

Towards Robust Natural-Language Proof Verification

Stanford CS224N Custom

Slim Barkallah

Department of Computer Science
Stanford University
slimbark@stanford.edu

Abstract

Recently, large language models (LLMs) have achieved strong performance on mathematical tasks [1, 2], reaching high accuracy on popular benchmarks such as AIME and MATH [3] through reinforcement learning from verified rewards. However, extending these gains to domains where such rewards are unavailable, such as proof generation, requires robust neural verifiers. In this work, we introduce a natural-language verifier that leverages a scaffolded pipeline of multiple state-of-the-art LLM calls to improve verification performance. Our approach achieves 80% accuracy on our subset of the IMO-GradingBench [4] benchmark, compared to a 74% baseline obtained with a single LLM call. Using our method, we also identified several incorrect labels in the benchmark, helping improve its reliability for the community. Although our system does not yet reach human-level performance, we hope this work opens a new direction for studying how to scaffold current reasoning models to achieve more robust natural-language verification. Such verifiers could enable new research directions and further advance LLM reasoning capabilities.

1 Key Information to include

- Mentor: Luke Bailey
- External Collaborators: Prof. Tenguy Ma
- Sharing project: No
- Late Days :

Team member	Late days remaining
Slim Barkallah	5
Total pooled	5
Team size	1
Late days applied for the report	3

2 Introduction

In reinforcement learning (RL) for mathematical reasoning, large language models (LLMs) are typically rewarded based on whether their predicted final answers match the ground-truth solutions to quantitative problems [5]. This paradigm has enabled frontier LLMs to achieve near-saturation performance on benchmarks that emphasize final-answer correctness, such as AIME and MATH [3]. However, this reward mechanism has a key limitation: it does not apply to theorem-proving tasks, where problems often lack numerical answers and instead require verifying rigorous derivations. Consequently, tasks such as proof generation require more robust verification methods.

Formal systems such as Lean [6] and Isabelle [7] address this verification problem by expressing proofs in a formal language with compilers that perform rigorous, step-by-step checking. In practice, however, their use typically relies on *autoformalization*: translating natural-language proofs into

fully formal representations [8, 9, 10, 11]. Unlike natural language, formal systems require explicit justification for even trivial steps, making autoformalization particularly challenging [12].

Prior work has also explored verifying proofs directly in natural language without relying on formal systems. Some approaches train natural-language verifiers on large human-labeled datasets [13, 14, 15, 16], but producing high-quality annotations is expensive and difficult to scale. Other efforts instead scaffold off-the-shelf LLMs for natural-language proof verification without additional training [17, 18, 19]. While these approaches improve over simple baselines, a substantial gap to human-level verification remains.

Our contributions are as follows:

- We introduce a novel scaffolding method that leverages off-the-shelf LLM reasoning models, achieving 80% accuracy on our evaluation subset of the IMO-GradingBench [4] benchmark, compared to a 74% baseline from a single LLM verification call.
- Using our method, we identify discrepancies and labeling errors in the IMO-GradingBench [4] benchmark, where some incorrect proofs with subtle mistakes were labeled as receiving full credit.
- We provide qualitative analyses of some successful and unsuccessful predictions, along with ablation studies demonstrating the contribution of some component of our method.

3 Related Work

Several approaches train dedicated verifier models to evaluate mathematical reasoning. DeepSeekMath-V2 [13] trains a natural-language verifier with reinforcement learning on human-labeled data and uses it to iteratively improve a prover through meta-verification. Heimdall [15] similarly trains a verifier using binary human labels and applies it at inference time to select the least-uncertain solution among sampled answers. While these methods improve performance, they rely on human-labeled supervision to train the verifier.

Other work studies the reliability of LLM-based proof grading. Luong et al. [4] evaluate LLMs as graders and find high correlation with human judgments when reference solutions are available, but substantially lower accuracy (44.3%) when grading without references. Pessimistic Verification [17] improves verification by recursively prompting a model to identify errors in proof chunks, achieving 86% accuracy on IMO-GradingBench, though still below human-level verification.

Several works explore using verification as part of the inference process. Zhao et al. [19] propose a sampling-based search framework that generates multiple candidate solutions and compares them to localize reasoning errors, showing that larger sample pools improve verification performance.

Another line of work integrates verification directly into the training objective. Wang et al. [20] train models to both solve and verify problems using a modified GRPO objective that separates solution and verification rewards. Liu et al. [21] introduce a two-step reinforcement learning framework where models first generate solutions and then verify them during training. Multi-agent approaches such as Debate4MATH [22] employ debate between supporting and critic agents to label proof steps and train reward models.

Despite these advances, existing methods often rely on human-labeled data, focus primarily on final-answer verification, or still fall short of human-level proof verification.

4 Approach

4.1 Baselines

- **Single LLM call:** We compare two prompting strategies: with and without a reference solution. In both cases, the model predicts whether the proof is correct.
- **Majority voting:** We run n LLM calls and predict that a proof is correct if the majority of the model calls classify it as correct.
- **Pessimistic approach:** By pessimistic approach, we mean that for each proof we run n evaluations and predict that a proof is correct only if all model calls classify it as correct.

4.2 Current Approach

4.2.1 Motivation.

Two observations motivate our approach.

- Some errors are easier to detect when reasoning is examined in smaller blocks rather than in a long proof.
- Each block must retain sufficient context to remain interpretable. Without this context, a verifier may incorrectly flag a block due to missing definitions or previously established results.

4.2.2 Method

Our verification pipeline consists of the following steps:

1. **Proof decomposition.** We decompose the proof into k blocks. The proof is first split using the delimiter `\n\n`, after which segments are regrouped into k blocks while maximizing the minimum block length. The full algorithm is described in Appendix B.
2. **Context construction.** For each block, we make one LLM call that takes the current block together with summaries of previous blocks. The model extracts key mathematical claims and definitions introduced in the current block, which will be used as context for subsequent blocks.
3. **Block-level verification.** For each block, we run n independent verification calls. Each call receives the block and the summary of previously established results as input.
4. **Meta-verification.** Finally, we perform a meta-verification step that verifies the verifier outputs. We collect block-level reviews that flagged errors and provide them, along with the full proof, to the model. The model then double-check detected errors and determines whether they are valid.

