The Open Proof Corpus: A Large-Scale Study of LLM-Generated Mathematical Proofs

Authors: Jasper Dekoninck [*1] Ivo Petrov [*2] Kristian Minchev [2] Mislav Balunović [1 2] Martin Vechev [1 2]
Dataset Contributors: Miroslav Marinov [3] Maria Drencheva [2] Lyuba Konova [4] Milen Shumanov [2] Kaloyan Tsvetkov [2] Nikolay Drenchev [2] Lazar Todorov [2] Kalina Nikolova [2 5] Nikolay Georgiev [2] Vanesa Kalinkova [2] Margulan Ismoldayev [2 5]

https://proofcorpus.ai/

https://huggingface.co/datasets/INSAIT-Institute/OPC

Abstract

In recent months, large language models (LLMs) have made significant progress in mathematical proof generation, but further advancement is hindered by the lack of a large-scale, high-quality dataset of human-evaluated proofs. While expensive to create, such a dataset is essential for driving improvements in training and enabling a rigorous analysis of proof generation capabilities. In this work, we present *the Open Proof Corpus* (OPC), a dataset comprising over 5,000 human-evaluated proofs produced by state-of-the-art LLMs. The OPC was specifically designed for broad applicability and downstream usage in proof generation research and is the first to include a substantial number of correct, LLM-generated solutions to problems from prestigious mathematics competitions such as the USAMO and IMO. Using the OPC, we explore critical questions in automated proof generation: (1) the performance gap between natural language and formal proof generation, (2) the discrepancy between final-answer accuracy and full-proof validity, and (3) the impact of best-of-n selection on proof quality. Finally, to showcase the utility of the OPC, we finetune an 8B-parameter model on the dataset, obtaining a model that performs on par with the best model, GEMINI-2.5-PRO, on the task of evaluating proof correctness.

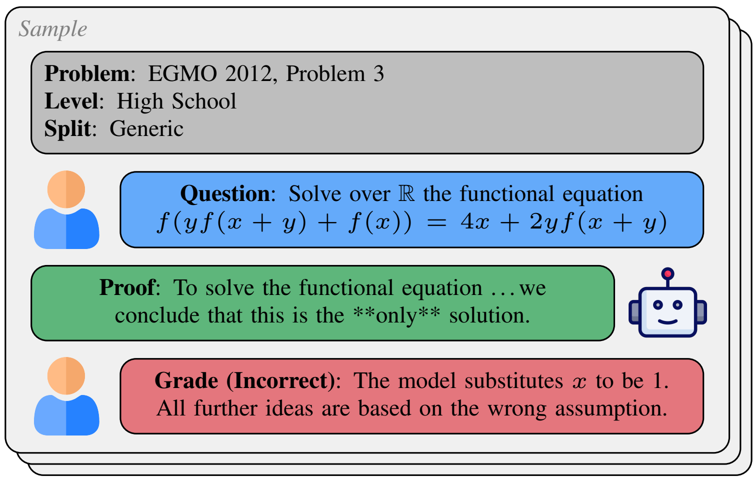
*Equal contribution [1]ETH Zurich [2]INSAIT, Sofia University "St. Kliment Ohridski" [3]Institute of Mathematics and Informatics, Bulgarian Academy of Sciences [4]Sofia University "St. Kliment Ohridski" [5]Massachusetts Institute of Technology. Correspondence to: Ivo Petrov <ivo.petrov@insait.ai>.

1. Introduction

Large language models (LLMs) have recently achieved remarkable progress in mathematical reasoning, attaining top-competitor performance on various final-answer benchmarks such as AIME and HMMT (Balunović et al., 2025). However, growing evidence suggests that these benchmarks fail to capture the full breadth of mathematical reasoning capabilities, as they do not require models to produce complete proofs or detailed intermediate steps (Mahdavi et al., 2025; Guo et al., 2025b). Such step-by-step reasoning is critical for applications in theorem proving, mathematical research, and education.

Proof benchmarking To address this gap, several evaluation efforts have emerged, revealing that LLMs significantly underperform on proof generation compared to existing final-answer benchmarks (Petrov et al., 2025; Mahdavi et al., 2025; Guo et al., 2025b). Despite their value, these benchmarks are severely limited in their use for broader analysis, training, and future development purposes. Specifically, they are small in size (Petrov et al., 2025; Guo et al., 2025b), rely on outputs from outdated models (Frieder et al., 2023), contain few correct proofs (Mahdavi et al., 2025), and are not fully open-sourced (Mahdavi et al., 2025; Guo et al., 2025b).

Open questions Furthermore, key questions about proof generation capabilities remain unanswered. First, while it is widely claimed that final-answer benchmarks are insufficient for evaluating proof generation capabilities, this claim is not yet substantiated by evaluating LLM-generated proofs on such benchmarks. Second, with recent advances in formal proof generation using systems like Lean (Ren et al., 2025; Wang et al., 2025), the performance gap between natural language and formal proof generation remains unclear. Third, the potential of best-of-n sampling strategies to improve proof quality has not been explored.



(a) Sample from the OPC

(b) Conclusions from the OPC

Figure 1: Overview of the OPC and its conclusions. On the left, we show a typical sample from the OPC, including a question taken from a high-quality mathematical competition, a proof generated by an LLM, and a human judgment of the proof's correctness. On the right, we summarize the main conclusions drawn from the OPC.

Our work: the Open Proof Corpus To address these challenges, we introduce the *Open Proof Corpus (OPC)*: a large-scale, human-validated dataset comprising over 5,000 LLM-generated proofs across more than 1,000 problems. As shown in Fig. 1(a), each OPC sample includes (1) a problem from a high-quality mathematical competition such as the International Mathematical Olympiad (IMO), (2) a proof generated by a state-of-the-art LLM, and (3) a binary human evaluation of the proof's correctness with feedback. The OPC was specifically designed for downstream usage in proof generation research, enabling both training and evaluation of LLMs on proof generation tasks. To address the open questions outlined earlier, the OPC includes problems from specific sources: problems from the PutnamBench (Tsoukalas et al., 2024) enable comparisons between formal and informal reasoning, while problems from MathArena (Balunović et al., 2025) support evaluation of proof correctness for problems with final answers.

Key findings Despite the difficulty of problems in the OPC, state-of-the-art LLMs demonstrate surprisingly strong performance. For instance, O4-MINI correctly solves almost 20% of the problems in the IMO Shortlist, and the OPC generally consists of 43% correct proofs. Furthermore, as shown in the top left of Fig. 1(b), LLMs exhibit strong capabilities in evaluating proofs: GEMINI-2.5-PRO achieves 88.1% accuracy in judging proof correctness—nearly matching human performance at 90.4%. To showcase the utility of the OPC, we fine-tune DEEPSEEK-R1-QWEN3-8B (Guo et al., 2025a) using GRPO (Shao et al., 2024) on the OPC, resulting in an open model that achieves 88.1% judgment accuracy, on-par with GEMINI-2.5-PRO and outperforming all other models.

Answering open questions The OPC allows us to empirically resolve the open questions outlined above, with all conclusions shown in Fig. 1(b). First, as shown on the top right, we find that natural language proof generation significantly outperforms formal proof generation: on PutnamBench, GEMINI-2.5-PRO solves 11 times more problems than the best formal model, DEEPSEEK-PROVER-V2 (Ren et al., 2025). Second, as shown on the bottom left, we observe a substantial gap between final-answer accuracy and proof correctness. While GEMINI-2.5-PRO loses only 8% of its final-answer accuracy when proof correctness is required, O3 suffers a drop of almost 30%. Third, as shown on the bottom right, we find that best-of-n strategies yield large gains in performance. Interestingly, while standard best-of-n selection methods moderately improve accuracy from 26% to 36%, a ranking-based approach inspired by (Liu et al., 2025) achieves the highest performance of 47%.

Main Contributions Our key contributions are:

- A scalable pipeline for generating and evaluating natural language mathematical proofs (§3).
- The *Open Proof Corpus*, a large, human-validated dataset of over 5,000 LLM-generated proofs (§4).
- New benchmarking results on the OPC, resulting in novel insights into current LLM capabilities (§5).
- An open-source, 8B-parameter model fine-tuned on the OPC that achieves 88.1% judgment accuracy, tied with the best model on this task (§5).

2. Related Work

We briefly recapitulate the relevant literature on mathematical reasoning benchmarks and datasets.

Final-answer benchmarks Final-answer benchmarks evaluate models by comparing a final answer from the model's output with a ground-truth answer, typically using rule-based parsers. With the advent of reasoning LLMs (Jaech et al., 2024; Guo et al., 2025a), older benchmarks have become saturated (Cobbe et al., 2021; Lightman et al., 2024), and even more recent ones are nearing saturation (Gao et al., 2025; He et al., 2024; Balunović et al., 2025; Gulati et al., 2024). Only private benchmarks such as FrontierMath (Glazer et al., 2024) and HLE (Phan et al., 2025) remain sufficiently challenging for the latest models. However, these benchmarks do not require the generation of full proofs or detailed reasoning steps. Moreover, their private nature hinders reproducibility and broader community engagement, both key factors in driving progress.

Formal proof generation Another growing line of work involves training LLMs to generate formal proofs in languages such as Lean (de Moura and Ullrich, 2021) or Isabelle (Nipkow et al., 2002), which can then be automatically verified by these systems (Zheng et al., 2022; Tsoukalas et al., 2024; Yu et al., 2025). While this paradigm enables rigorous correctness checking, it typically requires models to be specifically finetuned for formal proof generation (Ren et al., 2025; Wang et al., 2025; Lin et al., 2025). In contrast, state-of-the-art general-purpose models like O3 and GEMINI-2.5-PRO struggle with formal proof generation. As a result, formal reasoning currently does not allow to make full use of the natural language capabilities of these general-purpose models, and, as we show in §5, there remains a significant performance gap between formal and natural language proof generation, with the latter being substantially more effective.

