RELIABLE FINE-GRAINED EVALUATION OF NATURAL LANGUAGE MATH PROOFS

Wenjie Ma[1*] **Andrei Cojocaru**[1†] **Neel Kolhe**[1†] **Bradley Louie**[1†] **Robin Said Sharif**[1†]
Haihan Zhang[3†] **Vincent Zhuang**[2‡] **Matei Zaharia**[1] **Sewon Min**[1,4]
[1] UC Berkeley [2] Google DeepMind [3] Peking University [4] Allen Institute for AI

ABSTRACT

Recent advances in large language models (LLMs) for mathematical reasoning have largely focused on tasks with easily verifiable final answers; however, generating and verifying natural language math proofs remains an open challenge. We identify the absence of a reliable, fine-grained evaluator for LLM-generated math proofs as a critical gap. To address this, we propose a systematic methodology for developing and validating evaluators that assign fine-grained scores on a 0–7 scale to model-generated math proofs. To enable this study, we introduce **PROOFBENCH**, the first expert-annotated dataset of fine-grained proof ratings, spanning 145 problems from six major math competitions (USAMO, IMO, Putnam, etc) and 435 LLM-generated solutions from Gemini-2.5-pro, o3, and DeepSeek-R1. Using PROOFBENCH as a testbed, we systematically explore the evaluator design space across key axes: the backbone model, input context, instructions and evaluation workflow. Our analysis delivers **PROOFGRADER**, an evaluator that combines a strong reasoning backbone LM, rich context from reference solutions and marking schemes, and a simple ensembling method; it achieves a low Mean Absolute Error (MAE) of 0.926 against expert scores, significantly outperforming naive baselines. Finally, we demonstrate its practical utility in a best-of-n selection task: at $n = 16$, PROOFGRADER achieves an average score of 4.14/7, closing 78% of the gap between a naive binary evaluator (2.48) and the human oracle (4.62), highlighting its potential to advance downstream proof generation.

1 INTRODUCTION

USAMO 2025 P1. Fix positive integers k and d. *Prove* that for all sufficiently large odd positive integers n, the digits of the base-$2n$ representation of n^k are all greater than d.	**IMO 2024 P2.** Determine all positive integers a and b such that there exists a positive integer g with $\gcd(a^n + b, b^n + a) = g$ for all sufficiently large n.

Figure 1: Two example proof problems selected from well-established competitions.

Large language models (LLMs) have recently achieved remarkable progress in mathematical reasoning, attaining strong performance on a variety of benchmarks. Such models are especially strong at solving final-answer problems because they can be trained using reinforcement learning against simple answer verifiable rewards (Shao et al., 2024; DeepSeek-AI et al., 2025; Yang et al., 2025; Yu et al., 2025; Wang et al., 2025). However, these methods do not transfer to proof generation for two reasons: (i) many proof problems do not admit a single, easily checkable final answer (Figure 1 left); and (ii) even when a final answer exists (Figure 1 right), verifying it is insufficient to assess proof validity, as the reasoning may contain substantial intermediate errors (Petrov et al., 2025; Dekoninck et al., 2025). Because proof-generation tasks constitute a large share of mathematical problem solving in research and education, this necessitates reliable proof evaluation methods.

We identify **the absence of a reliable proof evaluator** as a key bottleneck for improving proof generation, which is essential for providing faithful assessments of model capabilities and accurate

*Correspondence: `windsey@berkeley.edu`
†Dataset contributors.
‡Served in an advisory capacity only.

reward signals for training models. Expert grading, while accurate, is slow and costly. While formal math (e.g., Lean) offers absolute certainty, it remains detached from the natural language used in most human mathematics education and research; furthermore, automatically translating natural-language proofs into formal languages is brittle and remains extremely challenging (Gao et al., 2025; Liu et al., 2025a). Our work therefore focuses on the critical and complementary task of evaluating proofs in their natural representation. While LLM-as-a-judge (Zheng et al., 2023; Sheng et al., 2025; Dekoninck et al., 2025; Huang & Yang, 2025) is promising, its application to math proofs is unsettled, and outcomes are sensitive to evaluator design—model choice, available context, rubric construction, and prompting—all of which are poorly understood.

This paper tackles this bottleneck by developing a methodology to create high-fidelity, fine-grained evaluators, and demonstrates for the first time that they can nearly match human oracle performance in downstream tasks. Our objective is to design an automated evaluator, $\mathcal{E}$, that takes a problem p, a model-generated solution s, and an optional set of contextual materials $\mathcal{C}_{\text{context}}$ to produce a fine-grained integer score, $\hat{y} \in \{0, 1, \ldots, 7\}$. We identify the optimal evaluator not through model training, but by searching over a discrete space of configurations $\mathbb{C}$. The goal is to find the best configuration c^* that minimizes an empirical loss function L on our dataset:

$$c^* = \arg\min_{c \in \mathbb{C}} \frac{1}{|D_{test}|} \sum_{(p,s,y) \in D_{test}} L(\mathcal{E}_c(p, s, \mathcal{C}_{\text{context}}), y),$$

where y represents the ground-truth score assigned by human experts. We adopt the 0–7 scoring scale used in premier competitions to enable nuanced assessment beyond binary correctness.

First, to support our study, we introduce **ProofBench**, the first expert-annotated dataset for fine-grained proof evaluation that spans problems from multiple contests and years. It spans 145 problems from EGMO, USAMO, IMO, USA TST, APMO, and Putnam (2022–2025), with 435 LLM-generated solutions from state-of-the-art models (Gemini-2.5-Pro, o3, DeepSeek-R1). Data annotation follows a two-stage process: (i) generate a problem-specific marking scheme to standardize criteria while allowing valid alternative approaches; (ii) have experts score each solution with that scheme while allowing for valid alternative solutions.

Using ProofBench as a testbed, we systematically explore the evaluator design space, including four primary axes: backbone models, input context (such as reference solutions and problem-specific marking schemes), instruction sets and evaluation workflows. We consider single-pass evaluators as well as more advanced techniques, including (1) ensembling multiple evaluation runs and (2) staged workflows that decompose the complex evaluation task into focused, sequential steps. Our analysis delivers **ProofGrader**, an LLM-based evaluator that combines a strong backbone LM with informative context (both reference solutions and a marking scheme) and simple ensembling that is surprisingly effective; it achieves a low Mean Absolute Error (MAE) of 0.926 against expert scores, significantly outperforming naive baselines.

Finally, we validate the evaluators' practical utility in a downstream best-of-n selection task, a standard proxy for assessing its potential as a reward signal (Gao et al., 2022; Frick et al., 2024; Malik et al., 2025). At $n = 16$, ProofGrader achieves an average score of 4.14/7, closing 78% of the performance gap between a naive binary evaluator (2.48) and the human oracle (4.62). It also outperforms computationally intensive, pairwise selection methods such as Knockout tournament selection (Liu et al., 2025b). These results highlight ProofGrader's promise as a reward model for advancing proof generation.

To summarize, our contributions are three-fold:

- We introduce ProofBench[1], an expert-graded math proof dataset consisting of problems from well-established contests (USAMO, IMO, Putnam, etc) and solutions from state-of-the-art reasoning models (o3, gemini-2.5-pro and DeepSeek-R1) (§2).
- We conduct a systematic study of key evaluator design factors and introduce ProofGrader, our best-performing evaluator that combines a strong reasoning LM, a problem-specific marking scheme, and simple ensembling, achieving a strong alignment with expert ratings (§3).
- We show that ProofGrader is on par with experts in a best-of-n selection task, improving a score from 2.48 (binary evaluator) to 4.14, highlighting its promise as a reward model (§4).

[1]The full dataset is available at `huggingface.co/datasets/wenjiema02/ProofBench`.

Problem. Let $n > k \geq 1$ be integers. Let $P(x) \in \mathbb{R}[x]$ be a polynomial of degree n with no repeated roots and $P(0) \neq 0$. Suppose that for any real numbers $a_0, \ldots, a_k$ such that $a_k x^k + \cdots + a_1 x + a_0$ divides $P(x)$, the product $a_0 a_1 \ldots a_k$ is zero. *Prove* that $P(x)$ has a nonreal root.

Reference Solution. By considering any $k+1$ roots of P, WLOG assume $n = k+1$. Suppose $P(x) = (x+r_1)\ldots(x+r_n)$ has $P(0) \neq 0$. Then each polynomial $P_i(x) = P(x)/(x+r_i)$ of degree $n-1$ has ≥ 1 zero coefficient.
The leading and constant coefficients of each P_i are nonzero, leaving $n-2$ other coefficients. By pigeonhole, P_1 and P_2 share a zero coefficient position, say x^k for some $1 \leq k < n-1$.

Claim. If P_1 and P_2 both have x^k coefficient zero, then $Q(x) = (x+r_3)\ldots(x+r_n)$ has consecutive zero coefficients $b_k = b_{k-1} = 0$.

Proof. By Vieta, let $Q(x) = x^{n-2} + b_{n-3}x^{n-3} + \cdots + b_0$. The x^k coefficient of P_1, P_2 being zero means $r_1 b_k + b_{k-1} = r_2 b_k + b_{k-1} = 0$,
hence $b_k = b_{k-1} = 0$ (using r_i nonzero, distinct). □

Lemma. If $F(x) \in \mathbb{R}[x]$ has two consecutive zero coefficients, it cannot have all distinct real roots.

Proof 1 (Rolle). Say x^t, x^{t+1} coefficients are zero. $\cdots$ But $F^{(t)}(x)$ has a double root at 0, contradiction. □

Proof 2 (Descartes). Real roots are bounded by sign changes in $F(x)$ plus sign changes in $F(-x)$. $\cdots$
With b nonzero coefficients and z runs of zeros, real roots $\leq b - 1 + z \leq \deg F$. Two consecutive zeros make this strict. □

Marking Scheme (max 7 pts). *Checkpoints (additive):*
(1) [1pt] Problem reduction to $n = k+1$ and setup. Alt: complete proof for $k = 1$.
(2) [2pts] Pigeonhole: n polynomials, $n-2$ internal coefficient positions; two share a zero position.
(3) [2pts] Deduce consecutive zeros: from $[x^m]P_1 = [x^m]P_2 = 0$ and $P_i = (x+r_i)Q$, show $b_m = b_{m-1} = 0$.
(4) [2pts] Prove lemma (2pts for complete Rolle or Descartes proof; 1pt for partial).
Deductions: Cap 6/7 if no reduction justification; cap 5/7 if lemma flawed; cap 3/7 if stops after PHP; -1pt for minor gaps (e.g., not using $r_1 \neq r_2$).
Zero credit: Unjustified WLOG; merely stating theorems; specific examples only; noting $P(0) \neq 0 \Rightarrow$ nonzero roots.