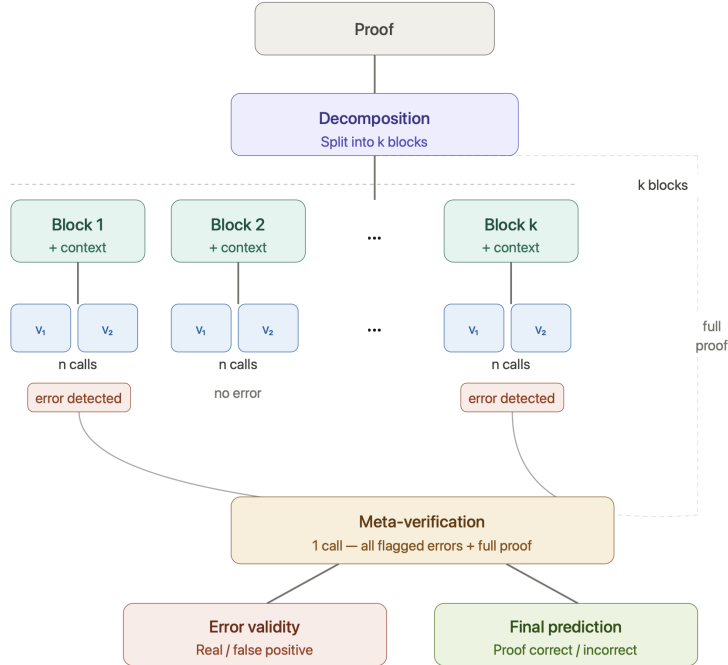


Figure 1: Overview of the verification pipeline.

4.2.3 Novelty

Although proof decomposition [17] and meta-verification [13] have each been explored previously, their combination for natural-language proof verification remains largely unexplored. To our knowledge, our approach is the first to combine explicit block-level decomposition with a meta-verification stage to improve verification reliability.

5 Experiments

5.1 Data

The data used in this project primarily comes from IMO-GradingBench [4]. IMO-GradingBench contains 1,000 samples. Each sample consists of a problem, a reference solution, grading guidelines, an LLM-generated solution, and human-assigned scores on a 0–7 scale. The benchmark is constructed from 30 advanced IMO-level problems. Except for 5.4.1, we evaluate the other methods on a subset of IMO-GradingBench. We select 150 proofs from 15 problems.

5.2 Evaluation method

We report accuracy as the evaluation metric. In IMO-GradingBench, ground truth scores range from 0 to 7. Following the paper’s protocol, the model classifies each proof into four categories: correct (7), almost correct (6), partial (1), or incorrect (0).

Since some experiments are conducted without a reference solution in the prompt, partial credit is difficult to assess, we use binary evaluation: a proof is labeled correct if the model assigns a score of 7, and incorrect otherwise.

5.3 Experimental details

In all the following experiments, we use GPT-5-mini as the inference model. For both the majority voting and pessimistic approaches, we set $n = 10$ as the maximum number of verification calls per problem.

For the main approach, we set $k = 5$ blocks and vary n from 1 to 8, where n denotes the number of verification calls per block. We use GPT-5-mini as the base model for both the block-level verification and the meta-verification stages. The prompts used in all experiments are provided in the appendix.A

5.4 Results

5.4.1 Single LLM call.

Figure 2 shows that the performance difference between the prompt with a reference solution and the prompt without a reference is minimal. Therefore, we continue the remaining experiments using the less constrained prompt (without reference). The single LLM call approach achieves 81.90% accuracy on the full benchmark and 74% on our evaluation subset.

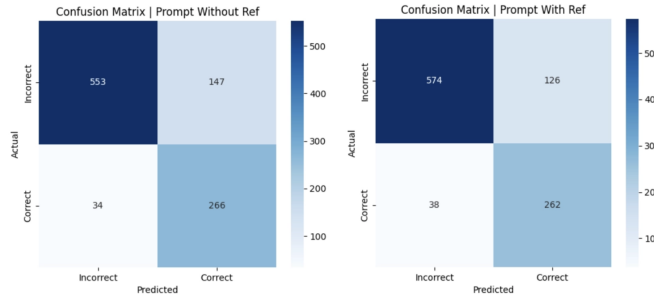


Figure 2: Comparison of confusion matrices with and without a reference solution in the prompt.

5.4.2 Majority voting vs. Pessimistic approach.

Increasing the number of verification calls does not significantly improve accuracy under majority voting. In contrast, the pessimistic approach scales better, reaching 78% accuracy. The results are presented in 3.

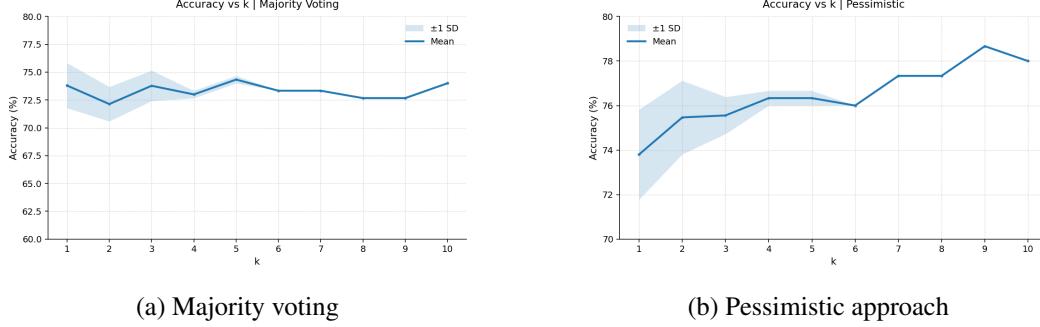


Figure 3: Accuracy vs. number of verification calls. We use the same 10 calls to estimate performance for $k < 10$. For example, when $k = 2$, we split the 10 calls into 5 disjoint pairs, compute the majority-vote accuracy for each pair, and report the mean and standard deviation across the 5 values.

5.4.3 Our Main approach

Figure 4 presents a comparison of accuracy, false positives, and false negatives across different methods and ablations.

Our method reaches 80% accuracy when using $n = 2$ verification calls per block, which is the highest score achieved so far. This represents a +6% improvement over the baseline that uses a single LLM call to verify the full proof.

We observe that adding meta-verification significantly improves performance compared to the method without this step (green vs. blue in Figure 4a), suggesting that the meta-verifier effectively filters invalid error detections. Adding contextual information yields a small additional improvement in accuracy (purple vs. blue in Figure 4a)

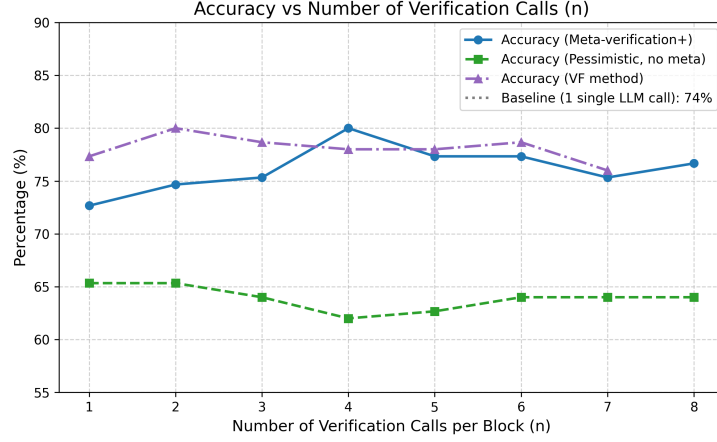
However, scaling the approach further does not lead to additional improvements. Performance slightly decreases when $n \geq 5$. Figures 4b 4c show that the number of false positives decreases and plateaus after $n = 5$, while the number of false negatives increases slightly.

