MathNLP 2025

MathNLP 2025: The 3rd Workshop on Mathematical Natural Language Processing

Proceedings of the Workshop

November 9, 2025

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
317 Sidney Baker St. S
Suite 400 - 134
Kerrville, TX 78028
USA
Tel: +1-855-225-1962
acl@aclweb.org

ISBN 979-8-89176-348-7

MathNLP 2025

The articulation of mathematical arguments is a fundamental part of scientific reasoning and communication. Across many disciplines, expressing relations and interdependencies between quantities is at the centre of scientific argumentation. Nevertheless, despite its importance, the application of contemporary NLP models for inference over mathematical text remains under-explored or subject to important limitations. MathNLP represents a forum for discussing new ideas to advance research on Mathematical Natural Language Processing, welcoming novel contributions on model architectures, evaluation methods and downstream applications.
Recent advances in Natural Language Processing (NLP) enabled by Deep Learning-based architectures bring the opportunity to support the interpretation of textual content at scale. The application of these methods can facilitate scientific discovery, reducing the gap between current research and the available large-scale scientific knowledge. Previous work has shown the potential of designing neural architectures for different mathematical natural language inference tasks, such as premise selection in natural language, expression derivation, and mathematical information retrieval.
However, there are still technical gaps that need to be addressed such as the availability of datasets and evaluation tasks, techniques for the joint interpretation of different modalities present in mathematical text (equational and natural language), the understanding of unique aspects of mathematical discourse and multi-hop models for mathematical inference.
We proposed this workshop as a continuation of our previous editions, with a new emphasis on the integration of Large Language Models (LLMs) and symbolic approaches with the goal of addressing these challenges and connect different experts in this field.
In this edition, MathNLP accepted a total of 40 papers, of which 16 are included in the proceedings. For additional details about MathNLP 2025, please visit the website: https://sites.google.com/view/mathnlp2025.

Organizing Committee

General Chair

Marco Valentino, University of Sheffield

Program Chairs

Deborah Ferreira, The MathWorks
Mokanarangan Thayaparan, The MathWorks
Leonardo Ranaldi, University of Edinburgh
Andre Freitas, Idiap Research Institute and University of Manchester

Program Committee

Program Committee

Kuan Lu
Lei Xu
Surangika Ranathunga
Pierre Beckmann
Xin Quan
Nisansa de Silva
Federico Ranaldi
Aishwarya Jadhav
Pinzhen Chen
Mardhiyah Sanni
Yingji Zhang
Giulia Pucci
Lan Zhang
Joshua Ong Jun Leang
Xingwei Tan
Matt Freestone
David L. J. Ho
Wenda Li
Gautam Pendse

Table of Contents

Syntactic Blind Spots: How Misalignment Leads to LLMs' Mathematical Errors

Dane Williamson, Yangfeng Ji, Matthew Dwyer
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
{dw3zn, yj3fs, md3cn}@virginia.edu

Abstract

Large Language Models (LLMs) demonstrate strong mathematical problem-solving abilities but frequently fail on problems that deviate syntactically from their training distribution. We identify a systematic failure mode, `syntactic blind spots`, in which models misapply familiar reasoning strategies to problems that are semantically straightforward but phrased in unfamiliar ways. These errors are not due to gaps in mathematical competence, but rather reflect a brittle coupling between surface form and internal representation. To test this, we rephrase incorrectly answered questions using syntactic templates drawn from correct examples. These rephrasings, which preserve semantics while reducing structural complexity, often lead to correct answers. We quantify syntactic complexity using a metric based on Dependency Locality Theory (DLT), and show that higher DLT scores are associated with increased failure rates across multiple datasets. Our findings suggest that many reasoning errors stem from structural misalignment rather than conceptual difficulty, and that syntax-aware interventions can reveal and mitigate these inductive failures.

1 Introduction

Large Language Models (LLMs) show strong performance on mathematical benchmarks like GSM8K, SVAMP, MultiArith, and ASDiv (Cobbe et al., 2021; Patel et al., 2021; Roy and Roth, 2015; Miao et al., 2020), yet they frequently make systematic errors, often reapplying familiar solution strategies even when the problem structure changes (Zheng et al., 2024; Bao et al., 2025; Huang et al., 2025).

These errors reflect an overreliance on surface-level pattern matching rather than adaptive reasoning. We focus on a specific class of such failures, which we term *syntactic misalignment*, cases where LLMs fail because a problem's phrasing deviates structurally from patterns they have learned to solve, even though the underlying logic remains unchanged. As shown in Figure 1, rephrasing a problem to reduce syntactic complexity can often reverse these failures.