Figure 2: Example problem (USAMO 2025 P2) with its reference solution and marking scheme.

2 PROOFBENCH: EXPERT-RATED MATH PROOF SOLUTIONS

We assess proof evaluators for *human alignment* on a 0–7 scale, which requires fine-grained expert annotations on model-generated proofs across diverse sources and years. Prior work such as MATH-ARENA focuses on a small set of contests like USAMO 2025 (Petrov et al., 2025); other studies of evaluators are limited to binary judgments of proof correctness (Dekoninck et al., 2025). We thus construct our own dataset, PROOFBENCH, consisting of 145 proof problems from six major math competitions across four years, with model-generated solutions by the state-of-the-art reasoning models: o3, Gemini-2.5-Pro, and DeepSeek-R1. This section describes the data collection, annotation process (§2.1), and dataset statistics (§2.2).

Why 0–7 Scale? The 0–7 scale aligns the dataset with the established grading standards of premier mathematics competitions[2], the source of our benchmark problems. Moreover, its fine-grained nature is critical for a nuanced assessment of proof quality, which is not captured by the binary "correct"/"incorrect" grading, as we show throughout §3 and §4.

2.1 DATASET CONSTRUCTION

Problem Sources. We collected 145 problems from the official websites of prestigious mathematics competitions, including the APMO, EGMO, IMO, PUTNAM, USA TST, and USAMO, spanning the years 2022–2025. Distribution of contests is shown in Figure 3a. Sourcing directly from official materials ensures the reliability of problem statements and ground-truth solutions, avoiding transcription errors common in secondary sources. Problems were parsed from official PDFs and normalized, and all available human solutions were included. The final collected data consists of a set of `(problem, reference solution(s), metadata)` triples, where metadata includes the source competition and year.

[2]The official Putnam Competition uses a 0–10 scale. For consistency across competitions, we normalize all expert annotations to a unified 0–7 scale.

Proof Generation. For each problem, we generate a proof from models (*generators*) using a standardized prompt (§A.7) that asks for a complete, self-contained proof. We generate proofs from three state-of-the-art reasoning models: OpenAI o3, Gemini-2.5-Pro, and DeepSeek-R1-0528, which span both proprietary and open-source families.

Expert Annotations. Finally, annotation proceeds in two stages: marking scheme generation and model-generated proof grading. All annotations were conducted by a team of five experts with Putnam-level or national Math Olympiad experience, using a carefully designed web interface.

Step 1: Automated Marking Scheme Generation: A central challenge in creating this benchmark was ensuring consistent and scalable expert grading. While human-written rubrics are the gold standard, automated generation offers superior scalability and a systematic way to recognize key steps from provided reference solutions. In our method, the marking scheme is produced by an LLM $\mathcal{M}_{\text{MS}}$. Given the problem x and reference solutions $\mathcal{S}$, $\mathcal{M}_{\text{MS}}$ is prompted to output (i) a list of conditions under which scores are awarded or deducted, and (ii) a list of trivial cases that should not be awarded any points. This generator was developed through a rigorous refinement process where experts also graded the quality of the generated marking schemes, providing feedback that led to the selection of gemini-2.5-pro as the final $\mathcal{M}_{\text{MS}}$ (see §A.2 for details). An example of a generated marking scheme together with problem text and reference solution is shown in Figure 2. From our expert evaluations, approximately 85% of the generated marking schemes in our final dataset were judged to be reasonable and of high quality.

Step 2: Proof Grading: With a problem-specific marking scheme, an expert annotator scores a given model-generated proof. Experts were instructed to treat the marking scheme as a detailed reference for the expected solution path, rather than a rigid checklist. This is particularly important for fairly evaluating proofs that employed a novel method different from the provided ground-truth solutions, ensuring that valid alternative reasoning paths were credited appropriately. The experts will assign a score on a 0–7 scale. Before large-scale annotation, our experts underwent a calibration phase to align their judgments and finalize rules for handling edge cases, a critical step for minimizing inconsistencies. We assign 40% of the solutions to more than one annotator and ensure their agreement is 90% or higher.

2.2 Dataset Statistics

In total, ProofBench consists of 145 proof problems and 435 expert-annotated evaluations of proofs generated by OpenAI o3, Gemini-2.5-Pro, and DeepSeek-R1-0528. Figure 3 summarizes data statistics across competitions (Figure 3a), scores (Figure 3b), and generators (Figure 3c).

Our analysis reveals several key findings about state-of-the-art reasoning models:

- Current models remain far from achieving high scores: even the strongest models obtain scores of 6 or higher on fewer than 30% of problems, as shown in Figure 3d.
- OpenAI-o3 performs best overall, with a mean score of 2.87, and is closely followed by Gemini-2.5-Pro, according to Figure 3c.
- Performance varies considerably across competitions (Figure 3e,f): all models perform best on Putnam, where the mean score is 3.11, but struggle on TST, where the mean drops to 1.28.

3 A Systematic Study of Evaluator Designs

Using ProofBench, we systematically study the key design factors for a math proof evaluator that produces fine-grained scores (0–7) for model-generated solutions. We demonstrate that (i) a strong reasoning backbone with informative context (like marking schemes) yields substantial gains, (ii) simple ensembling further improves accuracy and robustness, and (iii) a *staged* evaluation pipeline can refine weaker model backbones.

3.1 Evaluator Designs

Given the task of scoring model-generated solutions, a naive evaluator would simply take a problem and a generic instruction and assign a score. Instead, we systematically investigate design choices

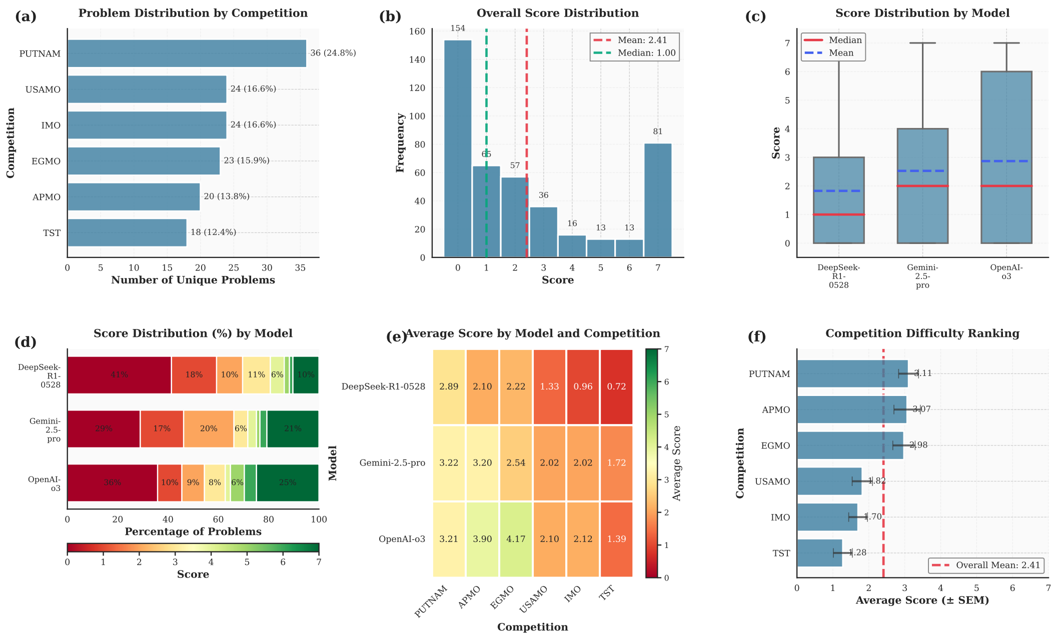


Figure 3: **Data statistics and model evaluation results.** (a) Problem statistics across six competitions. (b) Score distribution showing mean (2.41) and median (1.00). (c) Model performance comparison via box plots. Each box represents the interquartile range (IQR, middle 50% of scores), with the red line showing the median and blue dashed line showing the mean. (d) Per-generator score distribution. Stacked horizontal bar chart showing the percentage of problems receiving each score (0-7) for each model. Colors indicate score values (red=low, green=high). (e) Performance heatmap by generator and competition. (f) Competition difficulty ranking with error bars: TST is the most challenging competition (mean: 1.28) and Putnam is the easiest (mean: 3.11).

that enhance evaluation quality beyond this baseline. We first analyze single-pass evaluators along four key dimensions: (i) backbone model, (ii) contextual input, (iii) instruction set, and (iv) workflow design. We then extend this design to ensemble-based and multi-stage evaluation workflows.

Single-Pass Methods. A single-pass evaluator prompts a backbone model $\mathcal{M}$ (which may or may not be the same as $\mathcal{M}_{\text{MS}}$) to grade a solution s in one step. We analyze single-pass evaluator along three dimensions: the backbone LM, the context provided and the instructions given.

- **Backbone Model Choice.** This refers to the LLM that is prompted to execute the evaluation. We compare six models that span different model families, size and reasoning capabilities: O3, GPT-5 (GPT-5-Thinking), GEMINI (gemini-2.5-pro), O4-MINI, R1 (DeepSeek-R1-0528), and GPT-4O.
- **Context.** We consider four different context configurations, including: providing both the pre-generated marking scheme and the reference solution(s) (REF+MS); providing only the marking scheme (MS); providing only the reference solution(s) (REF); and a naive baseline where only basic grading instructions without any problem-specific context are provided (NONE).
- **Instruction.** The instruction set refers to the specific prompt that guides the evaluator on how to interpret and apply the provided context information and perform the task. In our study, for the most informative context setting, REF+MS, we compared three types of instructions to understand how guidance on using the context affects performance. These include NORM (Normal), a flexible instruction that directs the model to follow the marking scheme but allows it to map valid alternative approaches to equivalent checkpoints; STRICT, a rigid instruction that requires the model to adhere strictly to the provided marking scheme and penalize any deviation; and BASIC, an instruction with minimal guidance on how to use the provided materials.

We then further study whether extending single-pass evaluators, e.g., through ensembling or multi-stage evaluation workflows, improves performance.