6 Analysis

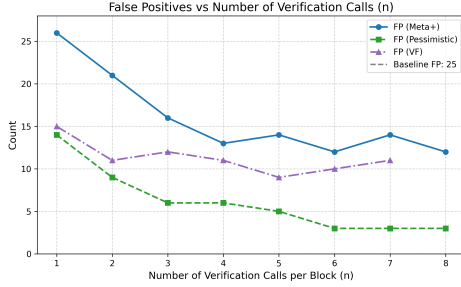
Since our method achieves 80% accuracy, we manually inspected the remaining 20% of cases (30 IMO level proofs), spending about 40 hours analyzing them. We found that 15 false negatives correspond to incorrect labels in the benchmark: these proofs contain clear mathematical flaws correctly identified by our method, suggesting the true accuracy is likely higher. For the false positives, the manual analysis focused on identifying errors that the model failed to detect.

We present two illustrative examples: one proof containing an error missed by our method, and one proof labeled as correct in the benchmark but for which our method correctly identifies a flaw.

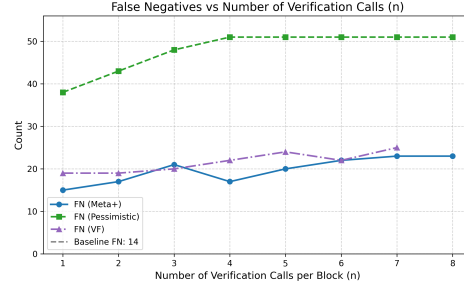
Example 1: GB-0433 This example corresponds to a problem that was not detected by our approach. The error arises from an incorrect use of an inequality. After applying Jensen’s inequality to S_+ and S_- , two inequalities are obtained with opposite directions. These inequalities cannot be directly combined to derive a valid upper bound for the total sum S . The full proof is provided in the appendix. C



(a) Accuracy



(b) False positives



(c) False negatives

Figure 4: Comparison of proof verification methods. We evaluate four approaches. A single LLM call that verifies the full proof (baseline). A pessimistic decomposition approach (green plot), where the proof is split into $k = 5$ blocks and the proof is predicted as incorrect if any block contains an error; this method does not use block-level context or meta-verification. Meta-verification (blue plot) extends the decomposition approach by adding a meta-verification step that filters detected errors. VF Method (purple plot, our method) augments the per-block verification prompt with contextual information from previously established results and performs a final meta-verification step to filter invalid errors.

Block with Error (Example 1: GB-0433)

We apply Jensen's inequality to the two parts of the sum $S = S_- + S_+$.

For the positive part $S_+ = \sum_{k \in I_+} w_k h(t_k)$. Since $t_k \geq 0$ and $h(t)$ is concave on $[0, \infty)$:

$$S_+ = W_+ \sum_{k \in I_+} \frac{w_k}{W_+} h(t_k) \leq W_+ h \left(\frac{\sum_{k \in I_+} w_k t_k}{W_+} \right) = W_+ h \left(\frac{T}{W_+} \right).$$

For the negative part $S_- = \sum_{k \in I_-} w_k h(t_k)$. Since $t_k \leq 0$ and $h(t)$ is convex on $(-\infty, 0]$:

$$S_- = W_- \sum_{k \in I_-} \frac{w_k}{W_-} h(t_k) \geq W_- h \left(\frac{\sum_{k \in I_-} w_k t_k}{W_-} \right) = W_- h \left(\frac{-T}{W_-} \right).$$

Since $h(t)$ is odd, $h(-T/W_-) = -h(T/W_-)$.

$$S_- \geq -W_- h \left(\frac{T}{W_-} \right).$$

Therefore,

$$S = S_+ + S_- \leq W_+ h\left(\frac{T}{W_+}\right) - W_- h\left(\frac{T}{W_-}\right).$$

Example 2 : GB-0254 This example corresponds to a false negative caused by an incorrect label in the benchmark, which was successfully identified by our method. The issue in the proof is the use of a general property of the Carmichael function that was assumed in the proof to be a "known" result. However, this property does not always hold. A counterexample, $c = 10$ (where equality holds), was identified by our method. The full proof is provided in the appendix. C

Subset of GPT-5-mini review

2) Incorrect / unsupported assertion about valuations of $\lambda(c)$ (important)

The proposer claims: "For any prime divisor p of c , $v_p(\lambda(c)) < v_p(c)$. This is a known property of the Carmichael function." This statement is not generally correct as written. Counterexample (simple): take $c = 21 = 3 \cdot 7$. Then

$$\lambda(21) = \text{lcm}(\lambda(3), \lambda(7)) = \text{lcm}(2, 6) = 6,$$

and $v_3(\lambda(21)) = 1$ while $v_3(21) = 1$; so

$$v_3(\lambda(c)) = v_3(c),$$

not strictly less. The proposer needs a correct argument that $\gcd(M, c) < c$. The text attempts to deduce this from $v_p(\lambda(c)) < v_p(c)$, but the inequality $v_p(\lambda(c)) < v_p(c)$ is not universally valid. Whether $\gcd(M, c) < c$ is true in fact for all $c > 1$ is a separate claim and needs proof; it cannot be taken as an immediate corollary of the incorrect valuation statement.

In short: the step showing $d = \gcd(M, c)$ is a proper divisor of c is not established by the argument given.

7 Conclusion

In this work, we propose an approach that leverages the reasoning capabilities of LLMs by combining proof decomposition with a meta-verification step to improve proof verification using off-the-shelf models. Our method achieves 80% accuracy on our subset of the IMO-GradingBench [4], compared to a 74% baseline obtained with a single LLM call. We also identified several incorrect labels in the current benchmark, highlighting the need for more robust evaluation.

Despite this progress, a significant gap remains between automated and human proof verification. Our method could be further improved by exploring more effective decomposition strategies or incorporating multiple meta-verification calls combined with majority voting to obtain a more reliable verification signal. In addition, due to cost constraints, all experiments were conducted using the gpt-5-mini model; larger models may yield further improvements. Finally, our experiments use only 15% of the current IMO-GradingBench, and evaluating on the full benchmark could provide clearer and less noisy insights.