Proof-generation evaluation efforts Going beyond final-answer accuracy has gained recent attention, with several works investigating the reasoning traces of recent LLMs to identify patterns and potential for improvement (Shojaee et al., 2025; Mondorf and Plank, 2024; Xia et al., 2025). However, only a few studies have focused directly on evaluating full proofs. Petrov et al. (2025) evaluated LLMs on the six problems from the USAMO 2025, uncovering significant flaws in the generated proofs. Similarly, Mahdavi et al. (2025) evaluated LLM performance on a large set of IMO Shortlist problems, finding that no model surpassed 5% accuracy. In contrast, Frieder et al. (2023) reported that GPT-3.5 and GPT-4 performed well on simpler tasks, generating correct proofs in a significant fraction of cases. Still, all these studies are limited by either the use of outdated models or the small scale of their evaluations. Two concurrent works have recently expanded this line of research. Sheng et al. (2025) focus specifically on inequality proofs and primarily emphasize the development of an LLM as a judge framework to mitigate the high cost of human evaluation. Guo et al. (2025b) highlight a notable gap between final-answer accuracy and the ability to generate correct proofs, a finding we confirm in §5. However, their analysis stops at this observation and does not evaluate performance on an established final-answer benchmark. Further, both studies are limited in scale compared to the OPC and have not open-sourced their human-annotated proofs.

Mathematical training datasets Several large-scale datasets have been developed to train LLMs on mathematical reasoning. One of the earliest, Li et al. (2024), compiled a large dataset of internet-sourced problems, including both final-answer questions and natural language proofs. However, it lacks LLM-generated proofs, human judgments, and examples of incorrect proofs. Other datasets have focused exclusively on final-answer tasks (Albalak et al., 2025; He et al., 2025; Moshkov et al., 2025), offering limited support for training or evaluating proof generation. Finally, Zhang et al. (2025) introduced a dataset of both valid and invalid problem statements, each accompanied by LLM-generated proofs. While this ensures that incorrect proofs exist for invalid statements, the dataset does not include human evaluation of proofs or any other information on the validity of the proofs.

LLM as a judge The widespread success of LLMs has led to the development of the "LLM as a judge" paradigm, which enables scalable, consistent, and cost-effective evaluation of complex outputs (Zheng et al., 2023; Gu et al., 2024). This approach serves as a promising alternative to traditional expert-based evaluation and opens up new possibilities for self-assessment during generation through methods such as best-of-n sampling, feedback-guided decoding, and self-training (Pan et al., 2023; Madaan et al., 2023; Weng et al., 2023; Jiang et al., 2025). However, concerns remain about the robustness of this framework in high-stakes domains like mathematical reasoning, where issues such as overconfidence and self-degradation can undermine reliability (Huang et al., 2024; Pan et al., 2023).

3. Methodology

Accurately evaluating LLM-generated proofs is a challenging task. Models often make difficult-to-detect errors, and they rarely acknowledge when they cannot solve a problem (see §5). In this section, we outline the methodology used to create the OPC, with particular emphasis on the complexities of evaluating LLM-generated proofs and our efforts to maximize dataset size. Since human judges cannot reasonably spend hours studying each problem, we developed a pipeline to support efficient grading. This pipeline consists of three main components: problem and judge preparation (§3.1), the grading procedure (§3.2), and monitoring and validation (§3.3).

3.1. Problem and Judge Preparation

Judge selection Judges were selected from among former IMO participants or individuals who reached the final stages of IMO selection in their respective countries. We contacted each judge personally to ensure they were qualified, motivated, and easily accessible for ongoing communication. A total of 13 judges were involved, each responsible for grading a varying number of problems. Three of the most active judges had prior experience with evaluating LLM-generated proofs. One judge served as the coordinator, maintaining regular communication, tracking progress, and ensuring consistency and motivation across the group.

Problem selection Problems were drawn from top-tier national and international mathematics competitions, with the goal of capturing a balanced mix of correct and incorrect proofs. All problems were sourced from official materials. Non-English problems were translated using GPT-4.1 and manually verified for accuracy. When available, official ground-truth solutions were also extracted and provided to the judges as references.

Proof generation Proofs were generated using a set of state-of-the-art LLMs known for their strength in mathematical reasoning. Specifically, we used O4-MINI and O3 from OpenAI (OpenAI, 2025), GEMINI-2.5-PRO from Google (Google DeepMind, 2025), GROK-3-MINI from xAI (xAI, 2025), QWEN3-235B-A22B from Qwen (Qwen Team, 2025), and the latest version of DEEPSEEK-R1 from DeepSeek (Guo et al., 2025a). DEEPSEEK-R1 was released mid-way through dataset construction and replaced GROK-3-MINI thereafter. We designed the model prompt to clearly instruct models to generate full solutions, refining it through small-scale pilot tests. The final prompt is shown in App. E.1. All models were run with default parameters and a maximum generation length of 64,000 tokens. If this limit was exceeded before a complete proof was generated, we re-sampled. For the MathArena subset (Balunović et al., 2025), we only retained solutions with a correct final answer, retrying generation if necessary. For PutnamBench (Tsoukalas et al., 2024), we appended the informal final answer (if present) to the problem statement to mirror the setup for formal models, allowing direct comparison between natural and formal proof outputs.

3.2. Grading Procedure

User interface We built a custom web interface to facilitate efficient grading. A sample instance is available at `https://judgeproofs.xyz/sample`, with screenshots included in App. D. The interface displays the problem, reference solution (if available), anonymized model-generated proof, and grading form. Judges could mark the problem or solution as malformed (to filter such cases from the dataset), grade the proof, and annotate sentences from the model-generated proof with comments. Continuous feedback from judges helped us refine the interface over time.

Judge instructions Judges were asked to label proofs as either correct or incorrect and provide written justification. Precise grading guidelines were critical to avoid inconsistencies in edge cases, such as minor omissions or shortcuts. To prevent overly strict grading, we clearly defined what level of omissions and frequency of mistakes were acceptable in a correct proof. The full guidelines are available at `https://judgeproofs.xyz/instructions/`, with a summary in App. D. These instructions were developed collaboratively with the judges and finalized after a pilot phase.

Abstention and uncertainty Judges were allowed to abstain from grading a proof if they lacked the necessary expertise or found the solution too complex or convoluted. They could also mark judgments as uncertain in borderline cases, which proved especially useful for near-correct proofs containing subtle errors. Less than 3% of proofs in the OPC are flagged as uncertain.

LLM issue summaries After several hundred graded proofs, we introduced a new feature to support grading: an automatically generated summary of the proof using O4-MINI. These summaries flagged potential issues—such as logical gaps or missing steps—based on a specially designed prompt (see App. E.2). Importantly, the model was instructed not to give a final verdict, only to indicate possible issues for human review. Judges reported that this significantly improved their efficiency and accuracy in detecting errors. To ensure the inclusion of these summaries did not bias our judges, we evaluated the agreement rate between O4-MINI as a judge and human graders before and after their introduction. There was no significant difference in agreement, suggesting no introduction of bias into the grading process. However, in experiments involving best-of-n selection—where the LLM judge acts as a selection mechanism—we omitted these summaries to avoid any form of compounding bias in the evaluation.

Problem distribution Problems were assigned to judges based on their mathematical background and availability. Judges not qualified to evaluate undergraduate-level problems were excluded from grading those problems.

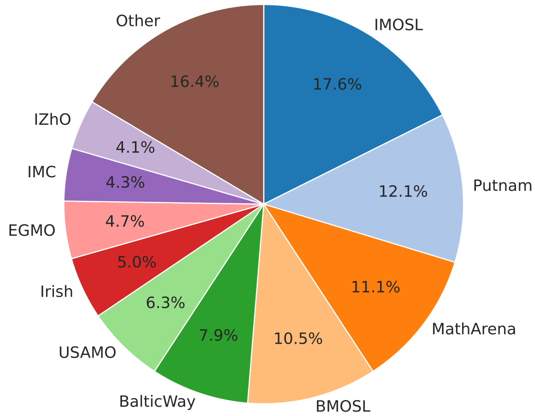


Figure 2: Overview of competitions included in the OPC.

3.3. Monitoring and Validation

To ensure grading quality and judge consistency, we implemented a set of monitoring and validation procedures.

Coordinator One experienced judge was assigned as coordinator, responsible for tracking grading progress, resolving issues, and ensuring motivation. As a senior author, the coordinator had detailed knowledge of the dataset's goals and methodology and was available to answer judges' questions throughout the process.

Pilot phase Prior to full-scale grading, we conducted a test run with a limited number of problems. This helped us identify interface issues, unclear instructions, and other inefficiencies. Based on feedback, we revised the interface and grading guidelines before launching the full evaluation.

Monitoring discrepancies Approximately 10% of the proofs were double-graded to evaluate inter-judge consistency. Disagreements were reviewed by the coordinator to determine whether they arose from misunderstandings, ambiguous instructions, or errors overlooked by a judge. If possible, instructions were further improved to prevent similar issues. However, most inconsistencies stemmed from overlooked errors in the proofs and could therefore not be resolved by clarifying the instructions. If the coordinator identified a significant number of discrepancies for a specific judge, they would discuss the issue with the judge to clarify instructions.

4. Open Proof Corpus

We now introduce the OPC, providing an overview of the competitions and models it includes and key dataset statistics. At a high level, the OPC is built to support the training and evaluation of proof generation models and to facilitate a rigorous analysis of their capabilities.

Table 1: Number of solutions evaluated per model.

Model	# Solutions
O4-MINI	1615
O3	892
QWEN3-235B-A22B	890
GEMINI-PRO	878
GROK-3-MINI	461
DEEPSEEK-R1	326

Basic properties The OPC consists of $5{,}062$ proofs across $1{,}010$ distinct problems, generated by six state-of-the-art LLMs. Each proof is labeled as either correct or incorrect by one or two human judges. Labels are accompanied by short justifications, with optional annotations highlighting specific sentences within the proof. Each problem may also include metadata such as its competition source, difficulty level, and other relevant attributes.