Model	Context	RMSE ↓	MAE ↓	WTA$_{\leq 1}$ (%) ↑	Kendall-τ ↑	Bias ≈	Quality
O3	REF+MS	**1.273** ± 0.16	**0.964** ± 0.12	**76.5** ± 4.3	**0.502**	**-0.008**	Excellent
	MS	1.418 ± 0.17	1.069 ± 0.14	72.8 ± 4.3	0.477	-0.381	Good
	REF	1.575 ± 0.14	1.330 ± 0.12	65.3 ± 4.6	0.481	0.478	Good
	NONE	1.901 ± 0.15	1.680 ± 0.15	49.5 ± 5.8	0.435	0.924	Fair
GEMINI	REF+MS	1.696 ± 0.17	1.342 ± 0.15	62.7 ± 4.9	**0.529**	0.626	Good
	MS	**1.502** ± 0.17	**1.142** ± 0.14	**70.0** ± 4.2	0.488	**0.151**	Good
	REF	2.177 ± 0.14	1.910 ± 0.15	39.9 ± 5.2	0.410	1.285	Poor
	NONE	2.397 ± 0.15	2.107 ± 0.16	36.6 ± 4.9	0.319	1.496	Poor
O4-MINI	REF+MS	1.816 ± 0.22	1.367 ± 0.19	67.6 ± 4.6	0.476	0.762	Good
	MS	**1.636** ± 0.22	**1.234** ± 0.18	**69.5** ± 4.8	**0.505**	**0.309**	Good
	REF	1.858 ± 0.19	1.504 ± 0.16	61.3 ± 4.8	0.465	0.950	Fair
	NONE	2.276 ± 0.23	1.914 ± 0.21	49.5 ± 5.3	0.430	1.569	Fair
GPT-5	REF+MS	1.353 ± 0.16	1.055 ± 0.13	73.2 ± 4.6	0.532	0.295	Excellent
	MS	**1.350** ± 0.17	**1.018** ± 0.14	**74.9** ± 4.2	**0.536**	**-0.175**	Excellent
	REF	1.617 ± 0.13	1.400 ± 0.12	57.0 ± 5.4	0.501	0.799	Good
	NONE	1.919 ± 0.14	1.708 ± 0.14	46.5 ± 5.4	0.404	1.034	Fair
R1	REF+MS	1.735 ± 0.22	1.357 ± 0.18	66.4 ± 5.2	0.429	0.732	Good
	MS	**1.682** ± 0.22	**1.298** ± 0.18	**68.5** ± 4.9	**0.450**	**0.422**	Good
	REF	3.187 ± 0.23	2.736 ± 0.22	30.3 ± 5.0	0.289	2.450	Bad
	NONE	3.273 ± 0.25	2.842 ± 0.25	33.5 ± 4.9	0.102	2.581	Bad
GPT-4O	REF+MS	2.599 ± 0.20	2.197 ± 0.20	39.7 ± 5.0	**0.479**	1.824	Poor
	MS	**2.245** ± 0.21	**1.827** ± 0.19	**50.4** ± 5.3	0.377	**1.001**	Fair
	REF	2.726 ± 0.19	2.371 ± 0.18	36.0 ± 4.9	0.343	1.887	Poor
	NONE	3.402 ± 0.26	3.001 ± 0.26	31.9 ± 4.9	0.208	2.614	Bad

Table 1: **Performance comparison of six LLMs on 0-7 scale evaluation tasks under different context designs.** Contexts: REF+MS (reference solution + marking scheme), MS (marking scheme only), REF (reference solution only), NONE (no context). Values shown as mean ± 95% confidence interval (CI) margin of error. Best values per model in **bold**; best ( ) and worst ( ) context highlighted. Arrows: ↓ lower better, ↑ higher better, ≈ closer to zero better. Quality: Excellent, Good, Fair, Poor, Bad.

Ensemble. We consider a simple ensembling technique, which runs the same evaluator independently multiple times and combines the individual ratings with an aggregation operator, such as the mean or median. This technique is expected to produce a more stable final score, reducing the variance of single-pass evaluators.

Staged Evaluation. We consider staged workflows that decompose the complex task of evaluation into a sequence of more focused reasoning steps. We implement two approaches. (1) **Binary & Errors → Fine-Grained** first predicts the binary correctness of the proof and identifies key errors, and then makes a second pass that uses this output to generate a more calibrated, 0–7 score. (2) **Evaluation→Reflection→Verdict** uses an iterative refinement process inspired by self-correction mechanisms (Guo et al., 2025). The LLM first performs a complete initial evaluation. It is then prompted to reflect on and critique its own assessment. Finally, it uses the original context, its initial judgment, and its self-critique to produce a final score.

3.2 Evaluation Metrics

We evaluate on the 0–7 scale using per-problem metrics. All problems have the same number of responses (n). Let P be the number of problems. For each problem $p = 1, \dots, P$ with expert scores y_{pi} and evaluator outputs $\hat{y}_{pi}$, we compute

$$\mathrm{MAE}_p = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}_{pi} - y_{pi}\right|, \quad \mathrm{RMSE}_p = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_{pi} - y_{pi}\right)^2},$$

$$\mathrm{Bias}_p = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_{pi} - y_{pi}\right), \quad \mathrm{WTA}_p(\leq 1) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{\,|\hat{y}_{pi} - y_{pi}| \leq 1\,\}.$$

MAE and RMSE both measure errors relative to expert scores; MAE averages absolute deviations, while RMSE takes the square root of mean squared deviations and therefore penalizes large mistakes more (lower is better). WTA(≤ 1) measures the fraction of predictions that land within one point of the expert score. Bias measures the average signed error (systematic shift), with positive values indicating over-scoring and negative values indicating under-scoring.

For ranking agreement within a problem, we use Kendall's τ_b (ties-adjusted). Detailed definitions of Kendall's τ_b as well as aggregation formulas are provided in §A.3.

3.3 What Factors Improve Single-pass Evaluator?

Effects of backbone model and contextual information. Table 1 shows the results of single-pass evaluators across different backbone models and context settings. First, *the strength of the backbone model* strongly correlates with performance: moving from weaker to stronger models brings better calibrations (o3 leads in nearly all metrics). Second, *contextual information* consistently improves all metrics for every backbone, with the marking scheme (MS) contributing the majority of the gain relative to the reference solution alone (Ref); combining both (Ref+MS) provides a small additional improvement primarily for the strongest backbone (o3).

Model	Instruction	RMSE ↓	MAE ↓	WTA$_{\leq 1}$ (%) ↑	kendall-τ ↑	Bias ≈
o3	Norm	**1.273**	**0.964**	**76.5**	**0.502**	**−0.008**
	Strict	1.420	1.095	72.8	0.457	−0.304
	Basic	1.348	1.039	73.0	0.501	0.165
o4-mini	Norm	1.816	1.367	67.6	**0.476**	0.762
	Strict	**1.718**	**1.266**	**69.2**	0.443	**0.396**
	Basic	1.817	1.360	67.6	0.437	0.724
Gemini	Norm	1.696	1.342	62.7	**0.529**	0.626
	Strict	**1.581**	**1.231**	**67.4**	0.475	**0.316**
	Basic	1.773	1.424	61.7	0.469	0.832

Table 2: **Instruction ablation under Ref+MS.** Three instruction styles guide how the evaluator uses context: Norm (flexible use of the marking scheme, allowing valid alternatives), Strict (literal adherence with penalties for deviations), and Basic (minimal guidance). For *Bias*, the closer to 0, the better; for other metrics, ↓ means the lower the better while ↑ means the opposite.

Instruction style under Ref+MS should match the backbone. With context fixed at Ref+MS, instruction choice modulates the calibration-ranking trade-off (Table 2). The strongest backbone (o3) attains the best overall accuracy and near-zero bias with the more flexible Norm prompt, whereas mid-tier models (e.g., Gemini, o4-mini) benefit from the more prescriptive Strict prompt. This pattern suggests that stronger models can reliably and flexibly apply the marking scheme (e.g., mapping alternative derivations to equivalent checkpoints), while mid-tier models require more prescriptive guidance to reduce over-crediting and variance.

Per-generator results. A natural question that can arise is the potential bias when evaluators assess outputs generated by their own models: are they more accurate because of familiarity, or less so because they overestimate their own generations? To investigate this, we evaluate three single-pass evaluators, each using the same reference solutions and marking scheme, on per-generator splits, i.e., response sets grouped by their source generator (R1, Gemini, o3). The results in Table 3 are mixed. On one hand, the strongest evaluator remains consistent across all generators: the o3-based evaluator performs best across all of them. On the other hand, each evaluator exhibits its highest MAE on outputs produced by its own model family, indicating a tendency toward

Evaluator	MAE ↓ by Generator Outputs		
	R1	Gemini	o3
o3	**0.993**	**0.778**	**1.120**
Gemini	1.183	1.616	1.225
R1	1.504	1.432	1.181

Table 3: Per-generator evaluation accuracy (MAE; lower is better). **Bold** marks, for each *generator* (column), the *best* (smallest) MAE across generators.

within-generator underperformance. Per-generator significance tests and additional breakdowns are provided in §A.4.

Summary

We observe effect sizes ordered as *backbone* $\gg$ *context* $\gg$ *instruction*. Consequently, the recommended configuration is: employ the strongest available backbone; include a marking scheme by default and align the rigidity of the instruction with the model's capacity.

3.4 Do Ensembling and Staged Workflows Improve Single-pass Evaluation?

We next extend single-pass evaluators through either ensembling or multi-stage pipelines, as described in §3.1.

Model	**Run**	RMSE $\downarrow$	MAE $\downarrow$	WTA$_{\leq 1}$(%)$\uparrow$	kendall-τ $\uparrow$	Bias $\approx$
O3	Single (mean±std; n=5)	1.265 (±0.039)	0.981 (±0.028)	75.5 (±1.126)	0.509 (±0.022)	0.018 (±0.020)
	Best single	1.225	0.944	77.1	0.540	0.009
	Mean	**1.169**	0.940	69.7	**0.578**	0.018
	Median	1.185	**0.926**	**77.7**	0.540	0.008
	Majority	1.186	**0.926**	75.6	0.523	**0.004**

Table 4: **Ensembling over multiple runs boosts performance and reduces variance for the O3 evaluator.** We compare five individual runs against three aggregation strategies. Both mean and median aggregation achieve a lower RMSE than the best single run. Mean aggregation is optimal for RMSE and ranking correlation (Kendall-τ), while median aggregation excels on MAE and WTA$_{\leq 1}$.

Ensembling helps. For ensembling, we aggregate five independent O3 evaluation runs under Ref+MS, with results reported in Table 4. Compared to the best single run, averaging all runs reduces RMSE by $\sim$0.06 (1.225$\rightarrow$1.169) and improves rank agreement (Kendall-τ from 0.540 to 0.578). Median aggregation delivers the lowest MAE and highest WTA$_{\leq 1}$, and Majority matches that MAE while giving the smallest bias. Bolded cells in Table 4 mark these optima.

Although the absolute performance gains may appear modest, the key advantage of ensembling is variance reduction: single-pass evaluators, even with rich context, can exhibit high variance, whereas ensembling substantially mitigates it.