References

- [1] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
- [2] OpenAI. Openai o1 system card, December 2024.
- [3] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *CoRR*, abs/2103.03874, 2021.
- [4] Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu,

- Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.
- [5] 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.
- [6] Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28: 28th International Conference on Automated Deduction*, pages 625–635, Virtual Event, 2021. Springer.
- [7] Tobias Nipkow, Markus Wenzel, and Lawrence C. Paulson. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002.
- [8] Jin Peng Zhou, Kilian Q. Weinberger, Charles Staats, Wenda Li, Yuhuai Wu, and Christian Szegedy. Don’t trust: Verify – grounding llm quantitative reasoning with autoformalization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.
- [9] L. Chen, J. Gu, L. Huang, W. Huang, Z. Jiang, A. Jie, X. Jin, X. Jin, C. Li, K. Ma, C. Ren, J. Shen, W. Shi, T. Sun, H. Sun, J. Wang, S. Wang, Z. Wang, C. Wei, S. Wei, Y. Wu, Y. Wu, Y. Xia, H. Xin, F. Yang, H. Ying, H. Yuan, Z. Yuan, T. Zhan, C. Zhang, Y. Zhang, G. Zhang, T. Zhao, J. Zhao, Y. Zhou, and T. H. Zhu. Seed-prover: Deep and broad reasoning for automated theorem proving. *CoRR*, abs/2507.23726, 2025.
- [10] AlphaProof and A. Teams. Ai achieves silver-medal standard solving international mathematical olympiad problems. <https://deepmind.google/blog/ai-solves-imo-problems-at-silver-medal-level>, 2024. DeepMind Blog.
- [11] Z. Z. Ren, Z. Shao, J. Song, H. Xin, H. Wang, W. Zhao, L. Zhang, Z. Fu, Q. Zhu, D. Yang, Z. F. Wu, Z. Gou, S. Ma, H. Tang, Y. Liu, W. Gao, D. Guo, and C. Ruan. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *CoRR*, abs/2504.21801, 2025.
- [12] Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. Rethinking and improving autoformalization: Towards a faithful metric and a dependency retrieval-based approach. In *Proceedings of the International Conference on Learning Representations (ICLR 2025)*, 2025.
- [13] Chengda Lu Z.Z. Ren Jiewen Hu Tian Ye Zhibin Gou Shirong Ma Xiaokang Zhang Zhihong Shao, Yuxiang Luo. Deepseekmath-v2: Towards self-verifiable mathematical reasoning, 2025.
- [14] Zijian Wu, Ling kai Kong, Wenwei Zhang, Songyang Gao, Yuzhe Gu, Zhongrui Cai, Tianyou Ma, Yuhong Liu, Zhi Wang, Runyuan Ma, Guangyu Wang, Wei Li, Conghui He, Dahua Lin, and Kai Chen. Opv: Outcome-based process verifier for efficient long chain-of-thought verification. *arXiv preprint arXiv:2512.10756*, 2025.
- [15] Wenlei Shi and Xing Jin. Heimdall: Test-time scaling on the generative verification, 2025.
- [16] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- [17] Zejin Lin Peng Li Yang Liu Yanxing Huang, Zihan Tang. Pessimistic verification for open ended math questions, 2025.
- [18] Zhuoxuan Jiang, Haoyuan Peng, Shanshan Feng, Fan Li, and Dongsheng Li. Llms can find mathematical reasoning mistakes by pedagogical chain-of-thought. 2024.
- [19] Eric Zhao, Pranjal Awasthi, and Sreenivas Gollapudi. Sample, scrutinize and scale: Effective inference-time search by scaling verification. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, May 2025. Poster presentation.

- [20] Xiaoxuan Wang, Bo Liu, Song Jiang, Jingzhou Liu, Jingyuan Qi, Xia Chen, and Baosheng He. From solving to verifying: A unified objective for robust reasoning in llms, November 2025.
- [21] Xiaoyuan Liu, Tian Liang, Zhiwei He, Jiahao Xu, Wenxuan Wang, Pinjia He, Zhaopeng Tu, Haitao Mi, and Dong Yu. Trust, but verify: A self-verification approach to reinforcement learning with verifiable rewards. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025.
- [22] Shaowei Zhang and Deyi Xiong. Debate4MATH: Multi-agent debate for fine-grained reasoning in math. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16810–16824, Vienna, Austria, July 2025. Association for Computational Linguistics.

A Prompts

Listing 1: Prompt for proofs verification. This prompt is adapted from the IMO-Bench paper, with the proof reference removed.

You are an expert grader for the International Mathematics Olympiad (IMO). Your task is to evaluate a proposed solution strictly and rigorously.

Keep in mind the standards at the IMO are extremely high: only arguments that are logically sound, complete, and precise should be rewarded.

General Scoring Rubric

Scores are assigned on a 0\7 scale. The general guidelines are:

* **7 Points (Correct):**

The solution is complete, correct, and fully rigorous.

If the submission contains incorrect attempts or lines of reasoning but ultimately presents a complete and correct solution, it should still be awarded full points; the presence of earlier, discarded work does not detract from the final correct proof.

* **6 Points (Almost Correct):**

The solution is almost correct with a sound core argument, but contains minor errors in calculation or small gaps in logic. Missing proofs for major components, unjustified claims, or sketchy arguments are **not** eligible for 6 points.

* **1 Point (Partial Progress):**

The solution demonstrates substantial progress. Initial observations, reformulating the problem without making substantive headway are generally **not** eligible for this score.

* **0 Points (Incorrect):**

The solution does not make substantial progress that is a key step in the full solution or is fundamentally flawed. All partial progress without key results or lacking rigor also fall in this category.

Input Data and Interpretation

You are provided with the following:

1. **Problem Statement:**

The IMO problem.

4. ****Proposed Solution:****
The student submission.

Evaluation Process

You must follow this structured process:

1. ****Step-by-Step Verification:****
Verify the logical validity and rigor of every step. Identify all flaws, gaps, assumptions, and errors. ****Make sure you fully understand every piece of logic behind each step of the proposed solution; be careful of solutions that pretend to be correct.****
2. ****Assess Progress:****
Determine the extent of non-trivial progress made.
3. ****Score Determination:****
Compare the findings against the Specific Grading Guidelines and the General Rubric to determine the final score.

Output Requirements

You must provide your final score in the format:

<points>N out of 7</points>

Ensure the '<points>' block is used **only once**, as your answer will be parsed based on the first '<points></points>' block that appears in your entire response.