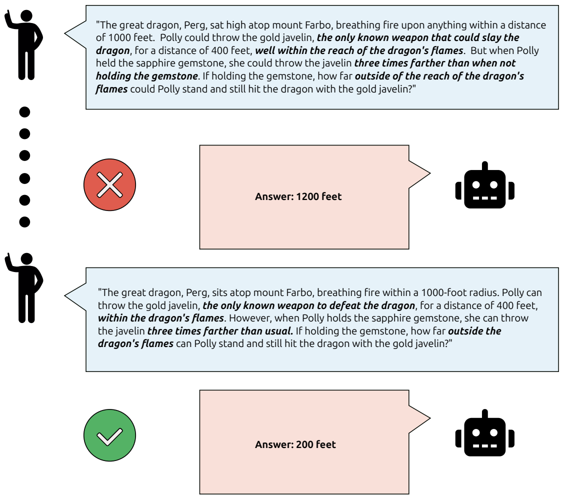


Figure 1: Structural rephrasing improves model accuracy by reducing syntactic complexity and dependency length.

Prior work has documented LLM sensitivity to superficial variation: word order changes, paraphrasing, or structural perturbations can all induce performance drops (Zhang et al., 2024; Srivastava et al., 2024; Huang et al., 2025). Models sometimes ignore altered constraints and produce answers consistent with the original phrasing.

We argue that this brittleness arises from a structural failure mode we call *syntactic induction*, the tendency to treat syntactic similarity as a proxy for problem similarity. This leads models to overapply familiar solution templates, even when the problem logic has changed. Inspired by cognitive science, we draw an analogy to *rule-based overgeneralizations* in human learners (Ben-Zeev, 1998; Karmiloff-Smith, 1986), where errors arise not from lack of competence but from misapplied procedural regularities.

To study this phenomenon, we develop a dependency-guided framework for identifying and

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 1–14

mitigating syntactic induction failures. We use Dependency Locality Theory (DLT) to quantify syntactic complexity and rephrase high-complexity questions using syntactic templates drawn from successful examples. This reveals that many reasoning errors stem not from mathematical difficulty, but from a mismatch between surface form and learned problem schemas.

1.1 Contributions

This work makes four contributions:

- Introduces *syntactic induction failures* as a structured, recurring error mode in LLM mathematical reasoning.
- Bridges LLM error behavior with cognitive science, highlighting parallels in schema-driven failure.
- Proposes a dependency-guided framework for detecting and rephrasing structurally misaligned problems.
- Demonstrates that rephrasing structurally complex math questions significantly improves model accuracy across datasets and models.

2 Related Work

LLM Sensitivity to Structural Variation. While techniques like in-context learning and chain-of-thought prompting have improved LLM math performance (Brown and Mann, 2020; Wei et al., 2022), models remain brittle under surface-level perturbations. Studies have shown that modifying word order, phrasing, or structure leads to significant performance drops (Huang et al., 2025; He et al., 2024; Kang et al., 2024; Zheng et al., 2024; Srivastava et al., 2024). Even when semantics are preserved, models often revert to memorized patterns (Zhang et al., 2024), suggesting an overreliance on surface form as a proxy for problem identity.

Beyond formatting, deeper failure modes have been linked to data contamination (Magar and Schwartz, 2022; Sainz et al., 2023), deductive errors (Ling et al., 2023), and spurious correlations (Zhou et al., 2024; Bao et al., 2025). Most relevantly, Stechly et al. (2025) show that models struggle when questions are phrased in unfamiliar syntactic forms, motivating our focus on syntactic misalignment.

Cognitive Accounts of Structural Sensitivity. These observations parallel well-known findings in cognitive psychology. Chi and et al (1981) distinguish between a problem's surface structure (e.g., phrasing, grammar) and its deep structure (underlying logic). Even experienced solvers often rely on surface cues to access problem schemas (Novick, 1988; Hinsley et al., 1977), which guide solution strategies. Analogical reasoning studies further show that surface similarity influences both novice and expert behavior (Holyoak and Koh, 1987; Ross, 1984).

Our work draws on this perspective, proposing that LLMs exhibit a similar schema-triggered behavior. We formalize this through *syntactic induction*: the tendency to conflate surface-form similarity with structural equivalence. To quantify this effect, we adopt Dependency Locality Theory (DLT) (Gibson, 1998, 2000) and show that higher DLT scores are associated with LLM failure. Rephrasing high-DLT questions often recovers accuracy, supporting a structural account of reasoning breakdown.