Competitions Problems were sourced from a wide range of prestigious mathematics competitions. A full breakdown is provided in App. A, with a summary shown in Fig. 2. App. A also reports the average accuracy of the best-performing model per competition, offering a rough proxy for difficulty. Most problems are at the high school level, with only a small portion drawn from undergraduate-level competitions. Competitions were selected to cover a broad range of difficulties while ensuring an overall average model accuracy of approximately 50%.

Models Table 1 summarizes the number of proofs generated by each model in the OPC. Not all models contributed to every problem, but most problems include solutions from at least five models. O4-MINI contributed the largest share of proofs, as it was used extensively in best-of-n and pass@n experiments.

Human performance and noise To estimate label reliability, we double-graded approximately 10% of the dataset. Among these, judges agreed on the proof's correctness in 90.4% of cases. Assuming independent judgments, we can estimate the individual judge error rate p by solving $0.904 = (1-p)^2 + p^2$, giving $p = 5\%$. This indicates a relatively low noise level, which is expected for a human-annotated dataset of this complexity.

Data splits The OPC is divided into four subsets, each serving a distinct purpose:

- **MathArena**: A test set of 112 problems from MathArena (Balunović et al., 2025), a final-answer benchmark for mathematical reasoning. 34 questions from the SMT 2025 will only be released after the competition authors have published the questions publicly.

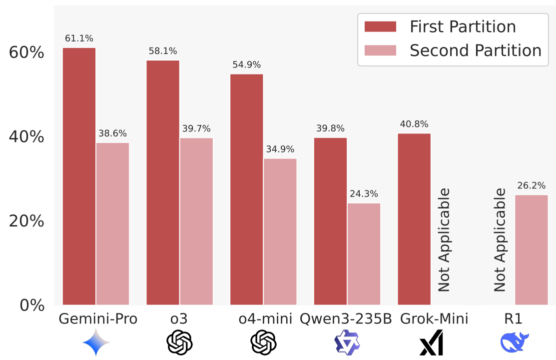


Figure 3: Average proof correctness of various models on the OPC. Data is split into two partitions, the first, resp. second, containing only problems answered by all models except DeepSeek-R1, resp. Grok-3-Mini. The discrepancy between the two partitions is due to the inclusion of harder competitions in the second partition.

- **PutnamBench**: A subset of 114 problems from PutnamBench (Tsoukalas et al., 2024), used to compare natural language and formal proof generation.
- **Best-of-n**: A subset of 152 problems from the IMOSL, USAMO, and BMOSL, each solved multiple times by o4-mini. For 60 of these problems, all 8 generations were human-evaluated. The rest include judgments only for the generation selected by a best-of-n selection strategy.
- **Generic**: A subset of 676 problems from various competitions, including the IMOSL, USAMO, BMOSL, and others, all solved by multiple models.

The MathArena and PutnamBench subsets are drawn from existing public benchmarks and should be treated strictly as test sets. The generic and best-of-n subsets, by contrast, are intended for training, validation, and further analysis. However, a small portion of the generic subset is held back for benchmarking purposes as well.

5. Results

We present the main findings derived from the OPC. In §5.1, we evaluate the proof-generation capabilities of various models. §5.2 evaluates the ability of LLMs to judge the correctness of proofs. In §5.3, we compare informal and formal proof-generation performance. §5.4 analyzes proof correctness given correct final answers. Finally, §5.5 examines the effectiveness of best-of-n selection strategies. In App. C, we additionally provide several qualitative observations about common mistakes in the generated proofs.

Table 2: Benchmarking LLMs as proof graders. Cost for running the model on the entire subset is given in USD.

Judge	pass@1	maj@5	Cost
Human	90.4	-	N/A
Gemini-2.5-Pro	85.4	88.1	135.47
OPC-R1-8B	83.8	88.1	N/A
o4-mini	83.8	85.3	29.57
o3	83.1	84.3	93.30
Gemini-2.5-Flash	82.7	86.0	86.95
Qwen3-235B-A22B	81.8	84.6	3.79
DeepSeek-R1	80.9	82.6	27.35
Qwen3-30B-A3B	74.0	75.4	N/A
DeepSeek-R1-Qwen3-8B	70.7	71.3	N/A
Claude-4-Sonnet	70.6	75.0	28.21
Qwen3-8B	64.4	63.6	N/A
GPT-4.1	61.4	60.8	20.33
Baseline	53.2	53.2	N/A

5.1. Gemini-2.5-Pro Is The Best Proof Generator

Fig. 3 shows the average proof correctness of each model on a filtered subset of the OPC, including only questions answered by all models except Grok-3-Mini or DeepSeek-R1. Gemini-2.5-Pro achieves the highest average accuracy, slightly outperforming o4-mini. In contrast, the two open-source models—Qwen3-235B-A22B and DeepSeek-R1—underperform significantly, highlighting the current performance gap between closed-source and open-source models in proof generation

Out of more than 1,700 incorrect solutions analyzed, models explicitly state their inability to solve the problem in only 114 instances, with all but five of those generated by o3. Even o3 is more likely to produce an incorrect proof than to acknowledge uncertainty. This widespread reluctance to admit failure highlights a key limitation: current LLMs lack effective mechanisms for confidence calibration and knowing their limitations. In domains like mathematics, this could undermine trust in systems that rely on LLMs for provably correct solutions, especially given the difficulty of evaluating proofs correctly.

5.2. LLMs Are Near-Human Level Judges

The OPC is fundamentally a dataset of binary human judgments on proof correctness, making it well-suited for training and evaluating LLMs as proof judges. To leverage this, we split the generic subset by problem into training and test sets. Using GRPO (Shao et al., 2024), we fine-tune DeepSeek-R1-Qwen3-8B on the training set with human labels for reward modeling.

The test set comprises 293 LLM-generated proofs. We evaluate both reasoning models, such as Gemini-2.5-Pro and o4-mini, and general-purpose models like GPT-4.1. As a baseline, we classify proofs under 100 characters as incorrect and guess randomly otherwise.

Table 3: Judgement accuracy breakdown split by solver, highlighting the lowest score for each judge. For visual clarity, we shortened GEMINI-2.5-PRO to GEMINI, O4-MINI to O4, and QWEN3-235B-A22B to QWEN.

Judge \ Prover	GEMINI	O4	O3	QWEN
GEMINI-2.5-PRO	**79.4**	87.1	91.6	80.6
O4-MINI	86.9	**81.3**	83.1	84.1
O3	85.9	84.8	**76.9**	87.8
QWEN3-235B-A22B	80.0	81.9	**79.1**	84.4

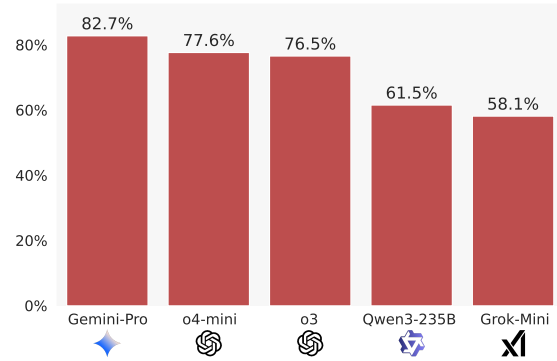


Figure 4: Average proof correctness on the PutnamBench.

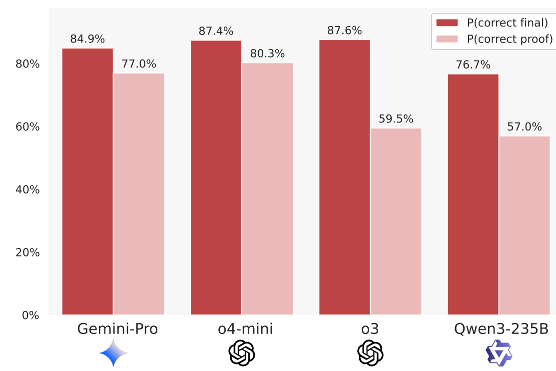


Figure 5: Comparison of final-answer accuracy and proof correctness on the MathArena subset.

As shown in Table 2, GEMINI-2.5-PRO achieves the highest judging accuracy: 85.4% with a single evaluation pass and 88.1% with majority voting, approaching the 90.4% human baseline. Notably, OPC-R1-8B matches GEMINI-2.5-PRO's majority voting performance and outperforms its base model by 17%, demonstrating the value of the OPC and its potential for advancing the field of proof evaluation and generation. However, since the train and test sets share the same distribution, OPC-R1-8B's performance may degrade on out-of-distribution data.

LLMs often favor their own generations (Panickssery et al., 2024). To investigate this, we evaluate how well models judge their proofs compared to those generated by others. In Table 3, we find that all models except QWEN3-235B-A22B perform significantly worse when judging their own proofs, indicating an inability to self-critique effectively. This suggests that LLMs struggle to recognize their mistakes, which is a critical limitation for applications requiring self-assessment or iterative improvement.

This positive result appears to contradict with Petrov et al. (2025), who reported poor performance of judge models. However, their evaluation was based on a limited set of problems, relied on older models, and focused on the more challenging task of scoring proofs on a continuous rather than a binary scale. In addition, we put considerable effort in crafting clear and comprehensible prompt instructions.

5.3. Formal Proof Generation Lags Behind

Using the PutnamBench subset, we compare formal and natural language proof-generation models. The best formal model, DEEPSEEK-PROVER-V2 (Ren et al., 2025), achieves less than 8% accuracy on PutnamBench. In contrast, Fig. 4 shows that the top informal model, GEMINI-2.5-PRO, reaches almost 83% accuracy on the evaluated subset, clearly outperforming all formal models. Despite this disparity, formal proofs offer a major advantage: automatic verifiability. While informal methods currently dominate in performance, formal approaches remain crucial for scalable, rigorous proof checking.