****PROBLEM STATEMENT****
{problem_statement}

****PROPOSED SOLUTION****
{student_answer}

Present your detailed thought process and formal justification based on the scoring rubric and grading guidelines, and finally present your final score in the format below.

[Select one of the following options]

<points>7 out of 7</points>

<points>6 out of 7</points>

<points>1 out of 7</points>

<points>0 out of 7</points>

Context Creation Prompt

You will receive the following inputs:

1. A mathematical problem.
2. A subset of a proof (a proof block).
3. A VF object.

The VF (Verification File) is a numbered sequence of mathematical claims extracted from earlier proof blocks.

Your task is to analyze the proof block and identify the important mathematical claims that should be added to the VF.

The goal of the VF is to capture the high-level logical structure of the proof, not every step of the reasoning.

Important rules:

1. The VF is an ordered numbered list of claims.
2. The VF already contains claims from previous blocks. Do NOT rewrite or repeat existing claims.
3. Your output should contain ONLY the new claims that must be appended to the VF.
4. Extract only ****high-level claims**** that are important for understanding the structure of the proof.
5. Do NOT extract routine steps such as: - algebraic manipulations - intermediate calculations - restatements of previous claims
6. Prefer claims that represent: - key derived properties - important logical consequences - lemmas or intermediate results - the main conclusion of the proof block
7. Each claim must be self-contained and clearly written.
8. Avoid vague references such as "this", "it", or "the above".
10. Avoid phrases such as: - "since" - "because" - "therefore" - "thus" - "Moreover" - "Consequently"

Inputs :

Current VF :

vf

Proof block :

block

Now output ****only**** the new claims that should be appended to the VF, written as numbered items continuing the sequence.

Block verification with context Prompt

You will receive:

1. A mathematical problem.
2. A summary of the previous results of the proof (VF).
3. A new block of the proof.

Your task is to verify whether the mathematical reasoning in the proof block is correct.

Instructions: - Use the previous results (VF) as established facts. - Check whether each step in the proof block logically follows. - Identify any incorrect reasoning, unjustified claims, or missing arguments.

Explain your reasoning clearly.

At the end of your answer, select exactly one of the following:

CORRECT INCORRECT

Problem : problem

Previous results (VF): vf

Proof block: block

Block verification without context Prompt

Can you verify the mathematics in the following part of the proof and determine whether each step is correct?

Provide a detailed justification for your evaluation. Conclude by selecting exactly one of the following options:

[Select one of the following] CORRECT INCORRECT

Part of the Proof : block

Meta-verification Prompt

You are an expert grader for the International Mathematics Olympiad (IMO). Your task is to evaluate a proposed solution strictly and rigorously. Keep in mind the standards at the IMO are extremely high: only arguments that are logically sound, complete, and precise should be rewarded.

General Scoring Rubric

Scores are assigned on a 0–7 scale. The general guidelines are:

* **7 Points (Correct):** The solution is complete, correct, and fully rigorous. If the submission contains incorrect attempts or lines of reasoning but ultimately presents a complete and correct solution, it should still be awarded full points; the presence of earlier, discarded work does not detract from the final correct proof.

* **6 Points (Almost Correct):** The solution is almost correct with a sound core argument, but contains minor errors in calculation or small gaps in logic. Missing proofs for major components, unjustified claims, or sketchy arguments are **not** eligible for 6 points.

* **1 Point (Partial Progress):** The solution demonstrates substantial progress. Initial observations, reformulating the problem without making substantive headway are generally **not** eligible for this score.

* **0 Points (Incorrect):** The solution does not make substantial progress that is a key step in the full solution or is fundamentally flawed. All partial progress without key results or lacking rigor also fall in this category.

Input Data and Interpretation

You are provided with the following:

1. **Problem Statement:** The IMO problem.
2. **Proposed Solution:** The student submission.
3. **Past Reviews:** Previous evaluations of the solution.

Evaluation Process

You must follow this structured process:

1. **Review Analysis:** Carefully read all the reviews and understand the errors they mention. For each review, verify whether the identified errors are valid. Be careful: some alleged errors may not actually be errors.

2. **Score Determination:** Compare the findings against the General Rubric to determine the final score.

Output Requirements

You must provide your final score in the format:

<points>N out of 7</points>

Ensure the '<points>' block is used **only once**, as your answer will be parsed based on the first '<points></points>' block that appears in your entire response.

PROBLEM STATEMENT problem

PROPOSED SOLUTION solution

PAST REVIEWS errors

Present your detailed reasoning regarding each review, and then provide your final score in the required format.

[Select one of the following options]

<points>7 out of 7</points> <points>6 out of 7</points> <points>1 out of 7</points>

<points>0 out of 7</points>

B Algorithm

C Data Analysis

C.1 Example : GB-0433

GB-0433

We are asked to prove the inequality

$$\frac{1}{b_1} + \frac{1}{b_2} + \cdots + \frac{1}{b_n} \geq \frac{1}{2} - \frac{1}{2^{n+1}}$$

where $n \geq 2$, $a_1 \leq a_2 \leq \cdots \leq a_n$ are positive real numbers satisfying $\prod_{k=1}^n a_k = 1$, and $b_k = 2^k \left(1 + a_k^{2^k}\right)$.

Algorithm 1 $k_balanced_blocks(text, k, sep)$

Require: Text string $text$, integer k , separator sep (default: "\n\n")