Toward a Taxonomy of Rational Errors. Finally, our perspective aligns with work on human error categorization. Rule-based overgeneralizations (Ben-Zeev, 1998; Ashlock, 2002), where valid strategies are misapplied in the wrong context, mirror LLM errors under syntactic shift. We argue that LLMs, like learners, may benefit from structured taxonomies of failure to guide robustness interventions (see Appendix Figure 9).

3 Method: Rephrasing for Reducing Syntactic Misalignment

LLMs often fail when problems are phrased in structurally unfamiliar ways. To diagnose and mitigate these errors, we quantify syntactic complexity using Dependency Locality Theory (DLT) and rephrase structurally complex questions to align with previously successful examples. This section outlines the framework.

3.1 Quantifying Syntactic Complexity with DLT

We define syntactic complexity using a scoring function based on Dependency Locality Theory (DLT). For a math word problem, $q = (w_1, \ldots, w_n)$, the total DLT score is the sum of three costs over all tokens:

$$\mathrm{DLT}(q) = \sum_{i=1}^{n} \Big(\mathrm{Integration}(w_i) + \mathrm{Storage}(w_i) + \mathrm{Discourse}(w_i)\Big) \tag{1}$$

Given the dependency trees of the sentences in q each component in Equation 1 is defined as follows:

Integration : For a token w_i, let h_i be its head token following the dependency tree. The integration cost is the number of new discourse referents between them: $\mathrm{Integration}(w_i) = \sum_{w_j \in \mathrm{Intervening}(w_i, h_i)} \mathbf{1}_{\mathrm{Referent}}(w_j)$, where $\mathbf{1}_{\mathrm{Referent}}(w_j) = 1$ if w_j has POS tag in {NOUN, PROPN, NUM, VERB}.

Discourse : A token introduces discourse cost if it adds a new referent: $\mathrm{Discourse}(w_i) = \mathbf{1}_{\mathrm{Referent}}(w_i)$.

Storage : This is the number of unresolved syntactic expectations at w_i, denoted $\mathrm{Storage}(w_i) = |\mathcal{P}_i|$, where $\mathcal{P}_i$ is the set of pending predictions.

This formulation enables theoretically-grounded, systematic scoring of syntactic complexity for each question based on its dependency parse.

Example: *"Melissa brushes 12 horses on Monday."*

- **Discourse:** All content words ("Melissa", "brushes", "12", "horses", "Monday") introduce discourse referents (PROPN, VERB, NUM, NOUN), yielding a total cost of 5.
- **Integration:** "Horses" depends on "brushes", with "12" intervening. Since "12" is a referent, it contributes an integration cost of 1.
- **Storage:** "Melissa" introduces an unresolved expectation for a verb, resolved by "brushes". New expectations (e.g., for an object) are tracked until resolved.

3.1.1 Normalization

To ensure fair comparison across questions of varying length and content density, we normalize two components of the DLT score. Integration cost is divided by the number of discourse referents, and peak storage cost is scaled by question length. The resulting normalized DLT score is:

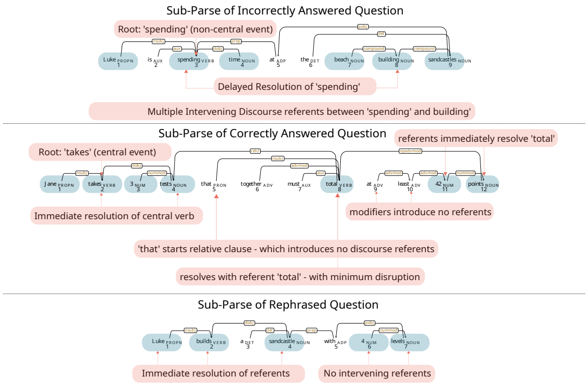


Figure 2: Dependency parses illustrating the rephrasing pipeline. The rephrased version reduces dependency depth and referential interference, lowering DLT-based processing cost.

$$\mathrm{DLT}_{\mathrm{norm}}(q) = \left(\frac{\sum \mathrm{Integration}}{\sum \mathrm{Discourse}}\right) + \left(\frac{\max \mathrm{Storage}}{|q|}\right) + \left(\sum \mathrm{Discourse}\right) \tag{2}$$

This yields a length-independent complexity measure. As shown in section 5, higher normalized-DLT scores correlate with model failure, making this a reliable predictor of syntactic brittleness.