5.4. Proof Generation and Final Answer Do Not Align

Although it is widely claimed that final-answer benchmarks are inadequate for evaluating proof generation capabilities (Petrov et al., 2025; Mahdavi et al., 2025; Guan et al., 2025), it remains unclear how often LLMs can produce a valid proof when they find the correct answer. To investigate this, we first collect instances from the MathArena subset where models generate correct final answers, and then manually evaluate the validity of the accompanying proofs. This enables us to estimate the overall proof correctness rate $P(\text{correct proof})$ and compare it with the final-answer accuracy $P(\text{correct final answer})$.

In Fig. 5, we report each model's accuracy alongside the stricter metric requiring both a correct final answer and a valid proof. Despite GEMINI-2.5-PRO, O4-MINI, and O3 achieving similar final-answer accuracy, their proof correctness rates differ significantly. Specifically, O3 performs notably worse, with only 59.5% of its answers containing a correct proof. This substantial variation across models highlights that final-answer accuracy alone is not a reliable indicator of proof generation capability.

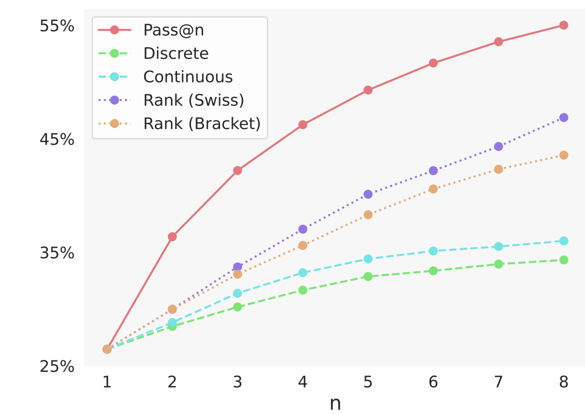


Figure 6: Pass@n metric on a small subset of the OPC compared with some best-of-n selection strategies.

5.5. Best-of-n Significantly Improves Performance

Best-of-n selection—generating multiple outputs and selecting the best one—is a common strategy for improving LLM performance. We evaluate this approach using O4-MINI by generating eight proofs per problem in the best-of-n subset and testing four selection methods:

- **Discrete**: O4-MINI classifies each proof as correct or incorrect and selects any correct proof.
- **Continuous**: O4-MINI scores proofs on a 0-7 scale and selects the one with the highest score.
- **Rank (Bracket)**: A ranking method proposed by Liu et al. (2025). Proofs are judged pairwise by O4-MINI in a knockout tournament until one proof remains. Note that Liu et al. (2025) finetuned their model on the judging task, while we use O4-MINI itself.
- **Rank (Swiss)**: Inspired by Liu et al. (2025), proofs are ranked via pairwise comparisons in a Swiss round-robin tournament. Ratings are computed using the Bradley-Terry model (Bradley and Terry, 1952), and the answer with the highest rating is selected. See App. B for details of this method.

Prompts for all methods can be found in App. E. Note that *Rank (Bracket)* only requires $O(n)$ comparisons, and is therefore as efficient as the discrete and continuous methods, while *Rank (Swiss)* requires $O(n^2)$ comparisons, making it more expensive.

In Fig. 6, we compare the performance of these methods with the pass@n metric on the 60 problems that contain human judgments for all eight proofs. We find that best-of-n selection strategies can significantly improve proof generation performance. Furthermore, the pairwise ranking methods significantly outperform the discrete and continuous methods by approximately 10%. Notably, while discrete and continuous methods plateau after $n = 5$, ranking approaches continue to scale. Finally, *Rank (Swiss)* slightly outperforms *Rank (Bracket)* by a 3% margin.

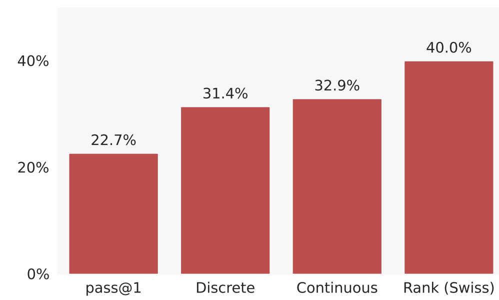


Figure 7: Performance of the best-of-n selection strategies.

In Fig. 7, we evaluate the performance of the best-of-n selection strategies 134 of the best-of-n subset of the OPC[1], except for the *Rank (Bracket)* method, which was not evaluated on the full subset. The improved performance of the ranking methods is confirmed on the entire best-of-n subset, with *Rank (Swiss)* improving accuracy by 17%.

6. Limitations

While the OPC represents a significant advancement in the development and evaluation of LLM proof-generation capabilities, it is not without limitations. First, since most problems in the OPC are sourced from publicly available mathematical competitions, there is a risk of data contamination. Some of these problems may have been encountered during the training of the models. However, since the main purpose of the OPC is not related to benchmarking but rather to provide a large, human-validated dataset of LLM-generated proofs, this issue is less critical. Second, the majority of problems in the OPC are derived from high school level competitions. As a result, the dataset does not cover more advanced mathematical domains, such as undergraduate or research-level mathematics.

7. Future Work

The OPC offers a valuable resource for advancing research in proof generation with LLMs, supporting both the training and evaluation of models for this task. Our analysis also highlights several critical gaps and challenges that warrant further research. First, the significant disparity between formal and natural language proof generation—outlined in §5.3—points to the need for more effective strategies to bridge this divide. Second, the OPC demonstrates that current benchmarks, which often rely solely on final answers, fail to capture the full complexity and quality of generated proofs, as discussed in §5.4. This underscores the need for the development of a scalable benchmarking pipeline tailored to proof generation tasks. Finally, while our results show that best-of-n sampling strategies can meaningfully improve proof quality (see §5.5), further research is required to better understand and optimize these methods.

[1] A small bug in the *Rank (Swiss)* method caused incorrect selections for 18 questions. These are excluded from the analysis.

8. Conclusion

In this work, we introduced the *Open Proof Corpus (OPC)*, a large-scale, human-validated dataset of over $5{,}000$ LLM-generated proofs across more than $1{,}000$ problems. Designed to support training, evaluation, and benchmarking, the OPC provides a robust foundation for advancing research in automated proof generation. Using the OPC, we conducted a thorough evaluation of current LLM capabilities in mathematical reasoning. Our analysis reveals several key findings: (1) modern LLMs can generate a significant number of correct proofs; (2) some models approach human-level performance in judging proof correctness; (3) natural language proof generation substantially outperforms formal proof generation; (4) final-answer benchmarks are a poor proxy for proof quality; and (5) best-of-n sampling strategies significantly improve proof-generation accuracy. Using the OPC, we also trained an open-source model that achieves 88.1% accuracy in judging proof correctness, matching the performance of the best closed-source model, GEMINI-2.5-PRO.

Acknowledgements

This work has received funding from the Swiss National Science Foundation (SNSF) [200021_207967], the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure), and the Swiss State Secretariat for Education, Research and Innovation (SERI).

References

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. URL https://arxiv.org/abs/2502.17387.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: The method of paired comparisons. *Biometrika*, 39(3-4):324–345, 12 1952. ISSN 0006-3444. doi: 10.1093/biomet/39.3-4.324. URL https://doi.org/10.1093/biomet/39.3-4.324.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *CADE*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer, 2021.

Jasper Dekoninck, Maximilian Baader, and Martin T. Vechev. Polyrating: A cost-effective and bias-aware rating system for LLM evaluation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=URPwT55i6O.

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.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=yaqPf0KAlN.

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. *CoRR*, abs/2411.04872, 2024. doi: 10.48550/ARXIV.2411.04872. URL https://doi.org/10.48550/arXiv.2411.04872.

Google DeepMind. Gemini Pro, March 2025. URL https://deepmind.google/technologies/gemini/pro/. Accessed: 2025-04-03.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. A survey on llm-as-a-judge. *CoRR*, abs/2411.15594, 2024. doi: 10.48550/ARXIV.2411.15594. URL https://doi.org/10.48550/arXiv.2411.15594.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *CoRR*, abs/2501.04519, 2025. doi: 10.48550/ARXIV.2501.04519. URL https://doi.org/10.48550/arXiv.2501.04519.

Aryan Gulati, Brando Miranda, Eric Chen, Emily Xia, Kai Fronsdal, Bruno de Moraes Dumont, and Sanmi Koyejo. Putnam-axiom: A functional and static benchmark for measuring higher level mathematical reasoning. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Jiaxing Guo, Wenjie Yang, Shengzhong Zhang, Tongshan Xu, Lun Du, Da Zheng, and Zengfeng Huang. Right is not enough: The pitfalls of outcome supervision in training llms for math reasoning, 2025b. URL https://arxiv.org/abs/2506.06877.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3828–3850. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.211. URL https://doi.org/10.18653/v1/2024.acl-long.211.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *CoRR*, abs/2504.11456, 2025. doi: 10.48550/ARXIV.2504.11456. URL https://doi.org/10.48550/arXiv.2504.11456.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, and Ilge Akkaya. Openai o1 system card. *CoRR*, abs/2412.16720, 2024. doi: 10.48550/ARXIV.2412.16720. URL https://doi.org/10.48550/arXiv.2412.16720.

Chunyang Jiang, Chi-Min Chan, Wei Xue, Qifeng Liu, and Yike Guo. Importance weighting can help large language models self-improve. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 24257–24265. AAAI Press, 2025. doi: 10.1609/AAAI.V39I23.34602. URL https://doi.org/10.1609/aaai.v39i23.34602.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-1.5](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-prover: A frontier model for open-source automated theorem proving. *CoRR*, abs/2502.07640, 2025. doi: 10.48550/ARXIV.2502.07640. URL https://doi.org/10.48550/arXiv.2502.07640.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. Pairjudge rm: Perform best-of-n sampling with knockout tournament, 2025. URL https://arxiv.org/abs/2501.13007.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. 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/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Hamed Mahdavi, Alireza Hashemi, Majid Daliri, Pegah Mohammadipour, Alireza Farhadi, Samira Malek, Yekta Yazdanifard, Amir Khasahmadi, and Vasant G. Honavar. Brains vs. bytes: Evaluating LLM proficiency in olympiad mathematics. *CoRR*, abs/2504.01995, 2025. doi: 10.48550/ARXIV.2504.01995. URL https://doi.org/10.48550/arXiv.2504.01995.