Ensure: List of k contiguous blocks of $text$ maximizing minimum block length

```
1:  $parts \leftarrow split(text, sep)$ 
2:  $n \leftarrow |parts|$ 
3:  $w \leftarrow [length(p) \text{ for each } p \in parts]$ 
4:  $sep\_len \leftarrow length(sep)$ 
    $can(target)$ 
5:  $count \leftarrow 0$ 
6:  $acc \leftarrow 0$ 
7:  $first\_in\_block \leftarrow \text{True}$ 
8: for  $w_i$  in  $w$  do
9:    $add \leftarrow w_i$  if  $first\_in\_block$  else  $w_i + sep\_len$ 
10:   $acc \leftarrow acc + add$ 
11:  if  $acc \geq target$  then
12:     $count \leftarrow count + 1$ 
13:     $acc \leftarrow 0$ 
14:     $first\_in\_block \leftarrow \text{True}$ 
15:  else
16:     $first\_in\_block \leftarrow \text{False}$ 
17:  end if
18: end for
19:
20: return  $count \geq k$ 
21:  $lo \leftarrow 0, hi \leftarrow \sum w + sep\_len \cdot (n - 1)$ 
22: while  $lo < hi$  do
23:   $mid \leftarrow \lfloor (lo + hi + 1) / 2 \rfloor$ 
24:  if  $can(mid)$  then
25:     $lo \leftarrow mid$ 
26:  else
27:     $hi \leftarrow mid - 1$ 
28:  end if
29: end while
30:  $best\_min \leftarrow lo$ 
31:  $blocks \leftarrow [], start \leftarrow 0, acc \leftarrow 0, first\_in\_block \leftarrow \text{True}$ 
32: for  $i = 0$  to  $n - 1$  do
33:   $add \leftarrow w[i]$  if  $first\_in\_block$  else  $w[i] + sep\_len$ 
34:   $acc \leftarrow acc + add$ 
35:   $remaining\_parts \leftarrow n - (i + 1)$ 
36:   $remaining\_blocks \leftarrow k - |blocks| - 1$ 
37:  if  $acc \geq best\_min$  and  $remaining\_parts \geq remaining\_blocks$  then
38:    Append  $join(parts[start : i + 1], sep)$  to  $blocks$ 
39:     $start \leftarrow i + 1$ 
40:     $acc \leftarrow 0$ 
41:     $first\_in\_block \leftarrow \text{True}$ 
42:  else
43:     $first\_in\_block \leftarrow \text{False}$ 
44:  end if
45: end for
46: if  $start < n$  then
47:  Append  $join(parts[start :], sep)$  to  $blocks$ 
48: end if
49: while  $|blocks| > k$  do
50:  Merge last two blocks:  $blocks[-2] \leftarrow join(blocks[-2], blocks[-1], sep)$ 
51:  Remove last block
52: end while
53: while  $|blocks| < k$  do
54:  Append empty string to  $blocks$ 
55: end while
56: return  $blocks$ 
```

First, we rewrite the inequality. The right-hand side is the sum of a geometric progression:

$$\frac{1}{2} - \frac{1}{2^{n+1}} = \sum_{k=1}^n \left(\frac{1}{2^k} - \frac{1}{2^{k+1}} \right) = \sum_{k=1}^n \frac{1}{2^{k+1}}.$$

The left-hand side is:

$$\sum_{k=1}^n \frac{1}{b_k} = \sum_{k=1}^n \frac{1}{2^k(1+a_k^{2^k})}.$$

The inequality is equivalent to:

$$\sum_{k=1}^n \frac{1}{2^k(1+a_k^{2^k})} \geq \sum_{k=1}^n \frac{1}{2^{k+1}}.$$

Rearranging the terms, we aim to prove:

$$\sum_{k=1}^n \left(\frac{1}{2^k(1+a_k^{2^k})} - \frac{1}{2^{k+1}} \right) = \sum_{k=1}^n \frac{1}{2^{k+1}} \left(\frac{2 - (1+a_k^{2^k})}{1+a_k^{2^k}} \right) = \sum_{k=1}^n \frac{1}{2^{k+1}} \frac{1-a_k^{2^k}}{1+a_k^{2^k}} \geq 0.$$

We introduce a change of variables. Let $t_k = 2^k \ln a_k$. Then $a_k^{2^k} = e^{t_k}$. We use the identity $\frac{1-e^t}{1+e^t} = -\frac{e^{t/2}-e^{-t/2}}{e^{t/2}+e^{-t/2}} = -\tanh(t/2)$. Let $h(t) = \tanh(t/2)$. The inequality becomes:

$$\sum_{k=1}^n \frac{1}{2^{k+1}} (-h(t_k)) \geq 0 \iff \sum_{k=1}^n \frac{1}{2^{k+1}} h(t_k) \leq 0.$$

Multiplying by 2, we aim to prove:

$$S = \sum_{k=1}^n w_k h(t_k) \leq 0,$$

where $w_k = 1/2^k$.

We analyze the constraints on t_k . The condition $\prod a_k = 1$ implies $\sum_{k=1}^n \ln a_k = 0$. Since $\ln a_k = t_k/2^k = w_k t_k$, the constraint is:

$$\sum_{k=1}^n w_k t_k = 0.$$

The condition $a_1 \leq \dots \leq a_n$ implies $\ln a_1 \leq \dots \leq \ln a_n$. Since t_k has the same sign as $\ln a_k$, the signs of t_k are ordered. Thus, there exists an index $m \in \{0, 1, \dots, n\}$ such that $t_k \leq 0$ for $k \in I_- = \{1, \dots, m\}$ and $t_k \geq 0$ for $k \in I_+ = \{m+1, \dots, n\}$.

We analyze the properties of the function $h(t) = \tanh(t/2)$. $h(t)$ is an odd function, and $h(0) = 0$. $h'(t) = \frac{1}{2 \cosh^2(t/2)}$. $h''(t) = -\frac{\sinh(t/2)}{2 \cosh^3(t/2)}$. For $t > 0$, $h''(t) < 0$, so $h(t)$ is strictly concave on $[0, \infty)$. For $t < 0$, $h''(t) > 0$, so $h(t)$ is strictly convex on $(-\infty, 0]$.

If $m = 0$ or $m = n$, then all $\ln a_k = 0$, so all $t_k = 0$. In this case, $S = 0$, and the inequality holds with equality.

Now assume $1 \leq m \leq n-1$. This means not all $a_k = 1$. Let $W_- = \sum_{k \in I_-} w_k$ and $W_+ = \sum_{k \in I_+} w_k$. Let $T = \sum_{k \in I_+} w_k t_k$. Since not all t_k are zero and I_+ is non-empty, $T > 0$. From the constraint $\sum w_k t_k = 0$, we have $\sum_{k \in I_-} w_k t_k = -T$.

We apply Jensen's inequality to the two parts of the sum $S = S_- + S_+$.

For the positive part $S_+ = \sum_{k \in I_+} w_k h(t_k)$. Since $t_k \geq 0$ and $h(t)$ is concave on $[0, \infty)$:

$$S_+ = W_+ \sum_{k \in I_+} \frac{w_k}{W_+} h(t_k) \leq W_+ h \left(\frac{\sum_{k \in I_+} w_k t_k}{W_+} \right) = W_+ h \left(\frac{T}{W_+} \right).$$

For the negative part $S_- = \sum_{k \in I_-} w_k h(t_k)$. Since $t_k \leq 0$ and $h(t)$ is convex on $(-\infty, 0]$:

$$S_- = W_- \sum_{k \in I_-} \frac{w_k}{W_-} h(t_k) \geq W_- h \left(\frac{\sum_{k \in I_-} w_k t_k}{W_-} \right) = W_- h \left(\frac{-T}{W_-} \right).$$

Since $h(t)$ is odd, $h(-T/W_-) = -h(T/W_-)$.

$$S_- \geq -W_- h\left(\frac{T}{W_-}\right).$$

Therefore,

$$S = S_+ + S_- \leq W_+ h\left(\frac{T}{W_+}\right) - W_- h\left(\frac{T}{W_-}\right).$$

We want to show that this expression is ≤ 0 .