Staged pipelines *may* improve weaker models, but not stronger ones. For staged pipelines, Tables 5 and 6 present results for two designs: Two-step (Binary+Errors $\rightarrow$ Fine-Grained) and Three-stage (Evaluate $\rightarrow$ Reflect $\rightarrow$ Verdict), respectively. For both, we include O3- and O4-MINI-based evaluators.

Both staged designs yield modest gains and often none. For instance, the two-step pipeline (Table 5) improves performance on the O4-MINI backbone (e.g., reducing RMSE from 1.816 to 1.650 and MAE from 1.367 to 1.245) but hurts the strong O3 backbone (e.g., increasing RMSE from 1.273 to 1.375 and MAE from 0.964 to 1.065). Reflection offers only small improvements, and the final verdict adds no value (Table 6).

Putting everything together: ProofGrader. Based on our findings, we introduce ProofGrader, the evaluator that is best-performing most consistently across all settings. ProofGrader builds on the O3 model with the Ref+MS configuration and the Norm instruction set, and incorporates simple ensembling techniques.

3.5 In-depth Analysis

Why are reference solutions and marking schemes helpful? Comparing evaluators based on O3 with None to those given a Ref+MS, we find that in over 60% of cases, the no-context evaluator *overestimates* the generation's correctness, i.e., it incorrectly assigns a *higher* score to the generation.

Model	Design	RMSE↓	MAE↓	WTA$_{\leq 1}$ (%)↑	Kendall-τ ↑	Bias≈
O3	**Single-pass (REF+MS)**	$\mathbf{1.273}$	$\mathbf{0.964}$	$\mathbf{76.5}$	$\mathbf{0.502}$	$\mathbf{-0.008}$
	Binary+Errors → Fine-Grained	1.375	1.065	73.0	0.444	−0.102
O4-MINI	Single-pass (REF+MS)	1.816	1.367	$\mathbf{67.6}$	0.476	0.762
	Binary+Errors → Fine-Grained	$\mathbf{1.650}$	$\mathbf{1.245}$	66.7	$\mathbf{0.503}$	$\mathbf{0.320}$
O3+O4-MINI	Binary+Errors → Fine-Grained	1.513	1.126	71.4	0.458	−0.095

Table 5: Comparison of two-step **Binary+Errors** → **Fine-Grained** pipeline versus single-pass **(REF+MS)** evaluation across backbones. Staged design boosts mid-tier O4-MINI but degrades strong O3 performance; bold marks the best within each model block.

Backbone	Stage	RMSE↓	MAE↓	WTA$_{\leq 1}$ (%)↑	Kendall-τ ↑	Bias≈
O3	Initial (A)	1.746	1.107	**73.0**	0.525	0.058
	Reflection (B)	**1.715**	**1.055**	71.8	**0.538**	**-0.060**
	Verdict (C)	1.756	1.058	71.8	0.478	−0.231
O4-MINI	Initial (A)	**2.313**	**1.400**	**67.4**	0.388	**0.701**
	Reflection (B)	2.362	1.433	65.5	0.423	0.858
	Verdict (C)	2.352	1.424	66.2	**0.448**	0.823

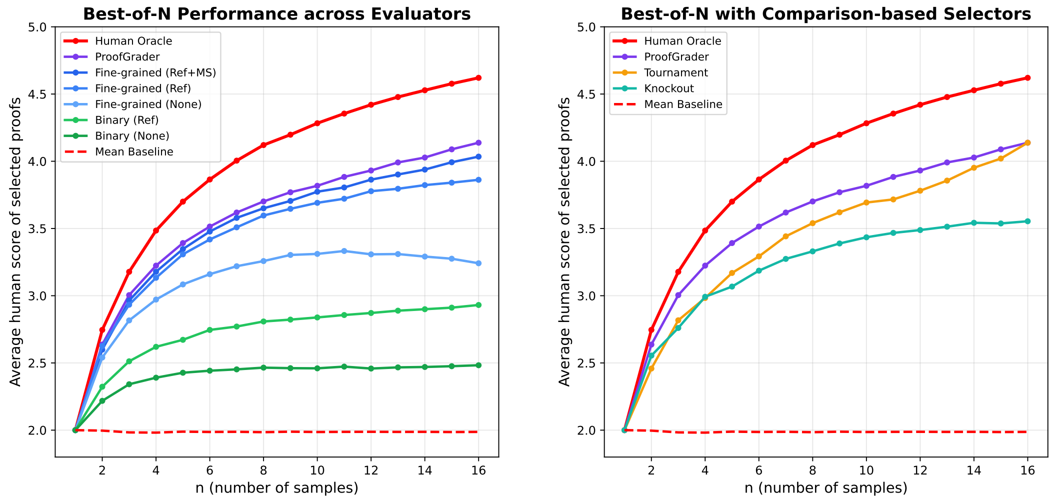
Table 6: Performance across stages of the three-step **Evaluate** → **Reflect** → **Verdict** pipeline, for O3 and O4-MINI backbones. Reflection yields marginal gains over initial evaluation, but final verdict adds little value; bold marks the best within each backbone block.

Crucially, this overestimation is not uniform: we observe a strong correlation ($r = 0.699$, $p < 0.001$) between proof quality and evaluation gap. For *low-quality* proofs (scored 0–2 by the context-aware evaluator), the no-context evaluator over-scores by an average of 1.7 points. In stark contrast, for *high-quality* proofs (scored 5–7), the context-aware evaluator assigns *higher* scores by +0.8 points on average. This asymmetric pattern supports our hypothesis that, without reference solutions or marking schemes, the evaluator struggles to gauge proof *progress*—how much has been correctly established and how far the argument is from completion—and thus is prone to rewarding fluent but irrelevant or unhelpful content.

This effect is exacerbated when the evaluator cannot fully solve the problem—a case that constitutes the majority of our dataset, as shown in Figure 3. When we examine problems that o3 (the evaluator model) cannot solve well (scoring ≤ 2 when acting as a prover), we find that the no-context evaluator over-scores by 1.40 points on average. In contrast, for problems that o3 can solve (scoring ≥ 5), the context-aware evaluator actually scores *higher* by 0.90 points. This 2.3-point swing ($t = -8.790$, $p < 0.001$) demonstrates that the evaluator's difficulty in solving a problem directly impairs its ability to assess partial progress on that problem without external guidance.

Case study. On Putnam-2022-B5 (generator: O3), the evaluator with context identified a critical logical flaw ("log-concavity inequality fails when $|a_i| > 1$") and assigned 1 point, while the no-context evaluator deemed the proof "essentially correct" and awarded 7 points. Manual inspection of the 20 cases with largest disagreements ($|\Delta| \geq 4$) reveals that 80% involve proofs with sophisticated presentation but fundamental logical errors, supporting the view that reference solutions help evaluators distinguish between fluent exposition and mathematical correctness.

What if the evaluator uses a different marking scheme from human experts? In our main setup, evaluators use the same marking schemes as the human graders. To test sensitivity to the rubric, we replace these with two alternatives: (i) regenerate a marking scheme using the *same* prompt/model (new sample), and (ii) generate a marking scheme with a *different* model. We then evaluate the dataset with O3 given the reference solution plus each alternative scheme. In both cases, performance degrades relative to using the original marking scheme across different metrics, suggesting that evaluator accuracy depends on close alignment with the marking scheme used by human although experts do not strictly follow them. Detailed results appear in §A.4.



(a) **Evaluators as BoN selectors.** Ensemble-based fine-grained evaluator closely track the Human-Oracle curve, while the binary evaluators (green lines) perform much worse.

(b) **Comparison-based selectors.** Our fine-grained evaluator performs the best. Some strategies show signs of performance degradation at larger values of n, while our approach remains robust.

Figure 4: **Best-of-n with different *selector*s.** Average best-of-n score over 29 problems for O3 (generator) as n increases from 1 to 16. PROOFGRADER is our ensemble evaluator: the median over five independent O3 runs using the reference solution and marking scheme.

4 DOWNSTREAM UTILITY: BEST-OF-N PROOF SELECTION

A key property of a good reward model is that it can reliably identify the best response in a given batch. To understand how this ability correlates with the calibration metrics in Section 3.2, we measure the best-of-n (BoN) performance of various evaluators. BoN is conditioned on the generator model and the sampling budget (n), directly reflecting the operations that drive model improvement cycles. It can reveal issues like over-optimization to a flawed evaluator and yields a practical utility curve as n varies. In general, a higher BoN score means the evaluator more reliably selects high-quality candidates for subsequent training, guiding real downstream gains. The applications include selecting high-quality training data via rejection sampling (e.g., distillation) (Nakano et al., 2022; Touvron et al., 2023) and providing a reward signal for Reinforcement Learning (RL) (DeepSeek-AI et al., 2025; Yu et al., 2025).

In summary, this section answers a critical question: Does the superiority of our fine-grained evaluators, as measured by offline metrics, translate to an improvement in a downstream selection task?

Setup. To create a testbed, we use O3 to generate 16 candidate proofs for each of 29 selected problems from 2025, resulting in 464 unique proofs. All 464 of these candidates were then scored by three human experts using the pipeline described in §2. This dataset is distinct from the section 3 split and serves a different purpose. For each problem, 8 responses are graded by two experts with disagreements resolved through discussion. For this analysis, we test a selection of the fine-grained evaluators studied previously (§4.1) and several comparison-based selection strategies including Knockout tournament style selection (§4.2). The performance of each selection method is measured using a **best-of-n (BoN) curve**. Since an exhaustive evaluation over all $\binom{16}{n}$ subsets is computationally intensive, we estimate the BoN curve using Monte Carlo subsampling. For each $n \in \{1, 2, \cdots, 16\}$, we sample a large number of subsets of size n without replacement. For each subset, the evaluator selects the single proof with the highest assigned score, using the initial ordering to break ties. The performance at n is the average human score of these selected proofs, aggregated over all subsamples and all 29 problems. We also include two key baselines: an oracle baseline and a mean baseline. The HUMAN ORACLE represents the performance ceiling, calculated as the average score of the best possible selection among the first n candidates. The MEAN

BASELINE is calculated as the cumulative average score of the first n candidates. All evaluators and selectors use O3 as the backbone model.

4.1 THE VALUE OF FINE-GRAINED SCORING

Results are shown in Figure 4a, addressing three key questions.

How close is PROOFGRADER to the human oracle? Our PROOFGRADER (an ensemble of five O3 runs with REF+MS) closely tracks the human-oracle curve, consistently selecting higher-quality proofs as n increases. At n=16, it achieves an average score of 4.14.

Is the ranking between evaluators from §3 preserved? Comparing the four different fine-grained evaluators from §3: REF+MS ensemble (PROOFGRADER), REF+MS, REF, and NONE, we find the same ranking holds. Notably, NONE performs significantly worse than those using a marking scheme, confirming that the marking scheme is a critical component.