3.2 Dependency-Guided Rephrasing

To correct syntactic misalignment, we rephrase a failed question $q_{\mathrm{incorrect}}$ to resemble a syntactically similar, correctly answered one q_{match}. We identify q_{match} using the Weisfeiler-Lehman Graph Kernel (WLK) (Shervashidze et al., 2011):

$$q_{\mathrm{match}} = \arg\max_{q \in \mathcal{Q}_{\mathrm{correct}}} \mathrm{WLK}(G_{\mathrm{incorrect}}, G_q) \tag{3}$$

We then prompt an LLM to rewrite $q_{\mathrm{incorrect}}$ to match the structure of q_{match}, while preserving semantics:

$$q'_{\mathrm{incorrect}} = \mathcal{M}(q_{\mathrm{incorrect}}, q_{\mathrm{match}}, \mathcal{P}) \tag{4}$$

For example, consider this original question:

> "Luke is spending time at the beach building sandcastles. He eventually notices that. . ."

Its syntactic embedding leads to high DLT cost. The rephrased version:

> "Luke builds a sandcastle with 4 levels, where each level has half the square footage. . ."

flattens dependencies, reduces referential interference, and improves model accuracy.

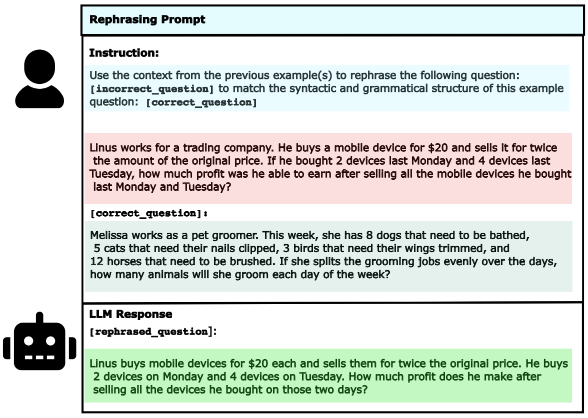


Figure 3: Format of rephrasing prompt. The LLM is prompted to generate a rephrased variant that more closely matches the surface structure of the correctly answered question.

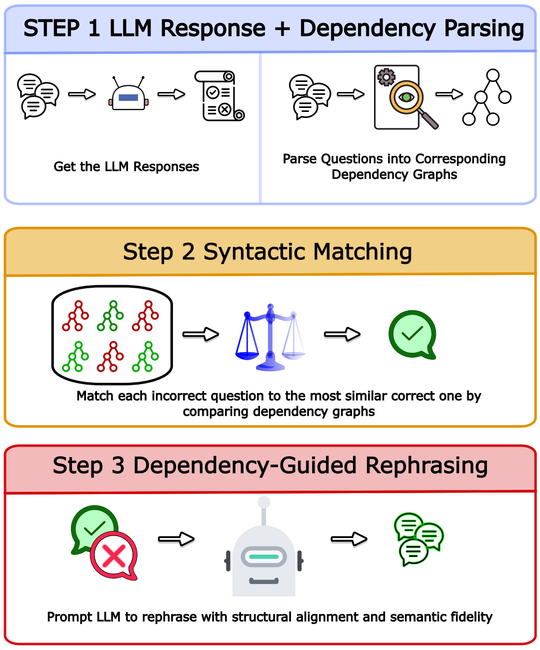


Figure 4: Rephrasing pipeline. An incorrectly answered question is aligned to a syntactically similar, correctly answered one via WL Kernel matching. A k-shot prompt then guides the LLM to generate a syntactically aligned but semantically identical rephrasing.

3.3 Procedure Overview

Our pipeline consists of:

1. Query the model; collect incorrect responses $Q_{\text{incorrect}}$.
2. Parse all questions using spaCy to extract dependency trees.
3. For each $q \in Q_{\text{incorrect}}$, find $q_{\text{match}} \in Q_{\text{correct}}$ via WLK similarity.
4. Prompt the LLM with k-shot examples to rephrase q syntactically in the form of q_{match}.

We then re-query the model on the rephrased versions q' and evaluate whether failures are recovered.

4 Experiment Setup

To assess the impact of syntactic restructuring, we re-evaluate the LLM on the rephrased variants $q'_{\text{incorrect}}$. If accuracy improves significantly, we attribute the original failure to a *syntactic induction failure*: the model's inability to generalize over unfamiliar surface forms despite semantic equivalence.

This evaluation allows us to systematically characterize and quantify a core weakness in LLM reasoning and establish the importance of syntactic alignment for mathematical understanding.

This section outlines our experimental framework for evaluating how syntactic structure influences LLM reasoning performance. We begin by stating our research question, (Section 4.1). We then describe the datasets and preprocessing methods (Section 4.2). Finally, we provide implementation details regarding parsing, tree similarity computation, and model evaluation (Section 4.3).