Philipp Mondorf and Barbara Plank. Beyond accuracy: Evaluating the reasoning behavior of large language models - A survey. *CoRR*, abs/2404.01869, 2024. doi: 10.48550/ARXIV.2404.01869. URL https://doi.org/10.48550/arXiv.2404.01869.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. AIMO-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *CoRR*, abs/2504.16891, 2025. doi: 10.48550/ARXIV.2504.16891. URL https://doi.org/10.48550/arXiv.2504.16891.

Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283. Springer Science & Business Media, 2002.

OpenAI. Openai o3 and o4-mini system card, April 2025. URL https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *CoRR*, abs/2308.03188, 2023. doi: 10.48550/ARXIV.2308.03188. URL https://doi.org/10.48550/arXiv.2308.03188.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/7f1f0218e45f5414c79c0679633e47bc-Abstract-Conference.html.

Ivo Petrov, Jasper Dekoninck, Lyuben Baltadzhiev, Maria Drencheva, Kristian Minchev, Mislav Balunovic, Nikola Jovanovic, and Martin T. Vechev. Proof or bluff? evaluating llms on 2025 USA math olympiad. *CoRR*, abs/2503.21934, 2025. doi: 10.48550/ARXIV.2503.21934. URL https://doi.org/10.48550/arXiv.2503.21934.

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, Mantas Mazeika, Daron Anderson, Tung Nguyen, Mobeen Mahmood, Fiona Feng, Steven Y. Feng, Haoran Zhao, Michael Yu, Varun Gangal, Chelsea Zou, Zihan Wang, Jessica P. Wang, Pawan Kumar, Oleksandr Pokutnyi, Robert Gerbicz, Serguei Popov, John-Clark Levin, Mstyslav Kazakov, Johannes Schmitt, Geoff Galgon, Alvaro Sanchez, Yongki Lee, Will Yeadon, Scott Sauers, Marc Roth, Chidozie Agu, Søren Riis, Fabian Giska, Saiteja Utpala, Zachary Giboney, Gashaw M. Goshu, Joan of Arc Xavier, Sarah-Jane Crowson, Mohinder Maheshbhai Naiya, Noah Burns, Lennart Finke, Zerui Cheng, Hyunwoo Park, Francesco Fournier-Facio, John Wydallis, Mark Nandor, Ankit Singh, Tim Gehrunger, Jiaqi Cai, Ben McCarty, Darling Duclosel, Jungbae Nam, Jennifer Zampese, Ryan G. Hoerr, Aras Bacho, Gautier Abou Loume, Abdallah Galal, Hangrui Cao, Alexis C. Garretson, Damien Sileo, Qiuyu Ren, Doru Cojoc, Pavel Arkhipov, Usman Qazi, Lianghui Li, Sumeet Motwani, Christian Schröder de Witt, Edwin Taylor, Johannes Veith, Eric Singer, Taylor D. Hartman, Paolo Rissone, Jaehyeok Jin, Jack Wei Lun Shi, Chris G. Willcocks, Joshua Robinson, Aleksandar Mikov, Ameya Prabhu, Longke Tang, Xavier Alapont, Justine Leon Uro, Kevin Zhou, Emily de Oliveira Santos, Andrey Pupasov Maksimov, Edward Vendrow, Kengo Zenitani, Julien Guillod, Yuqi Li, Joshua Vendrow, Vladyslav Kuchkin, and Ng Ze-An. Humanity's last exam. *CoRR*, abs/2501.14249, 2025. doi: 10.48550/ARXIV.2501.14249. URL https://doi.org/10.48550/arXiv.2501.14249.

Qwen Team. Qwen3: Think Deeper, Act Faster, April

2025. URL https://qwenlm.github.io/blog/qwen3/. Qwen Blog.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *CoRR*, abs/2504.21801, 2025. doi: 10.48550/ARXIV.2504.21801. URL https://doi.org/10.48550/arXiv.2504.21801.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Jiayi Sheng, Luna Lyu, Jikai Jin, Tony Xia, Alex Gu, James Zou, and Pan Lu. Solving inequality proofs with large language models, 2025. URL https://arxiv.org/abs/2506.07927.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity, 2025. URL https://arxiv.org/abs/2506.06941.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/1582eaf9e0cf349e1e5a6ee453100aa1-Abstract-Datasets_and_Benchmarks_Track.html.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *CoRR*, abs/2504.11354, 2025. doi: 10.48550/ARXIV.2504.11354. URL https://doi.org/10.48550/arXiv.2504.11354.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 2550–2575. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.167. URL https://doi.org/10.18653/v1/2023.findings-emnlp.167.

xAI. Grok 3 beta — the age of reasoning agents, February 2025. URL https://x.ai/news/grok-3. Accessed: 2025-04-03.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. Evaluating mathematical reasoning beyond accuracy. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 27723–27730. AAAI Press, 2025. doi: 10.1609/AAAI.V39I26.34987. URL https://doi.org/10.1609/aaai.v39i26.34987.

Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, Yandong Wen, Ge Zhang, and Weiyang Liu. Formalmath: Benchmarking formal mathematical reasoning of large language models, 2025. URL https://arxiv.org/abs/2505.02735.

Ziyin Zhang, Jiahao Xu, Zhiwei He, Tian Liang, Qiuzhi Liu, Yansi Li, Linfeng Song, Zhenwen Liang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Deeptheorem: Advancing llm reasoning for theorem proving through natural language and reinforcement learning, 2025. URL https://arxiv.org/abs/2505.23754.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. 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/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html`.

Table 4: A list of competition sources for the problems in OPC.

Competition	Description	Problems	Solutions	Level	Type	Acc	Source
Main Analysis							
Balkan MO Shortlist	Competition between Balkan countries	74	368	HS	I	31.7%	Public
Baltic Way MO	Northern and Central European Olympiad	80	395	HS	I	68.1%	Public
British MO Final	Final round of the British Olympiad	23	114	HS	N	78.3%	Public
British MO Prelim	Preliminary round for the British Olympiad	28	139	HS	N	87.5%	Public
Bulgarian Seasonal Competitions	Seasonal Competitions hosted in Bulgaria (8th-12th grade)	49	242	HS	N	63.5%	Public
European Girls' MO	Europe-wide olympiad allowing only girls as participants	47	227	HS	I	37.8%	Public
IMC	International competition for university students	43	212	UG	I	62.2%	Public
IMO Shortlist	Shortlist of problems, from which the IMO is selected	128	629	HS	I	18.0%	Public
International Zhautykov MO	Kazakhstan-based olympiad with near-IMO-level questions	41	203	HS	I	34.2%	Public
Irish MO	Final round of the Irish Olympiad	51	255	HS	N	92.0%	Public
Putnam	Undergraduate competition, regarded as one of the most difficult	8	35	UG	I	100.0%	Public
Romanian Masters of Mathematics Extralist	IMO-level competition hosted in Romania	33	162	HS	I	43.3%	Private
Swiss MO	Various problems from the Swiss Olympiad	8	39	HS	N	87.5%	Public
USA Junior MO	USA olympiad that allows only junior students	25	121	HS	N	52.2%	Public
USAMO	The final round of the USA Math Olympiad	38	190	HS	N	35.1%	Public
PutnamBench							
Putnam	Undergraduate competition, regarded as one of the most difficult	114	564	UG	I	82.7%	Public
MathArena							
AIME 2025	Answer-based competition, serving as a qualifier for the USAMO	24	93	HS	N	95.7%	Public
BRUMO 2025	Answer-based competition hosted by Brown University	28	114	HS	N	100.0%	Public
HMMT February 2025	Answer-based competition hosted by Harvard and MIT	26	103	HS	N	97.8%	Public
SMT 2025	Answer-based competition hosted by Stanford	34	128	HS	N	92.6%	Private
Best-of-n							
Balkan MO Shortlist	Competition between Balkan countries	45	287	HS	I	62.5%	Public
IMO Shortlist	Shortlist of problems, from which the IMO is selected	57	269	HS	I	30.8%	Public
USAMO	The final round of the USA Math Olympiad	40	173	HS	N	66.7%	Public

A. Competitions in the OPC

The OPC contains over 1000 problems that were sourced from national and international competitions of varying difficulty. In Table 4, we present the problem and sample distribution for each. We also include the following additional information:

- **Level**: the education level the problems are appropriate for, either high school (HS) or undergraduate (UG).
- **Type**: whether the competition is hosted internationally (I) or only nationally (N).
- **Source**: we link the source, from which we obtained the problems. Any source that is not publicly available was marked as "Private".
- **Acc**: the average accuracy of the best-performing model on the competition problems, which serves as a rough proxy for difficulty.

B. Swiss Ranking Methodology

We briefly describe the Swiss ranking method used as a best-of-n selection strategy. In this approach, a round-robin tournament is performed where each proof competes against every other. In each "game", two proofs are compared by O4-MINI, which decides which proof is better, or if they are equally good.

To determine the overall winner, we compute a rating for each proof using the Bradley-Terry model (Bradley and Terry, 1952), a probabilistic model for paired comparisons commonly applied in LLM evaluation (Zheng et al., 2023; Dekoninck et al., 2025). The Bradley-Terry model estimates the probability that a proof with rating r_i beats a proof with rating r_j as:

$$P(i \text{ beats } j) = \frac{1}{1 + \exp(r_j - r_i)}.$$

We fit the model to the outcomes of the round-robin tournament using maximum likelihood estimation, resulting in a rating for each proof. The proof with the highest rating is selected as the best.

C. Qualitative Observations

Building on our quantitative analysis, we present several qualitative insights drawn from our judges' evaluations. These include common failure modes and notable model behaviors.