We first compare W_- and W_+ . Recall $w_k = 1/2^k$. $W_- = \sum_{k=1}^m \frac{1}{2^k} = 1 - \frac{1}{2^m}$. $W_+ = \sum_{k=m+1}^n \frac{1}{2^k} = (1 - \frac{1}{2^n}) - (1 - \frac{1}{2^m}) = \frac{1}{2^m} - \frac{1}{2^n}$. We examine the difference:

$$W_- - W_+ = 1 - \frac{1}{2^m} - \left(\frac{1}{2^m} - \frac{1}{2^n}\right) = 1 - \frac{2}{2^m} + \frac{1}{2^n} = 1 - \frac{1}{2^{m-1}} + \frac{1}{2^n}.$$

Since $1 \leq m \leq n-1$. If $m = 1$, $W_- - W_+ = 1 - 1 + \frac{1}{2^n} = \frac{1}{2^n} > 0$. If $m \geq 2$, then $1/2^{m-1} \leq 1/2$. $W_- - W_+ \geq 1 - 1/2 + 1/2^n = 1/2 + 1/2^n > 0$. In all cases, $W_- > W_+$. Now we define the function $g(W) = Wh(T/W)$ for $W > 0$. Since $T > 0$ is fixed. We want to show $g(W_+) \leq g(W_-)$. We analyze the monotonicity of $g(W)$. The derivative is:

$$g'(W) = h(T/W) + Wh'(T/W) \left(-\frac{T}{W^2}\right) = h(T/W) - \frac{T}{W} h'(T/W).$$

Let $x = T/W$. Since $T > 0$ and $W > 0$, we have $x > 0$. $g'(W) = h(x) - xh'(x)$.

Since $h(x)$ is strictly concave on $(0, \infty)$ and $h(0) = 0$, the slope of the secant line from the origin, $h(x)/x$, is strictly greater than the slope of the tangent line, $h'(x)$. Thus $h(x) > xh'(x)$. Alternatively, we can check the derivative of $k(x) = h(x) - xh'(x)$. $k'(x) = h'(x) - (h'(x) + xh''(x)) = -xh''(x)$. Since $h''(x) < 0$ for $x > 0$, $k'(x) > 0$. Since $k(0) = 0$, $k(x) > 0$ for $x > 0$. Thus, $g'(W) > 0$. The function $g(W)$ is strictly increasing for $W > 0$.

Since $0 < W_+ < W_-$ and $g(W)$ is strictly increasing, we have $g(W_+) < g(W_-)$.

$$W_+ h\left(\frac{T}{W_+}\right) < W_- h\left(\frac{T}{W_-}\right).$$

Therefore,

$$S \leq g(W_+) - g(W_-) < 0.$$

This proves that $S \leq 0$ (and the inequality is strict if not all $a_k = 1$). This is equivalent to the original inequality.

$$\frac{1}{b_1} + \frac{1}{b_2} + \cdots + \frac{1}{b_n} \geq \frac{1}{2} - \frac{1}{2^{n+1}}.$$

<ctrl100>

Subset of GPT-5-mini review

By Jensen on each side, we have:

$$S_+ := \sum_{k \in I_+} w_k h(t_k) \leq W_+ h\left(\frac{T}{W_+}\right),$$

$$S_- := \sum_{k \in I_-} w_k h(t_k) \geq W_- h\left(-\frac{T}{W_-}\right) = -W_- h\left(\frac{T}{W_-}\right).$$

Hence,

$$S \leq W_+ h\left(\frac{T}{W_+}\right) - W_- h\left(\frac{T}{W_-}\right).$$

Evaluation : Predicted Score : 7 | Human Score : 0

The issue concerns the direction of the inequalities. After applying Jensen's inequality on S_+ and S_- , we obtain two inequalities with opposite directions. Therefore, they cannot be directly combined to obtain a valid upper bound for the total sum S .

C.2 Example : GB-0254

GB-0254

We want to prove that for any positive integers a, c and any integer b , there exists a positive integer n such that $a^{F_n} + n - b$ is a multiple of c . This is equivalent to proving that the congruence $a^{F_n} + n - b \equiv 0 \pmod{c}$ has a solution for some positive integer n .

We will prove this claim by induction on c . Let $P(c)$ be the statement that for any positive integer a and any integer b , the congruence has a solution.

****Base Case:** $c = 1$ ****** For $c = 1$, the congruence is $a^{F_n} + n - b \equiv 0 \pmod{1}$. This is true for any integers a, b and any positive integer n , since any integer is congruent to 0 $\pmod{1}$. Thus, $P(1)$ is true.

****Inductive Step**** Assume that $P(k)$ is true for all positive integers $k < c$. We want to prove $P(c)$.

The congruence we need to solve is $a^{F_n} + n \equiv b \pmod{c}$.

To handle the tower of exponents in the definition of F_n , we use the Carmichael function λ . Let's define a sequence of moduli: $m_0 = c$, and for $i \geq 0$, let $m_{i+1} = \lambda(m_i)$. Since $\lambda(k) < k$ for $k > 2$ and $\lambda(k)$ is even for $k > 2$, this sequence is strictly decreasing until it reaches 1 or 2. Let K be the first index such that $m_K \in \{1, 2\}$.

Let $M = \text{lcm}(m_1, m_2, \dots, m_K)$. If $c = 2$, we can solve the congruence directly: - If a is even, for $n \geq 2$, $F_n = n^{F_{n-1}} \geq 2$ is a positive integer, so $a^{F_n} \equiv 0 \pmod{2}$. The congruence becomes $n \equiv b \pmod{2}$. We can choose any large enough n with the required parity. - If a is odd, $a^{F_n} \equiv 1 \pmod{2}$. The congruence becomes $1 + n \equiv b \pmod{2}$, or $n \equiv b - 1 \pmod{2}$. Again, we can find such an n . So, we can assume $c > 2$. In this case, $m_1 = \lambda(c)$ is even, which implies M is even.