4.1 Research Questions

This work investigates whether syntactic complexity contributes to reasoning failures in LLMs and whether syntactic restructuring can mitigate those failures. Specifically, we ask:

1. **How effectively does the proposed DLT-based complexity framework differentiate between correctly and incorrectly answered math questions?**

2. **To what extent does syntactic rephrasing, guided by structural similarity to successfully answered questions, improve model accuracy on previously failed examples?**

Before evaluating the effects of syntactic rephrasing, we first investigate whether syntactic complexity alone can be predictive of model failure. For each model, we compute DLT complexity scores for all questions and divide the dataset into two groups: those answered correctly and those answered incorrectly.

To assess whether there is a statistically significant difference in complexity between the two groups, we apply Welch's t-test. This test is appropriate when comparing the means of two samples with potentially unequal variances and sample sizes, conditions that naturally arise given varying model accuracies. The resulting *t-statistic* quantifies the separation between group means relative to their variances, while the corresponding *p-value* indicates whether the observed difference is likely to have occurred by chance.

This analysis allows us to test the hypothesis that higher syntactic complexity, independent of semantic content, is associated with increased model error. A significant result would suggest that DLT complexity serves as a useful predictor of LLM failure, motivating our subsequent rephrasing intervention.

We report the results of this comparison in subsection 5.1 and interpret its implications in subsection 6.1.

4.2 Datasets and Pre-processing

We evaluate five open-source LLMs, **LLaMA**, **Mistral**, **Qwen**, **Gemma**, and **Granite**, in a zero-shot setting on four established math benchmarks: **GSM8K**, **SVAMP**, **MultiArith**, and **ASDiv** (Touvron et al., 2023; Jiang and et al, 2023; Yang et al., 2024; Team et al., 2024; Mishra and et al, 2024). These datasets span diverse reasoning skills and syntactic forms, from simple arithmetic (GSM8K, MultiArith) to structurally perturbed (SVAMP) and linguistically varied (ASDiv) problems.

To analyze syntactic structure, we parse each question using the spaCy NLP toolkit (Honnibal and Montani, 2017), yielding dependency trees that capture syntactic relations. Let $T_{\text{incorrect}}$ denote the tree of an incorrectly answered question, and $\mathcal{T}_{\text{correct}}$ the set of trees from correctly answered ones.

4.3 Implementation Details

Dependency parsing and tree similarity computations are implemented using `spaCy` and `nltk`. We use Hugging Face implementations of all LLMs. [1] All experiments are conducted on NVIDIA RTX 2080 GPUs.

5 Experimental Results

We first test whether surface-level syntactic complexity predicts model failure. We then assess whether syntactic restructuring can recover accuracy on previously incorrect questions.

[1]See Appendix Table 4 for model and hyperparameter details.

Dataset	Model	Correct Mean	Incorrect Mean	Δ DLT
GSM8K	Gemma	22.90	25.71	**+2.81***
	Granite	22.17	25.43	**+3.27***
	LLaMA	23.53	28.10	**+4.57***
	Mistral	22.98	26.21	**+3.23***
	Qwen	23.86	29.01	**+5.15***
SVAMP	Gemma	17.34	18.70	**+1.36***
	Granite	16.95	18.58	**+1.62***
	LLaMA	17.60	18.88	**+1.28***
	Mistral	17.42	18.60	**+1.18***
	Qwen	17.80	18.57	+0.78
MultiArith	Gemma	17.24	17.29	+0.05
	Granite	17.39	17.11	-0.28
	LLaMA	17.25	17.15	-0.10
	Mistral	17.36	16.98	-0.38
	Qwen	17.19	19.26	**+2.07***
ASDiv	Gemma	16.70	16.87	+0.17
	Granite	16.10	17.37	**+1.27***
	LLaMA	16.39	18.43	**+2.04***
	Mistral	16.17	17.84	**+1.67***
	Qwen	16.70	17.58	+0.88

Table 1: Mean DLT complexity scores for correctly and incorrectly answered questions across datasets and models. Δ DLT is the difference. **Bolded values with * indicate statistically significant differences** ($p < 0.01$, Welch's t-test).[2]

5.1 Syntactic Complexity of Incorrect Questions

Table 1 reports the mean normalized DLT complexity scores on both sets of questions. GSM8K exhibits unanimously higher syntactic complexity scores on incorrectly answered questions across all models, with Welch's $p < 0.01$ in every case (full test statistics are provided in the supplementary material). On SVAMP, all deltas are positive, with four reaching statistical significance. MultiArith and ASDiv show weaker or inconsistent trends, with smaller or statistically insignificant differences.