Incorrect handling of inequalities Many models sometimes make mistakes when manipulating inequalities. A frequent issue is the improper reversal of inequality signs during intermediate steps without mathematical justification. This results in incorrect final answers despite seemingly coherent reasoning. Below is an example of such a mistake made by DEEPSEEK-R1, where the incorrect sign reversal is marked with "**".

DEEPSEEK-R1 – Baltic Way 2004 P3

...

Summing these inequalities gives

$\sum_{\text{cyc}} \frac{a+b}{a+b+1} \geq \sum_{\text{cyc}} \left(1 - \frac{1}{3(ab)^{1/3}}\right) = 3 - \frac{1}{3}\sum_{\text{cyc}}(ab)^{-1/3}.$

...

By AM-GM again,

$a^{1/3} + b^{1/3} + c^{1/3} \geq 3\sqrt[3]{a^{1/3}b^{1/3}c^{1/3}} = 3(abc)^{1/9} = 3 \cdot 1^{1/9} = 3,$

with equality if and only if $a^{1/3} = b^{1/3} = c^{1/3}$, i.e., $a = b = c$. Thus,

$\sum_{\text{cyc}} \frac{a+b}{a+b+1} ** \geq ** 3 - \frac{1}{3} \cdot 3 = 3 - 1 = 2.$

...

Struggles with synthetic geometry reasoning In Olympiad-level geometry problems, models often default to coordinate-based methods, i.e., using computations with complex numbers or a specific coordinate system to "bash" the problem, especially on easier problems where such techniques are tractable. However, for harder problems, models tend to shift toward synthetic approaches, relying on geometric theorems and properties. These harder problems typically require keeping track of multiple different but related angles and a robust understanding of geometric relationships. Here, models frequently struggle: they sometimes confuse angle orientations, misidentify congruent angles, reverse point orderings, or apply geometric properties incorrectly. In the following example, GEMINI-2.5-PRO correctly identifies the base of an isosceles triangle in one case, but subsequently deduces the wrong pair of equal angles—illustrating a common angle-reasoning failure.

GEMINI-2.5-PRO – IMOSL 2013 G4

...

Thus, $\angle PBA = \angle QBA$ where these angles are $\angle(BP, BA)$ and $\angle(AB, BQ)$. The problem states these are equal to γ.

So, $\angle(BP, BA) = \gamma$ and $\angle(AB, BQ) = \gamma$. This means
$\angle PBQ = \angle(BP, BA) + \angle(AB, BQ) = \gamma + \gamma = 2\gamma$.

We are given that D is an interior point of segment BQ and $PD = PB$.

Thus, $\triangle PBD$ is an isosceles triangle with base BD.

The angle $\angle PBD$ in $\triangle PBD$ is $\angle PBQ$, so $\angle PBD = 2\gamma$.

The base angles are
$\angle BPD = \angle BDP = \frac{180^\circ - 2\gamma}{2} = 90^\circ - \gamma$.
So, $\angle PDB = 90^\circ - \gamma$. Since D is on segment BQ, this is $\angle(DP, DB) = 90^\circ - \gamma$.

...

Incorrect citation of IMOSL 2004 G8 by GEMINI-2.5-PRO A curious behavior observed exclusively in GEMINI-PRO is the repeated and inappropriate citation of the 2004 IMO Shortlist geometry problem G8. This reference appears in over 15 different solutions, often in problems unrelated to geometry. Notably, these citations consistently occur at points where the model appears to get stuck. No other model in our evaluation displayed this behavior. An illustrative example is shown below.

EGMO 2018 P3

The n contestants of EGMO are named $C_1, C_2, \ldots, C_n$. After the competition, they queue in front of the restaurant according to the following rules.

- The Jury chooses the initial order of the contestants in the queue.
- Every minute, the Jury chooses an integer i with $1 \leq i \leq n$.
- If contestant C_i has at least i other contestants in front of her, she pays one euro to the Jury and moves forward in the queue by exactly i positions.
- If contestant C_i has fewer than i other contestants in front of her, the restaurant opens and the process ends.

For every n, prove that this process must terminate and determine the maximum number of euros that the Jury can collect by cunningly choosing the initial order and the sequence of moves.

Gemini-2.5-Pro

The problem asks us to prove that the process always terminates and to find the maximum number of euros the Jury can collect. One euro is collected for each move a contestant makes.

...

So we have two relations:

$k \cdot m_k = j_k$ for each $k = 1, \ldots, n$ (positional balance for each C_k).

$\sum_{k=1}^n k \cdot m_k = \sum_{k=1}^n j_k$ (total jump events balance).

Substitute (1) into (2):

$\sum_{k=1}^n k \cdot m_k = \sum_{k=1}^n (k \cdot m_k)$.

This equation is an identity and does not help by itself.

Let us use the argument from the ISL 2004 G8 solution notes:

The sum of positions is $P = \sum_{k=1}^n p_k = n(n+1)/2$, which is constant.

The sum of costs, $C = \sum_{k=1}^n (\text{position of } C_k) \times k$.

...

Overgeneralization from simple cases Consistent with prior work (Petrov et al., 2025; Mahdavi et al., 2025), we observe that models often overgeneralize patterns from small or simple examples. This tendency leads to incorrect solutions, particularly when the identified properties do not extend to the general case. An example illustrating this pattern overreach is included below.

Grok 3 Mini – IMO SL 2000 G4

Let $A_1A_2A_3$ be an acute-angled triangle. Denote the foot of the altitude from vertex A_i to the side opposite A_i as K_i, and the point where the incircle touches the side opposite A_i as L_i. The lines K_1K_2, K_2K_3, and K_3K_1 are the sides of the orthic triangle, and the lines L_1L_2, L_2L_3, and L_3L_1 are the sides of the contact triangle.

...

In the special case of an equilateral triangle, the orthic triangle and contact triangle coincide, and the reflections result in the same lines, with vertices on the incircle. For a general acute-angled triangle, coordinate geometry calculations in specific examples (e.g., a triangle with vertices at $B(0,0)$, $C(4,0)$, $A(1,2)$) confirm that the intersection points lie on the incircle, with numerical discrepancies attributable to rounding errors.

Given the symmetry of the configuration and the properties of the incircle as the unique circle tangent to the sides, the intersection points P, Q, and R lie on the incircle for any acute-angled triangle. Thus, the triangle formed by M_{12}, M_{23}, and M_{31} has vertices on the incircle.

D. Grading Interface and Instructions

This appendix outlines the grading interface and the accompanying instructions provided to judges. The full interface and documentation can be accessed online at `https://judgeproofs.xyz/sample` and `https://judgeproofs.xyz/instructions`.

Judge ID Each judge received a unique identifier, which served as their login credential on `https://judgeproofs.xyz`. This ID was used to track grading progress while maintaining judge anonymity in the resulting dataset. To facilitate discussion and resolve ambiguities, a shared communication channel was created between all judges.

Grading interface The grading interface was designed for clarity and ease of use. Figs. 8–10 illustrate its main components. The left panel contains a navigation bar for switching between problems and competitions. The right panel displays the problem statement and the ground-truth solution, along with options for flagging issues in either. Below, the generated solution is shown, accompanied by an automated summary and potential issues identified by an LLM judge. Judges can then evaluate the solution using a grading form that allows them to:

- Indicate whether the solution is correct or incorrect
- Provide a brief justification
- Highlight specific parts of the solution relevant to their decision
- Indicate uncertainty or abstain from grading

Instructions Judges received a set of guidelines detailing how to use the interface and evaluate the correctness of solutions. Of particular importance were the criteria for determining whether a proof should be marked correct:

Instructions for judges on when a proof is correct

```
A solution should be considered correct even if it would earn 5+/7 points in a full grading. Examples of small penalties worth 1 point
 are if the solution:
- Makes a small computational mistake that can be easily fixed
- Misses an edge case which can be easily proven/disproven
- Skips over a step that follows without much reasoning or manual work

A solution should be marked as incorrect if:
- It marks a step as trivial, if it is not immediately obvious why this would be the case
- It omits algebra-heavy computational steps, regardless of whether or not it has outlined the methodology
- Generalizes over a pattern without rigorously describing the pattern, or without proving any relevant properties.
- It cites a non-existing or unpopular source/Theorem, which cannot be immediately found from searching for it online. Thus, any
theorems that can be immediately found and have a Wikipedia article are allowed.

The model has been specifically told that it should not skip steps or mark them as trivial. Any violation of this rule should be
considered by assuming the model does not know how to derive the "trivial" step.
```

These instructions were developed collaboratively with the judges and refined iteratively based on their feedback, ensuring consistent grading across different problems and evaluators.

Figure 8: Main grading interface. The left panel provides navigation across problems and competitions. The right panel displays the problem and ground-truth solution, with options to report issues.

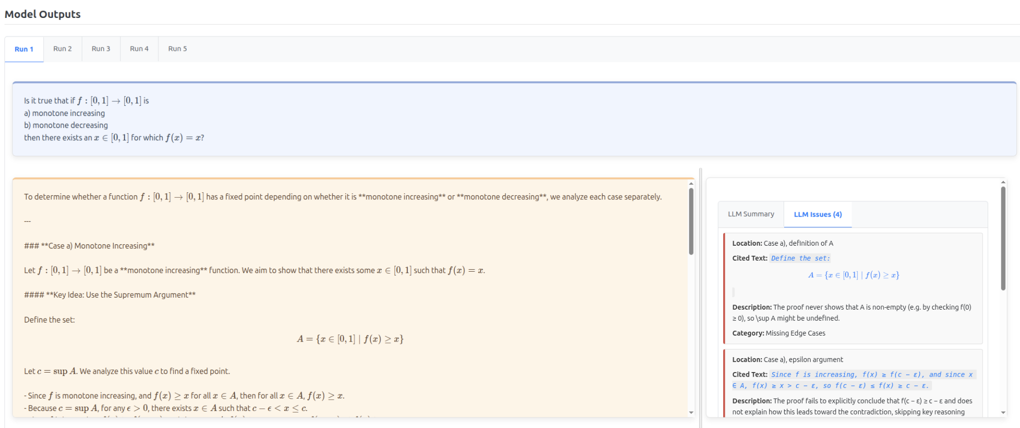


Figure 9: Interface for evaluating a generated solution. The problem is repeated for reference. The generated solution appears on the left, and the LLM's summary and identified issues on the right.