Do fine-grained scores provide a stronger selection signal than binary labels? We further compare with binary evaluators: the O3-based models prompted to classify each proof as "Correct" or "Incorrect" rather than assigning a 0–7 score.

In contrast to PROOFGRADER, which reaches a score of 4.14 at n=16, the binary evaluator performs only slightly better than the average score of all candidates. This limitation comes from collapsing all "correct" (or "incorrect") proofs into a single category, losing the ability to rank solutions. When multiple correct candidates exist, it cannot distinguish an adequate proof (e.g., a 5/7) from an excellent one (7/7). We thus conclude that fine-grained scoring preserves relative ordering, making it essential for effective reward models in mathematical reasoning.

4.2 ROBUSTNESS AGAINST COMPARISON-BASED SELECTION STRATEGIES

Prior work (Liu et al., 2025b) explored relative comparison methods for improving best-of-n (BoN), e.g., iteratively performing pairwise comparison between generations to select the best one. This is significantly more computationally expensive than using PROOFGRADER, which scores each candidate independently without pairwise evaluation. We compare PROOFGRADER against two comparison-based methods: Tournament (all-pairs) and Knockout (single-elimination).

Tournament. For candidates $s_1, \ldots, s_n$, the selector performs pairwise comparisons for every unordered pair (i, j). Let $W_{ij} \in 0, 1$ indicate whether s_i beats s_j. We form a win-rate matrix (W) and compute $w_i = \sum_{j \neq i} W_{ij}$. The selected answer is $\arg\max_i w_i$ (tie-break by head-to-head or margin if applicable). This uses $\binom{n}{2}$ comparisons.

Knockout. Candidates are placed in a bracket; the selector compares pairs, and the winner advances until one remains. This requires $(n - 1)$ comparisons per bracket. (Optionally, we average over multiple random brackets and select the modal winner or the highest aggregate win rate.)

Figure 4b shows that PROOFGRADER consistently dominates all selection variants across n, indicating that a strong, context-rich scoring function matters more than re-querying the same signal with pairwise procedures. Iterative strategies (Tournament, Knockout) still yield clear gains over mean baseline. Tournament may continue improving beyond n=16, but its quadratic cost $\mathcal{O}\left(\binom{n}{2}\right)$ is substantially higher than alternatives. Finally, as Tournament and Knockout can also be applied at test time to boost proof quality, the results suggest that proof generation benefits from increased test-time compute, echoing prior findings (Dekoninck et al., 2025).

5 RELATED WORK

While prior work has established the importance of proof-based evaluation and explored the "LLM-as-a-judge" paradigm, a critical gap remains. No existing work has conducted a systematic, empirical study of the evaluator design space for mathematical proofs, nor provided a comprehensive dataset with the fine-grained, expert-annotated proofs necessary for such a study. Consequently, the key factors that determine evaluator robustness remain largely uninvestigated. Our work addresses these gaps, and we situate our contributions in relation to several primary lines of research below.

Benchmarks for Mathematical Reasoning. Datasets such as GSM8K (Cobbe et al., 2021), which targets grade-school word problems, and MATH (Hendrycks et al., 2021) and AIME, which extend coverage to algebra, calculus, and contest problems, have served as important benchmarks in math reasoning, but primarily focus on short, closed-form answers. More recent competition based benchmarks, including Omni-MATH (Gao et al., 2024), OlympiadBench (He et al., 2024), HARP (Yue et al., 2024), and MathArena (Balunović et al., 2025), are substantially more challenging; however, they still largely emphasize final answer matching, leaving mathematical proof problems underrepresented. MathArena includes proof problems, but only focuses on latest competitions. Meanwhile, proof generation has advanced rapidly—Gemini (Luong & Lockhart, 2025) and unreleased OpenAI systems have reached IMO gold level performance, and model-agnostic verification-and-refinement enables comparable results for base models (Huang & Yang, 2025). However, these studies largely depend on manual grading to assess proof quality (Petrov et al., 2025).

Automated Evaluation for Generative Outputs. LLM-as-a-judge methods score open-ended outputs along axes such as correctness, enabling scalable evaluation at reduced human cost (Zheng et al., 2023; Li et al., 2024). Subsequent work probes judge reliability on harder tasks (Tan et al., 2025; Frick et al., 2024) and improves fidelity with prompt-specific rubrics that encode task-adapted criteria (e.g., relevance, depth) directly in the prompt (Kim et al., 2024; Liu et al., 2023b; Wadhwa et al., 2025). For mathematical proofs, automated evaluation remains limited: prior efforts are either domain-specific (e.g., inequality proofs in IneqMath (Sheng et al., 2025)) or provide only coarse labels (binary correctness in Dekoninck et al. (2025)).

Formal Proof Generation. A complementary line of work leverages interactive theorem provers (ITPs), where LLMs generate proofs in formal languages like Lean or Isabelle. The correctness of these proofs can be automatically verified by the ITP, guaranteeing their logical soundness. Major benchmarks in this area include miniF2F (Zheng et al., 2021), FIMO (Liu et al., 2023a), PutnamBench (Tsoukalas et al., 2024), which formalize competition math problems; and LeanWorkbook (Ying et al., 2024), which provides a broad Lean corpus for training and evaluation. While this approach is promising, our work focuses on informal proofs for two reasons. First, natural language remains the primary medium for mathematical communication in education and research. Second, the task of automatically translating informal proofs into a formal language, i.e., autoformalization, is itself an exceptionally challenging research problem (Gao et al., 2025; Liu et al., 2025a).

Evaluation for Reward Modeling. Evaluating reward models (RMs) is crucial for ensuring their alignment with human preferences in downstream tasks. Early works pioneered the use of best-of-n (BoN) sampling as a lightweight proxy for RM quality, where multiple LM generations are scored and the top-n selected, with performance measured via win rates against gold-standard RMs (Stiennon et al., 2022; Nakano et al., 2022). Subsequent studies explored BoN's vulnerabilities, such as overoptimization and reward hacking, through scaling experiments on preference datasets like HH-RLHF (Gao et al., 2022). Domain-specific evaluations, such as in math reasoning, leverage pairwise RMs for BoN tournaments, boosting accuracies on benchmarks, such as MATH-500 (Liu et al., 2025b). Comprehensive benchmarks like RewardBench 2 further correlate BoN performance with RM scores on diverse tasks (Malik et al., 2025).

6 CONCLUSION

In this work, we addressed the critical challenge of reliably evaluating natural language mathematical proofs. We introduce a systematic methodology for developing and validating fine-grained evaluators and create PROOFBENCH, the first comprehensive annotated dataset of fine-grained proof grading that spans multiple contests and years. Using PROOFBENCH, we systematically explore the evaluator design space and find that quality depends on a strong model backbone, informative context, and problem-specific marking schemes, and that ensembling further improves accuracy and robustness. Our analysis yields PROOFGRADER, the strongest evaluator considering all design choices. Finally, we demonstrate the practical utility of the evaluators in a downstream best-of-n selection task, where it closes 78% of the performance gap to the human oracle. We expect our methodology, benchmark, and findings to provide a strong foundation for future research in hard-to-verify mathematical reasoning tasks.

Limitation and Future Work. While our work advances reliable evaluation of natural-language mathematical proofs, several limitations remain. First, our scope—olympiad-style proofs in natural language—does not yet cover research-grade arguments, specialized domains (e.g., algebraic geometry), or applied settings; extending to these regimes is an important next step. Second, evaluator quality can likely be improved through better prompting and workflow designs, Our dataset and analyses are intended as a foundation for such work, and we encourage future work to continue improve evaluators using PROOFBENCH as a testbed. Our evaluation focuses exclusively on mathematical correctness. Other crucial aspects of natural-language proofs, such as readability, clarity, and elegance, are not currently assessed, and we leave the development of metrics for these qualities as future work. Third, our strongest evaluators currently rely on closed models. Although our design is model-agnostic, further improving fully open-source evaluators to match those based on closed models is an important direction. To support this, we release prompts and evaluation scripts for reproduction with open models. Beyond evaluation, our graders could be used to synthesize supervision for reward-model training. While our best-of-n experiments use the evaluator as a selection signal, a first step toward training-time integration, we have not yet train with it; incorporating it into proof generator training is a promising future direction.

REFERENCES

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, 2025. URL https://arxiv.org/abs/2505.23281.

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, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Jasper Dekoninck, Ivo Petrov, Kristian Minchev, Mislav Balunovic, Martin Vechev, Miroslav Marinov, Maria Drencheva, Lyuba Konova, Milen Shumanov, Kaloyan Tsvetkov, et al. The open proof corpus: A large-scale study of llm-generated mathematical proofs. *arXiv preprint arXiv:2506.21621*, 2025.

Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios N. Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. How to evaluate reward models for rlhf, 2024. URL `https://arxiv.org/abs/2410.14872`.

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, 2024. URL `https://arxiv.org/abs/2410.07985`.

Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. Herald: A natural language annotated lean 4 dataset. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=Se6MgCtRhz`.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization, 2022. URL `https://arxiv.org/abs/2210.10760`.

Jiacheng Guo, Yue Wu, Jiahao Qiu, Kaixuan Huang, Xinzhe Juan, Ling Yang, and Mengdi Wang. Temporal consistency for llm reasoning process error identification, 2025. URL `https://arxiv.org/abs/2503.14495`.

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, 2024. URL `https://arxiv.org/abs/2402.14008`.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Yichen Huang and Lin F. Yang. Gemini 2.5 pro capable of winning gold at imo 2025, 2025. URL `https://arxiv.org/abs/2507.15855`.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models, 2024. URL `https://arxiv.org/abs/2310.08491`.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline, 2024. URL `https://arxiv.org/abs/2406.11939`.

Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, et al. Fimo: A challenge formal dataset for automated theorem proving. *arXiv preprint arXiv:2309.04295*, 2023a.

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 *The Thirteenth International Conference on Learning Representations*, 2025a. URL `https://openreview.net/forum?id=hUb2At2DsQ`.

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

Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Calibrating llm-based evaluator, 2023b. URL `https://arxiv.org/abs/2309.13308`.

Thang Luong and Edward Lockhart. Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad, July 2025. URL https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/. Blog post.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation, 2025. URL https://arxiv.org/abs/2506.01937.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022. URL https://arxiv.org/abs/2112.09332.

Ivo Petrov, Jasper Dekoninck, Lyuben Baltadzhiev, Maria Drencheva, Kristian Minchev, Mislav Balunović, Nikola Jovanović, and Martin Vechev. Proof or bluff? evaluating llms on 2025 usa math olympiad. *arXiv preprint arXiv:2503.21934*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/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.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2022. URL https://arxiv.org/abs/2009.01325.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges, 2025. URL https://arxiv.org/abs/2410.12784.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

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. *Advances in Neural Information Processing Systems*, 37: 11545–11569, 2024.