5.2 Accuracy Gains from Rephrasing

We define performance improvement in terms of the change in accuracy after rephrasing, denoted by ΔA. Let Q_{total} denote the full set of questions, and $Q'_{\text{correct}} \subseteq Q_{\text{incorrect}}$ represent the set of previously incorrect questions that are now answered correctly after rephrasing. We compute:

$$\Delta A = \frac{|Q'_{\text{correct}}|}{|Q_{\text{total}}|} \tag{5}$$

Final model accuracy is then updated as:

$$\text{New Accuracy}(A) \tag{6}$$
$$= \text{Original Accuracy}(A_0) + \Delta A \tag{7}$$

[2]See Appendix Figure 5 for supporting visualizations.

Model	GSM8K				SVAMP				MultiArith				ASDiv			
	A_0	ΔA	A	#Recovered	A_0	ΔA	A	#Recovered	A_0	ΔA	A	#Recovered	A_0	ΔA	A	#Recovered
Gemma-7B	37.76	8.26	46.02	109	61.71	7.14	68.85	50	77.62	4.76	82.38	20	59.61	9.11	68.72	210
Granite-7B	24.03	11.68	35.71	154	44.43	11.86	56.29	83	50.00	15.48	65.48	65	64.08	9.68	73.75	223
LLaMA-8B	75.44	7.81	83.25	103	79.86	4.00	83.86	28	94.76	3.57	98.33	15	81.43	5.08	86.51	117
Mistral-7B	48.29	11.45	59.74	151	63.29	8.29	71.57	58	70.71	14.52	85.24	61	47.25	12.10	59.35	279
Qwen-7B	84.53	4.32	88.86	57	92.43	1.43	93.86	10	97.14	0.48	97.62	2	91.76	1.82	93.58	42

Table 2: Accuracy improvements and number of recovered answers from syntactic restructuring across GSM8K, SVAMP, MultiArith, and ASDiv. A_0 is baseline accuracy, ΔA is improvement after rephrasing, A is final accuracy, and #Recovered denotes incorrect answers corrected by rephrasing.

This formulation quantifies the overall gain attributable to syntactic restructuring, allowing us to isolate its impact.

Table 2 reports the accuracy improvements and the number of recovered answers from syntactic restructuring. All models improve on GSM8K and SVAMP, with lower-performing models (e.g., Gemma, Granite) showing the greatest relative gains. Improvements are less consistent on MultiArith and ASDiv, where most models already achieve high baseline accuracy or rephrasing yields fewer recoveries.

We observe that syntactic restructuring is most impactful on datasets with more narrative or structurally varied question phrasing (e.g., GSM8K, SVAMP), suggesting that syntactic mismatch contributes to model failures in these settings. Recovery counts ranged from 2 (Qwen on MultiArith) to 210 (Gemma on ASDiv), with rephrasing improving accuracy by as much as 15.5 % (Granite on MultiArith).

These findings provide strong empirical support for our hypothesis that LLMs fail on syntactically unfamiliar problems, and that rephrasing toward familiar structures mitigates these errors.

6 Further Analysis

The section provides further analysis regarding how syntactic structure influences LLM reasoning behavior. It examines four key dimensions: First, we show that elevated syntactic complexity, measured using DLT, predicts failure on narrative math tasks. Second, we demonstrate that rephrasing these complex questions into syntactically familiar forms improves model accuracy, supporting an interpretation of failure as schema misalignment. Third, we analyze these errors in light of cognitive theory, arguing that LLMs overapply familiar strategies to structurally novel inputs, a form-function misalignment. Finally, we outline implications for robustness and generalization, proposing syntax-aware interventions and cognitively grounded training approaches.

6.1 Syntactic Complexity Predicts Failure on Narrative Math Tasks

The results from Table 1 show that syntactic complexity, as measured by DLT scores, is positively associated with model failures, particularly on GSM8K and SVAMP. On GSM8K, all five LLMs exhibit statistically significant increases in complexity on incorrectly answered questions. On SVAMP, all deltas are positive, though only four reach statistical significance. For ASDiv, all models again show positive deltas, with three of them statistically significant. In contrast, the pattern is weaker on MultiArith, where only two of the five models show positive deltas and just one achieves significance. These results support the hypothesis that LLMs are sensitive to structural features of problem statements, especially on narrative-heavy datasets like GSM8K and SVAMP. By contrast, MultiArith's more uniform, low-complexity phrasing likely shifts the source of failure away from syntactic burden and toward reasoning depth.