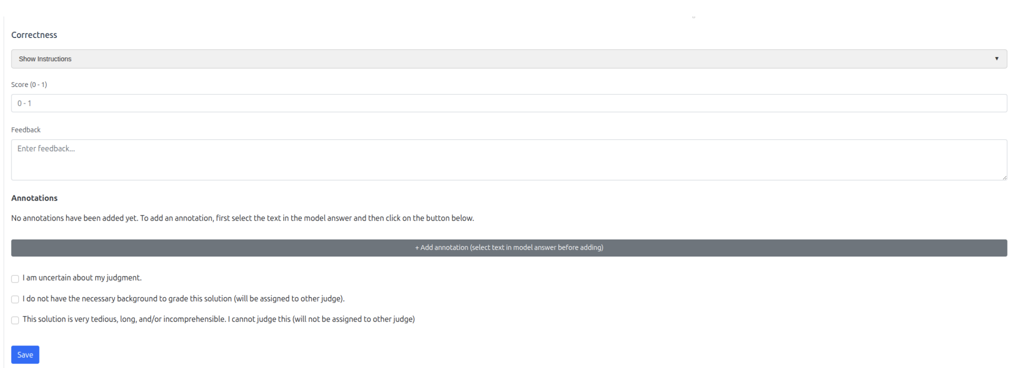


Figure 10: Grading form. Judges indicate correctness, provide a justification, highlight relevant content, and optionally express uncertainty or abstain.

E. Prompts

In this section, we provide the prompts used for various tasks in the OPC. The prompts are designed to be clear and concise, guiding the LLMs through the proof generation process while ensuring that they understand the requirements for correctness and clarity. In App. E.1, we present the prompts used for generating proofs. In App. E.2, we provide the prompt used to generate the LLM summary to aid human graders in identifying potential issues in the proof. In App. E.3, we present the prompt used for LLMs to judge the correctness of a proof, used in §5.2. In App. E.4–E.6, we provide all prompts used for the LLMs in best-of-n sampling, as described in §5.5.

E.1. Proof Generation Prompt

The following prompt is used for problems with no final answer:

Prompt

```
Your task is to write a proof solution to the following problem. Your proof will be graded by human judges for accuracy, thoroughness,
 and clarity. When you write your proof, follow these guidelines:

- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high school or low-level bachelor courses in mathematics should not be used. Any use of such results will immediately give you a
zero grade.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- You should use correct LaTeX notation to write equations and mathematical symbols. You should encompass these equations in
appropriate symbols ("\\(" and "\\)" for inline math, "\\[" and "\\]" for block math) to enhance the clarity of your proof. Do not use
 any unicode characters.
- Your proof should be self-contained.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.

{problem}
```

The following prompt is used for problems with a final answer:

Prompt

```
Your task is to write a proof solution to the following problem. Your proof will be graded by human judges for accuracy, thoroughness,
 and clarity. When you write your proof, follow these guidelines:

- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high school or low-level bachelor courses in mathematics should not be used. Any use of such results will immediately give you a
zero grade.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- You should use correct LaTeX notation to write equations and mathematical symbols. You should encompass these equations in
appropriate symbols ("\\(" and "\\)" for inline math, "\\[" and "\\]" for block math) to enhance the clarity of your proof. Do not use
 any unicode characters.
- Your proof should be self-contained.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.
- Put your final answer within \\boxed{{}}.

{problem}
```

E.2. Issues Interface Prompt

Prompt

```
Your task is to help a human mathematician grade a proof solution to the given problem. In this task, you will write a summary of the
provided proof and highlight potential issues with it.

### Input:
```

````
Your input will consist of the following components:
- **Problem Statement**: A mathematical problem that the proof is attempting to solve.
- **Ground-Truth Solution**: If available, the correct solution to the problem, which can be used as a reference. Note that ground-
truth solutions may not always be provided, can also contain mistakes, and are often overly succinct. The ground-truth proof is mainly
 provided to help you understand the problem better.
- **Proof Solution**: The proof that you need to evaluate. This proof may contain errors, omissions, or unclear steps. The proof was
generated by another language model, which was given the following instructions:
<model_prompt>
- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high school or low-level bachelor courses in mathematics should not be used. Any use of such results will immediately give you a
zero grade.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- You should use correct LaTeX notation to write equations and mathematical symbols. You should encompass these equations in
appropriate symbols ("\\(" and "\\)" for inline math, "\\[" and "\\]" for block math) to enhance the clarity of your proof. Do not use
 any unicode characters.
- Your proof should be self-contained.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.
</model_prompt>

### Summary Guidelines:

First, you should write a concise summary of the proof solution. The summary should capture the main ideas and steps of the proof, but
 it does not need to be exhaustive. The goal is to provide a clear overview of what the proof is attempting to accomplish.
A summary should consist of only a few sentences, and it should not contain any judgment or evaluation of the proof. It should be
purely descriptive.

### Potential Issues to Highlight:

Your main task is to identify potential issues in the proof solution. You should include any and all issues that you can find, no
matter how small. Here are some common types of issues to look for:
- **Overgeneralization**: The generated proof proceeds by proving the problem in one or more specific cases, and then concludes that
the result holds in general. However, it does not provide a proof for the general case.
- **Oversimplification**: The proof marks steps as trivial or obvious without proper justification. Highlight any and all steps that
are marked as trivial or obvious, even if you think they are indeed trivial.
- **Skipping Computation Steps**: Proofs that skip computation steps or do not explain transformations clearly can lead to
misunderstandings. Highlight any steps where the proof skips necessary calculations or explanations.
- **Citing Non-Standard Works or Theorems**: Some models may cite theorems or results that are not well-known or are not typically
taught in high school or low-level bachelor courses. Such theorems are only allowed if they are well known. If the proof cites such
results, highlight this as a potential issue, even if you think it is justified.
- **Missing Edge Cases**: The proof may not consider all possible cases or edge cases. If you notice that the proof does not address
certain scenarios, highlight this as a potential issue.
- **Wrong Final Answer**: If the proof arrives at a final answer that is incorrect, highlight this as a potential issue.
- **Other**: Any other issues that do not fit into the above categories but you believe are significant enough to be highlighted.

For each of these issues, you should identify where in the proof they occur, provide a brief explanation of the issue, and indicate
the category of the issue.

If there are more than four issues, you should only highlight the four most significant ones. Sort the issues by their significance,
with the most significant issue first.

### Additional Instructions:

- Do not provide a final grade or score for the proof. Your task is to summarize and highlight potential issues, not to evaluate the
proof as a whole.
- Be critical and thorough in your analysis. If you find no issues, you probably did not look closely enough.
- If you are unsure whether something is an issue, it is better to highlight it and let the human grader decide.
- Use clear and concise language in your summary and issue descriptions. The goal of your response is to help and speed up the human
grader's work, not to add extra work for them. The more clear and concise your response is, the better it will be for the human grader
.
- You should use correct LaTeX notation to write equations and mathematical symbols in your output JSON. You should encompass these
equations in appropriate symbols ("\\(" and "\\)" for inline math, "\\[" and "\\]" for block math) to enhance the clarity of your
proof. Do not use any unicode characters.
- Properly escape all symbols in your output JSON. For example, use `\\` for a single backslash.
- Spend special attention to producing valid JSON. It needs to be parsable by a standard JSON parser.

### Output Format:

Format your reply using a JSON object as follows:

```json
{{
"summary": "A concise summary of the proof solution.",
"issues": [
   {{
````
````

````
    "location": "A description of where the issue occurs in the proof",
    "text": "A citation or excerpt from the proof that contains the issue. If the issue is not contained to a very small part of the
    proof (e.g., a single sentence), you can leave this field empty.",
    "description": "A brief explanation of the issue.",
    "category": "The category of the issue (Overgeneralization, Oversimplification, Skipping Computation Steps, Citing Non-Standard
    Works or Theorems, Missing Edge Cases, Wrong Final Answer, Other)."
    }},
    ...
]
}}
```
If you truly cannot find any issues, you can return an empty issues array (either null or an empty list).

### Problem Statement:
{problem}

### Ground-Truth Solution:
{ground_truth_solution}

### Proof Solution:
{proof_solution}
````

E.3. LLM as Judge Prompt

Prompt

```
You are judging the correctness of an LLM-generated proof for a math problem.

### Input:

Your input will consist of the following components:
- **Problem Statement**: A mathematical problem that the proof is attempting to solve.
- **Proof Solution**: The proof that you need to evaluate. This proof may contain errors, omissions, or unclear steps. The proof was
generated by another language model, which was given the following instructions:
<model_prompt>
- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high school or low-level bachelor courses in mathematics should not be used. Any use of such results will immediately give you a
zero grade.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- You should use correct LaTeX notation to write equations and mathematical symbols. You should encompass these equations in
appropriate symbols ("\\(" and "\\)" for inline math, "\\[" and "\\]" for block math) to enhance the clarity of your proof. Do not use
 any unicode characters.
- Your proof should be self-contained.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.
</model_prompt>

### How the solution should be graded:
A solution should be considered correct even if it would earn 5+/7 points in a standard grading format. Examples of small penalties
worth 1 point are if the solution:
- Makes a small computational mistake that can be easily fixed
- Misses an edge case which can be easily proven/disproven
- Skips over a step that follows without much reasoning or manual work
Depending on the severity and the context, you may also not penalise a given error. On the other hand, a solution should be marked as
incorrect if:
- It marks a step as trivial, if it is not immediately obvious with little reasoning why this would be the case.
- It omits algebra-heavy computational steps, regardless of whether or not it has outlined the methodology. Skipping shorter
computations should be permitted.
- Generalizes over a pattern without rigorously describing the pattern, or without proving any relevant properties.
- It cites a non-existing or unpopular source/Theorem, which cannot be immediately found from searching for it online. Thus, any
theorems that can be immediately found and have a Wikipedia article are allowed.