We will show that for sufficiently large n , the sequence $n \mapsto a^{F_n} \pmod{c}$ is periodic with period M . Specifically, there is a large integer N_0 such that for all $n \geq N_0$, we have $a^{F_{n+M}} \equiv a^{F_n} \pmod{c}$. This property stems from the congruence $F_{n+M} \equiv F_n \pmod{\lambda(c)}$, which itself relies on a chain of congruences down the tower of moduli. We can choose N_0 large enough so that for all $n \geq N_0$, F_n exceeds the threshold required for the generalized Euler's totient theorem to apply for each modulus m_i . Let's prove by backward induction on i (from $i = K - 1$ down to 0) that for $n \geq N_0 + K$, $F_{n+M-i} \equiv F_{n-i} \pmod{m_{i+1}}$. - ****Base case ($i = K - 1$):**** We need $F_{n+M-K+1} \equiv F_{n-K+1} \pmod{m_K}$. Since $m_K \in \{1, 2\}$, this is trivial for $m_K = 1$. For $m_K = 2$, we check parity. For any integer $j \geq 2$, the parity of F_j is the same as the parity of j . By choosing N_0 sufficiently large, we ensure $n - K + 1 \geq 2$. As M is even, $n + M - K + 1$ has the same parity as $n - K + 1$. Thus, $F_{n+M-K+1}$ and F_{n-K+1} have the same parity. The base case holds. - ****Inductive step:**** Assume that for $i + 1$, the statement holds: $F_{n+M-(i+1)} \equiv F_{n-(i+1)} \pmod{m_{i+2}}$. We want to prove it for i . We have $F_{n+M-i} = (n + M - i)^{F_{n+M-i-1}}$ and $F_{n-i} = (n - i)^{F_{n-i-1}}$. Since m_{i+1} divides M , we have $n + M - i \equiv n - i \pmod{m_{i+1}}$. By the inductive hypothesis, $F_{n+M-i-1} \equiv F_{n-i-1} \pmod{m_{i+2}}$. So, $F_{n+M-i} \equiv (n - i)^{F_{n+M-i-1}} \pmod{m_{i+1}}$. Since $m_{i+2} = \lambda(m_{i+1})$ and F_{n-i-1} is large enough (by choice of N_0), Euler's generalized theorem implies that $(n - i)^x \equiv (n - i)^y \pmod{m_{i+1}}$ whenever $x \equiv y \pmod{\lambda(m_{i+1})}$. Thus $(n - i)^{F_{n+M-i-1}} \equiv (n - i)^{F_{n-i-1}} \pmod{m_{i+1}}$, which proves $F_{n+M-i} \equiv F_{n-i} \pmod{m_{i+1}}$. This completes the proof by backward induction. For $i = 0$, we get $F_{n+M} \equiv F_n \pmod{m_1}$, i.e., $F_{n+M} \equiv F_n \pmod{\lambda(c)}$. For $n \geq N_0$, this implies $a^{F_{n+M}} \equiv a^{F_n} \pmod{c}$.

Now we return to solving $a^{F_n} + n \equiv b \pmod{c}$. Let $d = \gcd(M, c)$. We claim that d is a proper divisor of c , i.e., $d < c$. For any prime divisor p of c , let $v_p(m)$ be the exponent of p in the prime factorization of m . We have $v_p(\lambda(c)) < v_p(c)$. This is a known property of the Carmichael function. Since $v_p(M) = \max_{i \geq 1} v_p(m_i)$ and the sequence $v_p(m_i)$ is

non-increasing, $v_p(M) = v_p(m_1) = v_p(\lambda(c)) < v_p(c)$. Then $v_p(d) = v_p(\gcd(M, c)) = \min(v_p(M), v_p(c)) = v_p(M) < v_p(c)$. Since this holds for every prime factor p of c , we must have $d < c$.

Since $d < c$, our induction hypothesis $P(d)$ is true. This means there exists a positive integer n_0 such that $a^{F_{n_0}} + n_0 \equiv b \pmod{d}$. The set of solutions modulo d is infinite, so we can choose a solution n_0 that is large enough for our periodicity argument to hold, i.e., $n_0 \geq N_0$. We now look for a solution for the modulus c of the form $n = n_0 + kM$ for some non-negative integer k . Since $k \geq 0$, we have $n \geq n_0 \geq N_0$. Thus, we can use the periodicity property: $a^{F_n} = a^{F_{n_0+kM}} \equiv a^{F_{n_0}} \pmod{c}$. Substituting this into the congruence modulo c : $a^{F_{n_0}} + (n_0 + kM) \equiv b \pmod{c}$. Rearranging the terms, we get: $(a^{F_{n_0}} + n_0 - b) + kM \equiv 0 \pmod{c}$.

Let $C_0 = a^{F_{n_0}} + n_0 - b$. By our choice of n_0 , we know that C_0 is a multiple of d . We are left with the linear congruence in the variable k : $kM \equiv -C_0 \pmod{c}$. A solution for k exists if and only if $\gcd(M, c)$ divides $-C_0$. We defined $d = \gcd(M, c)$, and we chose n_0 such that d divides C_0 . Thus, a solution for k is guaranteed to exist.

Let k_0 be any integer solution for k . We can find a non-negative solution by taking $k = k_0 + j \frac{c}{d}$ for a sufficiently large integer j . With such a non-negative k , we have $n = n_0 + kM$, which is a positive integer satisfying $a^{F_n} + n - b \equiv 0 \pmod{c}$.

This completes the induction step. By the principle of mathematical induction, the statement is true for all positive integers c .

Subset of GPT-5-mini review

2) Incorrect / unsupported assertion about valuations of $\lambda(c)$ (important)

- The proposer claims: "For any prime divisor p of c , $v_p(\lambda(c)) < v_p(c)$. This is a known property of the Carmichael function." This statement is not generally correct as written.

– Counterexample (simple): take $c = 21 = 3 \cdot 7$. Then

$$\lambda(21) = \text{lcm}(\lambda(3), \lambda(7)) = \text{lcm}(2, 6) = 6,$$

and $v_3(\lambda(21)) = 1$ while $v_3(21) = 1$; so

$$v_3(\lambda(c)) = v_3(c),$$

not strictly less.

- The proposer needs a correct argument that $\gcd(M, c) < c$. The text attempts to deduce this from $v_p(\lambda(c)) < v_p(c)$, but the inequality $v_p(\lambda(c)) < v_p(c)$ is not universally valid. Whether $\gcd(M, c) < c$ is true in fact for all $c > 1$ is a separate claim and needs proof; it cannot be taken as an immediate corollary of the incorrect valuation statement.
- In short: the step showing $d = \gcd(M, c)$ is a proper divisor of c is not established by the argument given.

Evaluation : Predicted Score : 1 | Human Score : 7

The manually annotated label in the dataset is true (7 points), while gpt-5-mini found a critical error.

The problem with the proof is the use of a general property regarding the Carmichael function (which was assumed to be "a known" property). $c = 57$ is another counterexample (the inequality is strict).