These findings are consistent with the previously outlined phenomenon of *syntactic induction*, in which models perform worse on problems that deviate from familiar surface forms. In our experiments, LLMs consistently exhibited higher failure rates on syntactically complex questions, particularly when those forms differed structurally from common patterns. This suggests that model predictions are sensitive to surface structure, and that unfamiliar phrasing can impair accuracy even when underlying reasoning demands remain constant.

From a cognitive perspective, these errors reflect a failure in structural fluency. The DLT framework quantifies this fluency as a function of integration cost, storage cost, and discourse load. Elevated scores among failure cases suggest that LLMs, like human solvers, are vulnerable to breakdowns in processing when these syntactic burdens accumulate beyond an internalized threshold.

6.2 Rephrasing as Schema Alignment

Table 2 demonstrates that rephrasing structurally complex questions into syntactically familiar forms yields substantial accuracy improvements. This pattern is most pronounced on GSM8K and SVAMP, particularly among lower-performing models such as Gemma and Granite. While high-performing models like LLaMA and Qwen show smaller deltas, they also exhibit measurable gains, supporting the interpretation that rephrasing facilitates access to familiar problem-solving patterns across models.

These improvements reinforce the interpretation of failure as a *schema alignment problem*. According to prior work in cognitive psychology, solvers often rely on surface cues to activate latent problem schemas. When surface form is misaligned with internal expectations, reasoning may fail despite latent competence. Rephrasing appears to bridge this gap, effectively priming models to recognize the underlying problem structure.

The consistency of this effect across architectures suggests that the phenomenon is not model-specific but a general feature of current LLM design.

6.3 Syntax Cues the Wrong Strategy: Evidence of Form-Function Misalignment

Our findings reflect a consistent pattern: models often fail when questions are phrased in structurally unfamiliar ways, even if the underlying reasoning task remains the same. This aligns with cognitive accounts of human error, such as those described by Ben-Zeev (1998), in which solvers misapply familiar procedures to novel input formats.

While LLMs are not rule-based agents in the human sense, our results suggest that they similarly rely on surface-level cues to guide problem-solving behavior. When syntactic structure deviates from familiar patterns, models are more likely to generate incorrect responses, even when they demonstrate competence on simpler or canonical formulations of the same task.

That even high-performing models benefit from syntactic restructuring indicates that many failures are not due to limitations in arithmetic ability per se, but rather in applying the right strategy under structural variation. This points to a challenge beyond learning correct computations: models must also determine *how* and *when* to apply learned behaviors, a process that appears sensitive to variations in syntactic form.

6.4 Toward Syntax-Aware Generalization

These results carry several implications for improving LLM robustness and interpretability. First, they emphasize the importance of training or prompting models to abstract beyond surface form. Sensitivity to syntactic variation can limit generalization, even in domains where the underlying reasoning is sound.

Second, our analysis highlights the potential of syntax-aware interventions. By measuring DLT complexity and selectively rephrasing high-complexity inputs, systems could anticipate and mitigate failure without retraining. This suggests a role for lightweight, dynamic preprocessing pipelines in real-world deployments.

Finally, our findings suggest promising directions for future research (1) **Syntactic curriculum learning**: Gradually exposing models to varied syntactic structures during training to improve generalization under structural variation. (2) **Schema-guided error analysis**: Building error taxonomies based on syntactic mismatches to inform evaluation and debugging.

Overall, our analysis suggests that syntactic induction is a significant source of LLM failure in math reasoning tasks. By quantifying structural complexity and aligning input form with successful patterns, we can better anticipate and mitigate a subset of reasoning errors rooted in input formulation.

Crucially, our findings suggest that syntactic complexity is not merely correlated with failure but may play a mediating role in model performance on structurally complex inputs. This highlights structurally guided rephrasing as a lightweight and scalable strategy for recovering from such errors, without modifying model weights or requiring additional supervision.

7 Conclusion

This work investigated how syntactic structure influences the reasoning behavior of LLMs on mathematical problems. Across four benchmarks: GSM8K, SVAMP, MultiArith, and ASDiv, we found that LLMs systematically fail on syntactically complex inputs, despite their semantic simplicity. These failures were reliably predicted by elevated Dependency Locality Theory (DLT) scores and mitigated through targeted syntactic rephrasing.

Our findings demonstrate that many reasoning errors stem not from a lack of mathematical competence, but from `syntactic induction failures`, a tendency to misapply known solution strategies when surface structure deviates from training priors. Rephrasing misaligned questions into syntactically familiar forms improved accuracy across all models, with gains particularly notable in lower-performing systems like Gemma and Granite. This supports the view, rooted in cognitive science, that schema activation in both humans and LLMs is highly sensitive to surface cues.