Manya Wadhwa, Jifan Chen, Junyi Jessy Li, and Greg Durrett. Using natural language explanations to rescale human judgments, 2025. URL https://arxiv.org/abs/2305.14770.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example, 2025. URL https://arxiv.org/abs/2504.20571.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. Lean workbook: A large-scale lean problem set formalized from natural language math problems. *Advances in Neural Information Processing Systems*, 37:105848–105863, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL `https://arxiv.org/abs/2503.14476`.

Albert S. Yue, Lovish Madaan, Ted Moskovitz, DJ Strouse, and Aaditya K. Singh. Harp: A challenging human-annotated math reasoning benchmark, 2024. URL `https://arxiv.org/abs/2412.08819`.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL `https://openreview.net/forum?id=uccHPGDlao`.

A Appendix

A.1 Data Information

Table 7 provides short descriptions of the contests which we sourced problems from. The data information for the collected problems are available in Table 8. Problem 4 from EGMO 2023 was intentionally removed because it contains figure in problem text.

A.2 Annotation Pipeline Details

Stage 1: Marking Scheme Finalization. *Pilot.* We compare marking scheme generators from two model families, each *with* and *without* in-context examples, using an initial prompt distilled from authoritative grading materials.

Quality rating. For each problem–solution pair, annotators independently rate the generated marking scheme on a 0–3 scale (0 = invalid; 3 = high-fidelity), then discuss to consensus.

Selection and refinement. We select the best configuration (gemini-2.5-pro, no in-context example), iteratively refine the prompt based on annotator feedback, and re-evaluate on additional problems until agreement stabilizes. We then **freeze the marking scheme generator for subsequent use.**

Stage 2: Solution annotation. *Pilot calibration.* Using the frozen rubric generator, we produce per-problem marking schemes and calibrate the scoring protocol. Two experts will both annotate 36 problems, 3 responses each. They will discuss disagreement to reach consensus and adjust their grading protocol.

Contest	Description
APMO	The Asia Pacific Mathematical Olympiad is a regional proof-based contest for high school students across Asia–Pacific. It features five challenging problems solved over a fixed time window, proctored locally.
EGMO	The European Girls' Mathematical Olympiad is an international two-day proof contest for female students, modeled after the IMO. It promotes participation and excellence among young women in mathematics.
IMO	The International Mathematical Olympiad is the premier global high school math competition. Over two days, students tackle six proof problems that test creativity, rigor, and depth.
PUTNAM	The William Lowell Putnam Mathematical Competition is a highly challenging proof exam for undergraduates in the U.S. and Canada. It consists of 12 problems emphasizing ingenuity and precise argumentation.
TST	Team Selection Tests are national-level olympiad exams used by USA to select their IMO teams. Problems mirror international difficulty and assess readiness for global competition.
USAMO	The United States of America Mathematical Olympiad is an invitational proof contest following AMC/AIME qualification. It identifies top U.S. high school problem solvers and feeds into further training and team selection.

Table 7: Brief descriptions of core contests.

Contest	2022	2023	2024	2025	Total
APMO	5	5	5	5	20
EGMO	6	5	6	6	23
IMO	6	6	6	6	24
PUTNAM	12	12	12	–	36
TST	–	6	6	6	18
USAMO	6	6	6	6	24
Total per year	**35**	**40**	**41**	**29**	**145**

Table 8: Core problem counts by contest and year.

Scale and quality control. We apply the calibrated protocol to 145 problems with 3 solutions each, yielding 435 rubric-guided annotations. We double-score 40% of items, run periodic drift checks, and adjudicate all flagged disagreements.

Screenshots of the annotation interface are shown in Figure 5 and Figure 6

A.3 Additional Details for Evaluation Metrics

Within-problem ranking agreement. For problem p, consider all pairs (i, j) with $i < j$. Define $\Delta_{ij}^{\text{exp}} = y_{pi} - y_{pj}$ and $\Delta_{ij}^{\text{eval}} = \hat{y}_{pi} - \hat{y}_{pj}$. Let C and D be the numbers of concordant and discordant pairs, respectively, and let

$$T_{\text{exp}} = \#\{(i,j) : \Delta_{ij}^{\text{exp}} = 0\}, \qquad T_{\text{eval}} = \#\{(i,j) : \Delta_{ij}^{\text{eval}} = 0\}.$$

Kendall's τ_b for problem p is

$$\tau_b(p) = \frac{C - D}{\sqrt{(C + D + T_{\text{exp}})\,(C + D + T_{\text{eval}})}},$$

and is undefined if the denominator is zero (we omit such p from aggregation).

Macro averaging. We macro-average per-problem metrics across problems. Writing $\mathcal{P}$ for the set of problems with defined τ_b and $P' = |\mathcal{P}|$:

$$\overline{\text{MAE}} = \frac{1}{P}\sum_{p=1}^{P} \text{MAE}_p, \quad \overline{\text{RMSE}} = \frac{1}{P}\sum_{p=1}^{P} \text{RMSE}_p, \quad \overline{\text{Bias}} = \frac{1}{P}\sum_{p=1}^{P} \text{Bias}_p,$$

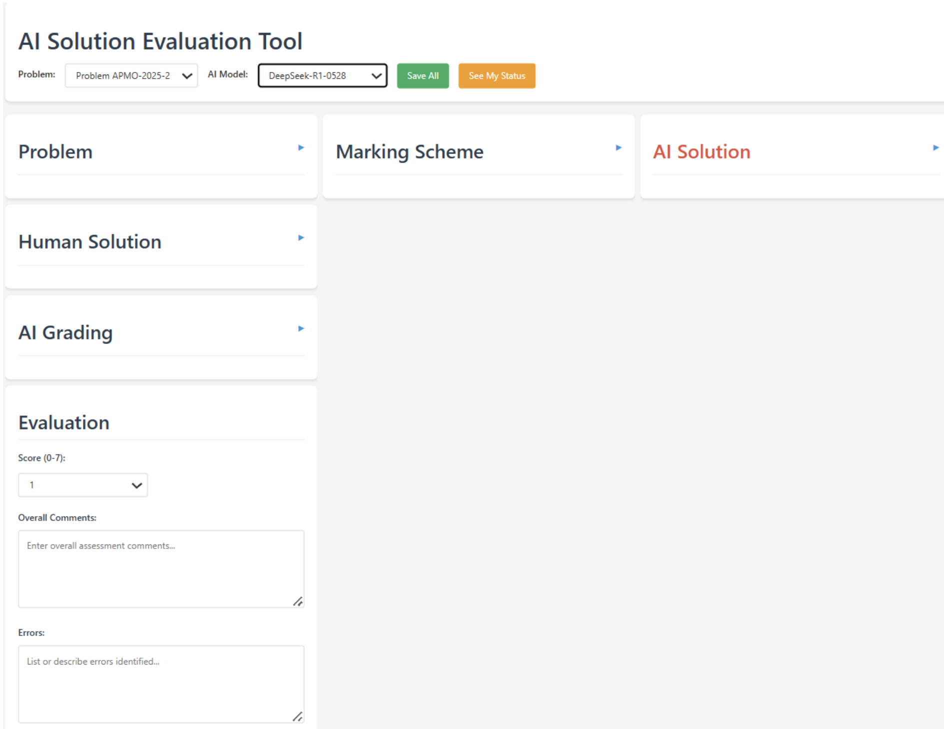


Figure 5: **View of Evaluation Platform Setup**

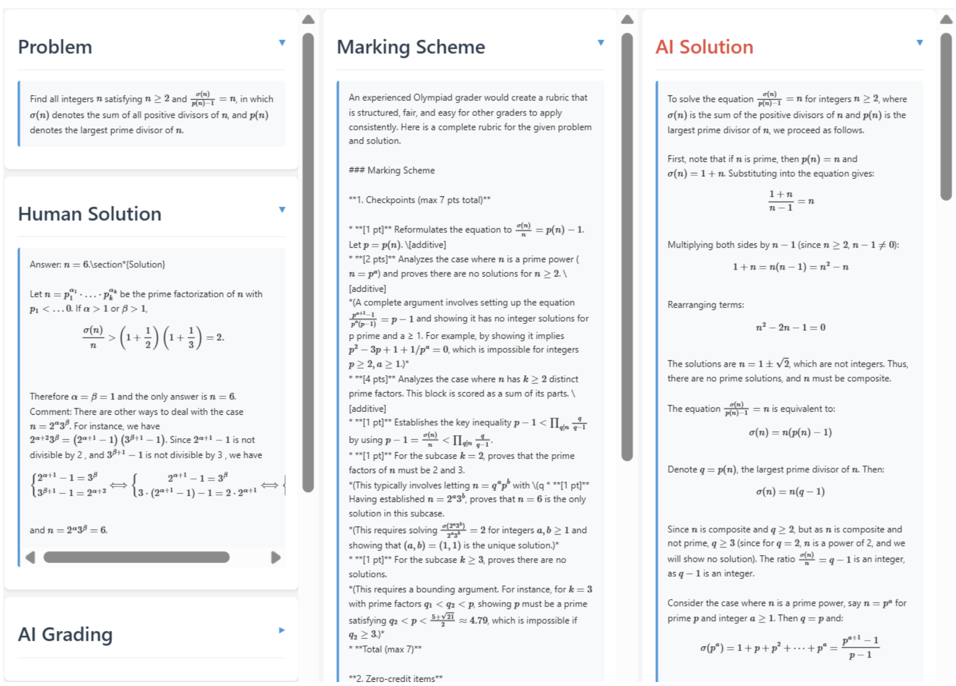


Figure 6: **View of Evaluation Platform Setup**

$$\overline{\mathrm{WTA}}(\leq 1) = \frac{1}{P}\sum_{p=1}^{P}\mathrm{WTA}_p(\leq 1), \overline{\tau_b} = \frac{1}{P'}\sum_{p\in\mathcal{P}}\tau_b(p).$$

(As noted in the main text, all problems have the same response count n.)

Evaluator	Comparison	MAE_{home}	MAE_{cross}	Δ MAE	Bootstrap CI (95%)	p-value
O3	O3 vs R1	1.120	0.993	+0.127	[-0.187, +0.440]	0.452
	O3 vs GEMINI	**1.120**	0.778	**+0.342**	[+0.067, +0.627]	0.017*
GEMINI	GEMINI vs R1	**1.616**	1.183	**+0.433**	[+0.123, +0.743]	0.006**
	GEMINI vs O3	**1.616**	1.225	**+0.391**	[+0.070, +0.711]	0.019*
R1	R1 vs GEMINI	1.508	1.433	+0.075	[-0.337, +0.492]	0.724
	R1 vs O3	1.539	1.199	+0.340	[-0.051, +0.731]	0.087

Note: Bootstrap confidence intervals (10,000 samples).
Significance: $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$.
Bold: significant home disadvantage (CI excludes zero, home > cross).