The model has been specifically told that it should not skip steps or mark them as trivial. Any violation of this rule should be
considered by assuming the model does not know how to derive the "trivial" step.

### Scoring instructions

If you believe the proof is correct, end your analysis with \\boxed{{correct}}. If you believe the proof is incorrect, end your
analysis with \\boxed{{incorrect}}.

### Problem Statement:
{problem}
```
```

```
### Model Solution:
{solution}
```

E.4. LLM as Discrete Judge Prompt

Prompt

```
You are judging the correctness of an LLM-generated proof for a math problem.

### Input:

Your input will consist of the following components:
- **Problem Statement**: A mathematical problem that the proof is attempting to solve.
- **Proof Solution**: The proof that you need to evaluate. This proof may contain errors, omissions, or unclear steps. The proof was
generated by another language model, which was given the following instructions:
<model_prompt>
- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high school or low-level bachelor courses in mathematics should not be used. Any use of such results will immediately give you a
zero grade.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- You should use correct LaTeX notation to write equations and mathematical symbols. You should encompass these equations in
appropriate symbols ("\\(" and "\\)" for inline math, "\\[" and "\\]" for block math) to enhance the clarity of your proof. Do not use
 any unicode characters.
- Your proof should be self-contained.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.
</model_prompt>

### How the solution should be graded:
A solution should be considered correct even if it would earn 5+/7 points in a standard grading format. Examples of small penalties
worth 1 point are if the solution:
- Makes a small computational mistake that can be easily fixed
- Misses an edge case which can be easily proven/disproven
- Skips over a step that follows without much reasoning or manual work
Depending on the severity and the context, you may also not penalise a given error. On the other hand, a solution should be marked as
incorrect if:
- It marks a step as trivial, if it is not immediately obvious with little reasoning why this would be the case.
- It omits algebra-heavy computational steps, regardless of whether or not it has outlined the methodology. Skipping shorter
computations should be permitted.
- Generalizes over a pattern without rigorously describing the pattern, or without proving any relevant properties.
- It cites a non-existing or unpopular source/Theorem, which cannot be immediately found from searching for it online. Thus, any
theorems that can be immediately found and have a Wikipedia article are allowed.

### Further Potential Issues:

Here are some common types of issues to look for:
- **Overgeneralization**: The generated proof proceeds by proving the problem in one or more specific cases, and then concludes that
the result holds in general. However, it does not provide a proof for the general case.
- **Oversimplification**: The proof marks steps as trivial or obvious without proper justification.
- **Skipping Computation Steps**: Proofs that skip computation steps or do not explain transformations clearly can lead to
misunderstandings.
- **Citing Non-Standard Works or Theorems**: Some models may cite theorems or results that are not well-known or are not typically
taught in high school or low-level bachelor courses. Such theorems are only allowed if they are well known.
- **Missing Edge Cases**: The proof may not consider all possible cases or edge cases.

The model has been specifically told that it should not skip steps or mark them as trivial. Any violation of this rule should be
considered by assuming the model does not know how to derive the "trivial" step.

### Scoring instructions

If you believe the proof is correct, end your analysis with \\boxed{{correct}}. If you believe the proof is incorrect, end your
analysis with \\boxed{{incorrect}}.

### Problem Statement:
{problem}

### Model Solution:
{solution}
```

E.5. LLM as Continuous Judge Prompt

Prompt

```
You are judging the correctness of an LLM-generated proof for a math problem.

### Input:

Your input will consist of the following components:
- **Problem Statement**: A mathematical problem that the proof is attempting to solve.
- **Proof Solution**: The proof that you need to evaluate. This proof may contain errors, omissions, or unclear steps. The proof was
generated by another language model, which was given the following instructions:
<model_prompt>
- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high school or low-level bachelor courses in mathematics should not be used. Any use of such results will immediately give you a
zero grade.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- You should use correct LaTeX notation to write equations and mathematical symbols. You should encompass these equations in
appropriate symbols ("\\(" and "\\)" for inline math, "\\[" and "\\]" for block math) to enhance the clarity of your proof. Do not use
 any unicode characters.
- Your proof should be self-contained.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.
</model_prompt>

### How the solution should be graded:
A solution should be graded out of a total of 7 points. Examples of small penalties worth 1 point are if the solution:
- Makes a small computational mistake that can be easily fixed
- Misses an edge case which can be easily proven/disproven
- Skips over a step that follows without much reasoning or manual work
Depending on the severity and the context, you may also not penalise a given error. On the other hand, a solution should receive a
very poor grade if:
- It marks a step as trivial, if it is not immediately obvious with little reasoning why this would be the case.
- It omits algebra-heavy computational steps, regardless of whether or not it has outlined the methodology. Skipping shorter
computations should be permitted.
- Generalizes over a pattern without rigorously describing the pattern, or without proving any relevant properties.
- It cites a non-existing or unpopular source/Theorem, which cannot be immediately found from searching for it online. Thus, any
theorems that can be immediately found and have a Wikipedia article are allowed.

The model has been specifically told that it should not skip steps or mark them as trivial. Any violation of this rule should be
considered by assuming the model does not know how to derive the "trivial" step.

### Further Potential Issues:

Here are some common types of issues to look for:
- **Overgeneralization**: The generated proof proceeds by proving the problem in one or more specific cases, and then concludes that
the result holds in general. However, it does not provide a proof for the general case.
- **Oversimplification**: The proof marks steps as trivial or obvious without proper justification.
- **Skipping Computation Steps**: Proofs that skip computation steps or do not explain transformations clearly can lead to
misunderstandings.
- **Citing Non-Standard Works or Theorems**: Some models may cite theorems or results that are not well-known or are not typically
taught in high school or low-level bachelor courses. Such theorems are only allowed if they are well known.
- **Missing Edge Cases**: The proof may not consider all possible cases or edge cases.

### Scoring instructions

Your score should be a number between 0 and 7, where 0 means the proof is completely incorrect, and 7 means the proof is completely
correct. Be very critical in your grading. If you find small errors, deduct points accordingly.

### Output Format:

At the end of your analysis, present your grade as a number between 0 and 7 in "□".

### Problem Statement:
{problem}

### Model Solution:
{solution}
```

E.6. LLM as Rank Judge Prompt

Prompt

```
You are judging which of the two LLM-generated proofs for a given math problem is better.

### Input:

Your input will consist of the following components:
- **Problem Statement**: A mathematical problem that the proof is attempting to solve.
- **Proof Solution A/B**: The proofs that you need to evaluate. This proof may contain errors, omissions, or unclear steps. Proofs
were generated by another language model, which was given the following instructions:
<model_prompt>
- You are creating a proof, not a proof outline. Each step should be carefully explained and documented. If not properly explained,
the judge will assume that you cannot explain it, and therefore decrease your grade.
- You can use general theorems and lemmas, but only if they are well-known. As a rule of thumb: if the result has a name and is famous
 enough to have a Wikipedia page or something similar to describe it, it is allowed. Any result from papers that would not be taught
in high school or low-level bachelor courses in mathematics should not be used. Any use of such results will immediately give you a
zero grade.
- Do not skip computation steps in your proof. Clearly explain what transformations were done and why they are allowed in each step of
 a calculation.
- You should use correct LaTeX notation to write equations and mathematical symbols. You should encompass these equations in
appropriate symbols ("\\(" and "\\)" for inline math, "\\[" and "\\]" for block math) to enhance the clarity of your proof. Do not use
 any unicode characters.
- Your proof should be self-contained.
- If you are not sure about a specific step, or do not know how to prove an intermediate result, clearly state this. It is much
preferable to indicate your uncertainty rather than making incorrect statements or claims.
</model_prompt>

### How the solution should be graded:
The following examples are small mistakes that should only be slightly penalised:
- Makes a small computational mistake that can be easily fixed
- Misses an edge case which can be easily proven/disproven
- Skips over a step that follows without much reasoning or manual work
On the other hand, a solution should should be severely penalised if:
- It marks a step as trivial, if it is not immediately obvious with little reasoning why this would be the case.
- It omits algebra-heavy computational steps, regardless of whether or not it has outlined the methodology. Skipping shorter
computations should be permitted.
- Generalizes over a pattern without rigorously describing the pattern, or without proving any relevant properties.
- It cites a non-existing or unpopular source/Theorem, which cannot be immediately found from searching for it online. Thus, any
theorems that can be immediately found and have a Wikipedia article are allowed.

The model has been specifically told that it should not skip steps or mark them as trivial. Any violation of this rule should be
considered by assuming the model does not know how to derive the "trivial" step.

### Further Potential Issues:

Here are some common types of issues to look for:
- **Overgeneralization**: The generated proof proceeds by proving the problem in one or more specific cases, and then concludes that
the result holds in general. However, it does not provide a proof for the general case.
- **Oversimplification**: The proof marks steps as trivial or obvious without proper justification.
- **Skipping Computation Steps**: Proofs that skip computation steps or do not explain transformations clearly can lead to
misunderstandings.
- **Citing Non-Standard Works or Theorems**: Some models may cite theorems or results that are not well-known or are not typically
taught in high school or low-level bachelor courses. Such theorems are only allowed if they are well known.
- **Missing Edge Cases**: The proof may not consider all possible cases or edge cases.

### Scoring instructions

You should compare the two proofs and determine which one is better. If you believe Proof A is better, end your analysis with \\boxed
{{A}}. If you believe Proof B is better, end your analysis with \\boxed{{B}}. If you believe both proofs are equally good, end your
analysis with \\boxed{{equal}}.

### Problem Statement:
{problem}

### Proof Solution A:
{solution_a}

### Proof Solution B:
{solution_b}
```