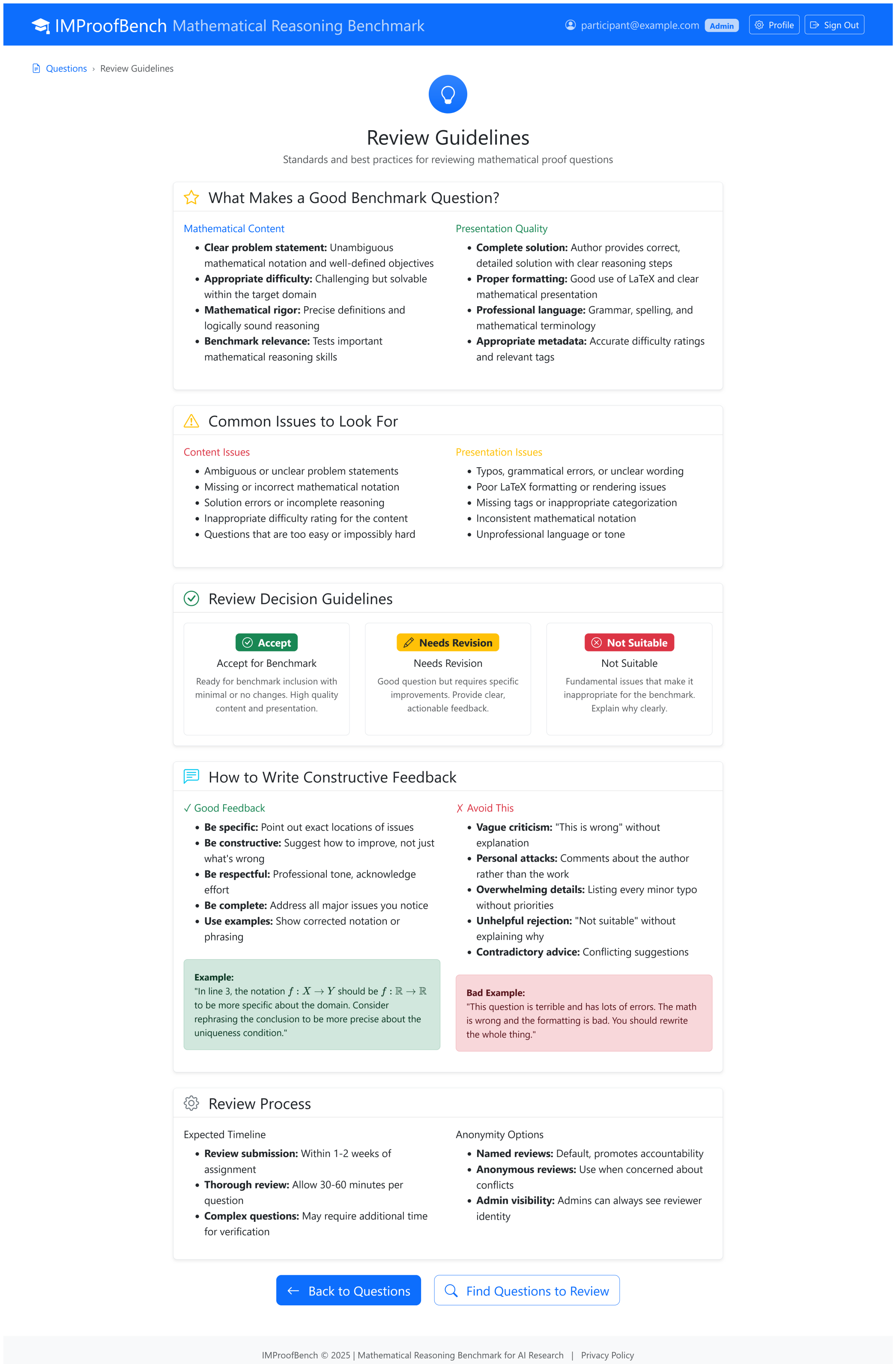

Figure 16: Detailed explainer of review instructions and process.

Figure 17: Grading form, displaying sample solution, model answer and scoring form side by side. Model identities (A–E) at top are randomized on starting the grading, and only revealed when grading is complete, to avoid bias.