Table 9: Statistical tests comparing same-generator vs. cross-generator MAE for each evaluator.

Model	**Marking Scheme**	RMSE↓	MAE↓	$WTA_{\leq 1}$ (%)↑	Kendall-τ ↑	Bias≈
O3	**Original**	**1.273**	**0.964**	**76.5**	0.502	$\mathbf{-0.008}$
	New marking scheme (GEMINI)	1.467	1.156	70.0	**0.515**	0.219
	New marking scheme (O3)	1.525	1.196	70.4	0.460	0.020

Table 10: **Marking Scheme Sensitivity.** We evaluate the single-pass O3 evaluator under REF+MSwith three marking-scheme sets—original (human), regenerated (same model/prompt), and O3-generated (same prompt). The original human schemes perform best.

A.4 Additional Results

We provide bootstrap data for comparing same-generator vs. cross-generator MAE for each evaluator mentioned in subsection 3.5. Table 10 reports results for O3 evaluators under REF+MS, comparing three marking-scheme variants: (i) the original schemes used by human experts, (ii) schemes regenerated by the same model with the same prompt, and (iii) schemes generated by O3 itself with the same prompt. The original human-provided schemes yield the strongest performance.

A.5 Generator Prompt

Default

```
Your task is to write a proof solution to the following
problem. Your proof will be graded by judges for correctness
and completeness. 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.
- 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.

FORMATTING GUIDELINES:
- You should write Markdown with LaTeX math. Do NOT use
code fences (no ```).
- You should use correct LaTeX notation to write equations
and mathematical symbols. You should encompass these
equations in correct delimiters ("\\(" and "\\)" for
inline math, "\\[" and "\\]" for block math) to enhance
the clarity of your proof. **Do not use any unicode
characters.**
- For multi-line derivations, wrap an aligned block
INSIDE display math.
- Do not use other LaTeX environments or packages.

PROBLEM: {problem}
````

A.6 MARKING SCHEME GENERATION PROMPT

Marking Scheme

````
You are an experienced Olympiad grader.
*Treat the official solution as fully correct and
authoritative; do not claim it contains errors or gaps.*
Your task is to write a complete, grader-friendly rubric for
the problem and official solution below. The rubric must map
a student's proof to an integer score from **0 to 7**.

--------------------------------------------------------------

### INSTRUCTIONS

Produce the marking scheme in **exactly** the following three
sections.
1. **Checkpoints (max 7 pts total)**

  * Break the solution into logically independent checkpoints
  with **integer** point values.

    * Allocate **>= 4 pts** to the main idea/critical steps;
    **<= 3 pts** to routine work.
  * If two items are mutually exclusive (solve the *same*
  logical gap), **nest** them and write **\award the larger
  only"**.
  * **Parallel solution paths (non-additive):**
    * If the official solution (or a student submission)
    admits more than one legitimate approach, write
    **parallel checkpoint chains** labeled \Chain A
````

```
    / Chain B / ... (idea: ...)".
    * **Start this section with a bold rule:** **Score
    exactly one chain | take the **maximum** subtotal
    among chains; do **not** add points across chains.**
    * Within a chain, checkpoints may be \[additive]. For
    steps **shared across chains** (e.g., a lemma usable by
    multiple approaches), place them under a \Shared
    prerequisites" group with **\[max k]**, and state
    **\count at most once regardless of chain."**
  * For every bullet (or group), append either
  **\[additive]** or **\[max k]** to make the scoring
  rule unambiguous.
  * Finish with a one-line **\Total (max 7)"** check that is
  consistent with the non-additivity rule.
  * Never demand cosmetic labels or specific variable names
  unless essential to the logic.

2. **Zero-credit items**

  * List common arguments or observations that **earn 0
  points** (e.g., conjectures without proof|especially in
  geometry, routine restatements of the problem, or
  dead-ends).

3. **Deductions**

  * Bullet each typical mistake with a **flat** penalty
  (**{1**, **{2**, or **\cap at x/7"**).
  * Apply **at most the single largest** deduction; never
  reduce a score below 0.
  * Target logic gaps, invalid claims, circular reasoning,
  or contradictions. Cosmetic slips (notation, arithmetic,
  wording) do **not** trigger deductions unless they break
  validity.
  * If a student gives **multiple distinct proofs**, **grade
  the best one only** (no stacking). If proofs contain
  **contradictory claims**, apply an appropriate deduction
  (e.g., **cap at 5/7**).

### IMPORTANT REQUIREMENTS

* Use **concise bullets** | no prose exposition of the
official solution.
* **Arithmetic sanity checks:** per-chain checkpoint sums
and \[max k]
caps must make it impossible to exceed **7** overall; at
least one chain must allow a perfect **7/7**.
* Do **not** introduce, \fix," or critique the official
solution.
* Avoid over-fragmenting: do not split routine algebra
into 3+ separate 1-pt bullets.
* Keep notation consistent with the official solution;
define any new symbols you introduce.

---------------------------------------------------------

### PROBLEM
```

```
{problem}

### OFFICIAL SOLUTION

{solution}
```

A.7 Evaluator Prompts

Below we list all prompts used in our evaluation. Each prompt is shown verbatim.

With Reference Solution and Marking Scheme

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

### Input

Your input will consist of:

* **Problem Statement**: A mathematical problem that the proof
is attempting to solve.
* **Reference Solution**: A correct solution or proof provided
for reference. This is **not necessarily the only valid solution**.
If the problem requires a final numeric or algebraic answer, this
section contains the correct answer, which should be the only
accepted final answer (though alternative reasoning paths are valid).
* **Marking Scheme**: A problem-specific grading rubric (0{7 scale)
with checkpoints, zero-credit items, and deductions. **Treat this
scheme as advisory guidance, not a script.** Use it to anchor
scoring, but **do not require** the proof to follow the same
order, lemmas, or technique if its reasoning is mathematically
sound.
* **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.

### Task

Analyze the proof carefully.

**Core principles (in order of precedence):**
1) **Mathematical validity** of the proof's reasoning and conclusion.
2) **Problem constraints** (e.g., unique required final value;
forbidden tools if stated).
3) **Advisory mapping to the marking scheme** (checkpoints/
deductions), allowing different orders and techniques.
4) **Reference solution** as an anchor for sufficiency, not
exclusivity.

**Alternative-approach policy:**
- If the proof uses a different but valid method, **map its
steps to equivalent rubric checkpoints** (same logical role)
and award points accordingly.
- **Do not penalize** solely for re-ordering steps, using
different lemmas, or giving a correct shortcut, **unless**
the problem forbids it.
- Apply zero-credit items/deductions **only when the underlying
issue actually occurs** in the given proof's approach; **do not
auto-penalize** for omitting a rubric step that is unnecessary
under the alternative method.
- Avoid double-counting mutually exclusive items; if two items
solve the same logical gap, **award the larger only**.
```

```
- If the final numeric/algebraic answer is wrong where uniqueness
is required, award only partial credit justified by correct
intermediate reasoning.

**Rigor and evidence:**
- Award credit for intermediate claims **only if adequately
justified** within the proof (not merely asserted).
- If a step is plausible but under-justified, award **conservative
partial credit** and note what is missing.

**What to produce:**
- Identify logical errors, incorrect steps, or unclear reasoning.
- Give a **score between 0 and 7** with a **detailed assessment**.
- **Within the assessment text**, show clearly how the score was
derived:
  - Which rubric checkpoints (or their **mapped equivalents**)
  were earned and the points you awarded.
  - Any zero-credit items or deductions you applied (and why).
  - How these add up to the final integer score in [0-7].

### Output Format

Respond with **only** well-formed XML using the structure below.
Do not include any extra text or Markdown.

**Requirements:**
- `<score>` must be an integer in [0, 7].
- `<assessment>` must be a **detailed analysis** that explains
your reasoning step-by-step and provides a clear **rationale for
the score**. Reference specific claims/lines if present. Include
the scoring breakdown **in prose** here (earned checkpoints or
mapped equivalents, deductions, and subtotal → final score).
- `<errors>` must be a list of specific issues (empty if score = 7).

Example output:

<score>0</score>
<assessment>The proof shows a good understanding of the main idea,
but has some unclear reasoning and minor mistakes...</assessment>
<errors>
  1. specific error 1,
  2. specific error 2,
  ...
</errors>

-----------------------------------------------------------
**Problem Statement**
{problem}

**Reference Solution**
{human_solution}

**Marking Scheme**
{marking_scheme}

**Proof Solution**
{solution}
```

Basic Evaluation Template

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

```
### Input

Your input will consist of:

* **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.

### Task

Analyze the proof carefully.

* Identify logical errors, incorrect steps, or unclear reasoning.
* Give an **integer** score between 0 and 7 with a brief overall
assessment.

### Output Format

Respond with **only** well-formed XML using the structure below.
Do not include any extra text or Markdown.

**Requirements:**
- `<score>` must be an integer in [0, 7].
- `<assessment>` must be a **detailed analysis** that explains your
reasoning step-by-step and provides a clear **rationale for the
score**. Reference specific claims/lines if present.
- `<errors>` must be a list of specific issues (empty if score = 7).

Example output:

<score>0</score>
<assessment>The proof shows a good understanding of the main idea,
but has some unclear reasoning and minor mistakes...</assessment>
<errors>
  1. specific error 1,
  2. specific error 2,
  ...
</errors>

### Scoring Guidelines (0-7 scale)

* **0**: Completely incorrect; proof is irrelevant, nonsensical, or
shows no understanding.
* **1-2**: Very poor; major logical flaws, does not solve the problem,
but may contain fragments of relevant reasoning.
* **3-4**: Partial progress; captures some correct reasoning or key
ideas, but has significant logical errors, missing steps, or
incomplete arguments that make the proof invalid overall.
* **5-6**: Largely correct; the proof is overall valid and reaches
the correct conclusion. Contains only **minor issues** (e.g., small
calculation mistakes, notation slips, or slightly unclear wording)
that do not undermine correctness.
* **7**: Fully correct; the proof is complete, logically sound, and
clearly presented with no substantive errors.

----------------------------------------------------------
**Problem Statement**
{problem}

**Proof Solution**
{solution}
```

With Reference Solution

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

### Input

Your input will consist of:

* **Problem Statement**: A mathematical problem that the proof is
attempting to solve.
* **Reference Solution**: A correct solution or proof provided for
reference. This is **not necessarily the only valid solution**. If
the problem requires a final numeric or algebraic answer, this
section contains the correct answer, which should be the only
accepted final answer (though alternative reasoning paths are valid).
* **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.

### Task

Analyze the proof carefully.

* Compare the proof against the reference solution where relevant.
* Identify logical errors, incorrect steps, or unclear reasoning.
* Give a score between 0 and 7 with a brief overall assessment.

### Output Format

Respond with **only** well-formed XML using the structure below.
Do not include any extra text or Markdown.

**Requirements:**
- `<score>` must be an integer in [0, 7].
- `<assessment>` must be a **detailed analysis** that explains your
reasoning step-by-step and provides a clear **rationale for the
score**. Reference specific claims/lines if present.
- `<errors>` must be a list of specific issues (empty if score = 7).

Example output:

<score>0</score>
<assessment>The proof shows a good understanding of the main idea
but has some unclear reasoning and minor mistakes...</assessment>
<errors>
  1. specific error 1,
  2. specific error 2,
  ...
</errors>

### Scoring Guidelines (0-7 scale)

* **0**: Completely incorrect; proof is irrelevant, nonsensical, or
shows no understanding.
* **1-2**: Very poor; major logical flaws, does not solve the problem,
but may contain fragments of relevant reasoning.
* **3-4**: Partial progress; captures some correct reasoning or key
ideas, but has logical errors, missing steps, or incomplete arguments
that make the proof invalid overall.
* **5-6**: Largely correct; the proof is overall valid and reaches
the correct conclusion. Contains only **minor issues** (e.g., small
calculation mistakes, notation slips, or slightly unclear wording)
that do not undermine correctness.
```

```
* **7**: Fully correct; the proof is complete, logically sound, and
clearly presented with no substantive errors.

----------------------------------------------------------
**Problem Statement**
{problem}

**Reference Solution**
{human_solution}

**Proof Solution**
{solution}
```

With Reference Solution and Marking Scheme (Strict)

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

### Input

Your input will consist of:

* **Problem Statement**: A mathematical problem that the proof is
attempting to solve.
* **Reference Solution**: A correct solution or proof provided for
reference. This is **not necessarily the only valid solution**. If
the problem requires a final numeric or algebraic answer, this
section contains the correct answer, which should be the only
accepted final answer (though alternative reasoning paths are valid).
 * **Marking Scheme**: A problem-specific grading rubric (0-7 scale)
 with checkpoints, zero-credit items, and deductions. You must
 follow this scheme when assigning points.
* **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.

### Task

Analyze the proof carefully.

* Compare the proof against the reference solution and the marking
scheme.
* Award points according to the marking scheme's checkpoints,
zero-credit items, and deductions.
* Identify logical errors, incorrect steps, or unclear reasoning.
* Give a score between 0 and 7 with a brief overall assessment.
* Show clearly how the score was derived:
  * Which checkpoints were earned (with awarded points).
  * Any zero-credit items or deductions applied.
  * How the subtotal leads to the final score (0-7).

### Output Format

Respond with **only** well-formed XML using the structure below.
Do not include any extra text or Markdown.

**Requirements:**
- `<score>` must be an integer in [0, 7].
- `<assessment>` must be a **detailed analysis** that explains
your reasoning step-by-step and provides a clear **rationale for
the score**. Reference specific claims/lines if present.
- `<errors>` must be a list of specific issues (empty if score = 7).
```

```
Example output:

<score>0</score>
<assessment>The proof shows a good understanding of the main idea
but has some unclear reasoning and minor mistakes...</assessment>
<errors>
  1. specific error 1,
  2. specific error 2,
  ...
</errors>

----------------------------------------------------------
**Problem Statement**
{problem}

**Reference Solution**
{human_solution}

**Marking Scheme**
{marking_scheme}

**Proof Solution**
{solution}
```

With Reference Solution and Marking Scheme (most basic)

```
You are an expert grader for math proofs. Judge the proof's
mathematical correctness based on the reference solution and
the marking scheme, return an integer score between 0 and 7.

INPUTS:
- Problem Statement
- Reference Solution (correct but not exclusive)
- Marking Scheme (0-7) with checkpoints and deductions | use
as guidance, not a script
- Proof Solution (from an LLM)

OUTPUT (XML only; no extra text):
<score>[integer 0-7]</score>
<assessment>[step-by-step rationale with scoring breakdown in prose]
</assessment>
<errors>[numbered list of specific issues; empty if none]</errors>

----------------------------------------------------------

**Problem Statement**
{problem}

**Reference Solution**
{human_solution}

**Marking Scheme**
{marking_scheme}

**Proof Solution**
{solution}
```

With Reference Solution and Marking Scheme (basic)

```
You are an expert grader for math proofs.

INPUTS:
```

```
- Problem Statement
- Reference Solution (correct but not exclusive)
- Marking Scheme (0-7) with checkpoints and deductions  use as
guidance, not a script
- Proof Solution (from an LLM)

TASK:
Judge the proof's mathematical correctness. Prefer validity > problem
constraints > marking scheme alignment > reference solution. If the
proof uses a different valid method, map its steps to equivalent
marking scheme checkpoints and award points. If a unique final answer
is wrong, give partial credit only for justified intermediate
reasoning.

OUTPUT (XML only; no extra text):
<score>[integer 0-7]</score>
<assessment>[step-by-step rationale with scoring breakdown in prose]
</assessment>
<errors>[numbered list of specific issues; empty if none]</errors>

----------------------------------------------------------

**Problem Statement**
{problem}

**Reference Solution**
{human_solution}

**Marking Scheme**
{marking_scheme}

**Proof Solution**
{solution}
```

With Marking Scheme (no reference solution)

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

### Input

Your input will consist of:

* **Problem Statement**: A mathematical problem that the proof is
attempting to solve.
* **Marking Scheme**: A problem-specific grading rubric (0-7 scale)
with checkpoints, zero-credit items, and deductions. You must follow
this scheme when assigning points.
* **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.

### Task

Analyze the proof carefully.

* Follow the marking scheme exactly: award checkpoints, apply
zero-credit items, and apply any deductions/caps as specified.
* Identify logical errors, incorrect steps, or unclear reasoning.
* Give a score between 0 and 7 with a brief overall assessment.
* Show clearly how the score was derived:
  * Which checkpoints were earned (with awarded points).
  * Any zero-credit items or deductions applied.
```

```
  * How the subtotal leads to the final score (0-7).

### Output Format

Respond with **only** well-formed XML using the structure below.
Do not include any extra text or Markdown.

**Requirements:**
- `<score>` must be an integer in [0, 7].
- `<assessment>` must be a **detailed analysis** that explains your
reasoning step-by-step and provides a clear **rationale for the
score**. Reference specific claims/lines if present.
- `<errors>` must be a list of specific issues (empty if score = 7).

Example output:

<score>0</score>
<assessment>The proof shows a good understanding of the main idea,
but has some unclear reasoning and minor mistakes...</assessment>
<errors>
  1. specific error 1,
  2. specific error 2,
  ...
</errors>

-----------------------------------------------------------
**Problem Statement**
{problem}

**Marking Scheme**
{marking_scheme}

**Proof Solution**
{solution}
```

With Reference Solution and Marking Scheme (more detailed)

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

### Input

Your input will consist of:

* **Problem Statement**: A mathematical problem that the proof is
attempting to solve.
* **Reference Solution**: A correct solution or proof provided for
reference. This is **not necessarily the only valid solution**. If
the problem requires a final numeric or algebraic answer, this section
contains the correct answer, which should be the only accepted final
answer (though alternative reasoning paths are valid).
* **Marking Scheme**: A problem-specific grading rubric (0-7 scale)
with checkpoints, zero-credit items, and deductions. You must follow
this scheme when assigning points.
* **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.

### How to Use the Marking Scheme (mandatory)

1. Checkpoints parsing & awarding
- Treat each checkpoint exactly as written. Respect its tag:
  - [additive]: award all applicable items in that bullet/group.
```

```
  - [max k]: award up to k points from the items in that
  bullet/group (choose the best-matching ones; do not exceed k).
- If items are nested with \award the larger only", and more than
one applies, award only the larger point value.
 - If the scheme presents parallel checkpoint chains (alternative
 legitimate paths), score the single chain or combination that yields
 the highest valid total without violating exclusivity or [max k]
 caps. Do not double-count equivalent steps across mutually
 exclusive paths.
- If a catch-all checkpoint is provided for a fully correct
alternative proof using the same underlying idea, you may award up
to its stated maximum only when the student's argument is complete
and logically valid for that idea.

2. Zero-credit items

If the proof relies on any listed zero-credit arguments, award 0
for those parts. Do not add points for restatements, conjectures
without proof (especially in geometry), or dead-ends.

3. Deductions (apply at most one)

- Identify applicable deductions and apply only the single largest
(e.g., -1, -2, or cap at x/7).
- Apply a cap by truncating the post-checkpoint subtotal to x before
finalizing the score.
- Never reduce the score below 0. Cosmetic slips (notation, arithmetic,
wording) do not trigger deductions unless they break validity.

4. Final answer consistency (when applicable)

If the reference solution gives a definitive final answer, the
candidate solution's final answer must be **correct/equivalent**.
If not, follow the marking scheme's checkpoints/deductions; typically,
a wrong final answer prevents awarding the \conclusion" checkpoint.

5. Arithmetic & bounds
- Checkpoint awards are integers. Subtotal <= 7 by construction.
- After applying the single largest deduction/cap, the final score
is an integer in [0, 7].


### Task

Analyze the proof carefully.

* Compare the proof against the reference solution and the marking
scheme.
* Award points according to the marking scheme's checkpoints,
zero-credit items, and deductions.
* Identify logical errors, incorrect steps, or unclear reasoning.
* Give a score between 0 and 7 with a brief overall assessment.
* Show clearly how the score was derived:
* Which checkpoints were earned (with awarded points).
* Any zero-credit items or deductions applied.
* How the subtotal leads to the final score (0-7).

### Output Format

Respond with **only** well-formed XML using the structure below.
Do not include any extra text or Markdown.

**Requirements:**
```

```
- `<score>` must be an integer in [0, 7].
- `<assessment>` must be a **detailed analysis** that explains your
reasoning step-by-step and provides a clear **rationale for the
score**. Reference specific claims/lines if present.
- `<errors>` must be a list of specific issues (empty if score = 7).

Example output:

<score>0</score>
<assessment>The proof shows a good understanding of the main idea,
but has some unclear reasoning and minor mistakes...</assessment>
<errors>
  1. specific error 1,
  2. specific error 2,
  ...
</errors>

----------------------------------------------------------
**Problem Statement**
{problem}

**Reference Solution**
{human_solution}

**Marking Scheme**
{marking_scheme}

**Proof Solution**
{solution}
```