By framing these errors within a rule-based taxonomy and formalizing complexity through DLT, we offer a structured explanation for inductive failure in LLMs. Rather than viewing mistakes as isolated or stochastic, we show they are predictable, syntax-sensitive, and recoverable through lightweight interventions.

Future Directions This work opens several directions for enhancing LLM robustness and interpretability. We highlight:

- **Syntactic diagnostic tools**: To identify schema misalignment based on DLT complexity or parse structure.
- **Structure-aware input representations**: Leveraging dependency graphs or programmatic abstractions to make problem structure more accessible.
- **Failure-aware training curricula**: Introducing controlled syntactic variation to encourage generalization beyond form-driven heuristics.

While our experiments focus on mathematical benchmarks, the implications are broader. Syntactic induction failures may underlie reasoning brittleness across domains. Addressing these failures offers a path toward LLMs that reason more like human experts: flexibly, structurally, and with awareness of when form does, and does not, align with function.

8 Ethics Statement

Our experiments were conducted on publicly available mathematical reasoning datasets, which do not contain sensitive personal data or pose identifiable risks to individuals or groups. The work does not involve human subjects or data collection. No known ethical risks were introduced, and all referenced work is properly cited and respected under academic norms.

9 Limitations

This study focuses on mathematical word problems and may not generalize to other domains. We evaluate only final-answer accuracy, without analyzing intermediate reasoning.

References

Robert B. Ashlock. 2002. *Error patterns in computation*, 8th ed. edition. Merrill PH, New Jersy.

Guangsheng Bao, Hongbo Zhang, and et al Wang. 2025. How Likely Do LLMs with CoT Mimic Human Reasoning? In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7831–7850, Abu Dhabi, UAE. Association for Computational Linguistics.

Talia Ben-Zeev. 1998. Rational errors and the mathematical mind. *Review of General Psychology*, 2(4):366–383.

Tom Brown and et al Mann. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Michelene T H Chi and Feltovich et al. 1981. Categorization and representation of physics problems by experts and novices. *Cogn. Sci.*, 5(2):121–152.

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. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Edward Gibson. 1998. Linguistic complexity: locality of syntactic dependencies. *Cognition*, 68(1):1–76.

Edward Gibson. 2000. The Dependency Locality Theory: A distance-based theory of linguistic complexity. In *Image, language, brain: Papers from the first mind articulation project symposium*, pages 95–126, Cambridge, MA. MIT Press.

Jia He, Mukund Rungta, David Koleczek, Arshdeep Sekhon, Franklin X. Wang, and Sadid Hasan. 2024. Does Prompt Formatting Have Any Impact on LLM Performance? *arXiv preprint*. ArXiv:2411.10541 [cs].

Dan Alvin Hinsley, J. R. Hayes, and Herbert A. Simon. 1977. From words to equations meaning and representation in algebra word problems.

K J Holyoak and K Koh. 1987. Surface and structural similarity in analogical transfer. *Mem. Cognit.*, 15(4):332–340.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, Yue Wu, Ming Yin, Shange Tang, Yangsibo Huang, Chi Jin, Xinyun Chen, Chiyuan Zhang, and Mengdi Wang. 2025. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations. *Preprint*, arXiv:2502.06453.

Albert Q. Jiang and Alexandre Sablayrolles et al. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Katie Kang, Eric Wallace, Claire Tomlin, Aviral Kumar, and Sergey Levine. 2024. Unfamiliar Finetuning Examples Control How Language Models Hallucinate. *arXiv preprint*. ArXiv:2403.05612 [cs].

Annette Karmiloff-Smith. 1986. From meta-processes to conscious access: Evidence from children's metalinguistic and repair data. *Cognition*, 23(2):95–147.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. *Preprint*, arXiv:2306.03872.

Inbal Magar and Roy Schwartz. 2022. Data Contamination: From Memorization to Exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165, Dublin, Ireland. Association for Computational Linguistics.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the ACL*, pages 975–984, Online. Association for Computational Linguistics.

Mayank Mishra and Matt Stallone et al. 2024. Granite code models: A family of open foundation models for code intelligence. *Preprint*, arXiv:2405.04324.

L R Novick. 1988. Analogical transfer, problem similarity, and expertise. *J. Exp. Psychol. Learn. Mem. Cogn.*, 14(3):510–520.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of NAACL 2021: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Brian H. Ross. 1984. Remindings and their effects in learning a cognitive skill. *Cognitive Psychology*, 16(3):371–416.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. Nlp evaluation in trouble: On the need to measure llm data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10776–10787, Singapore. Association for Computational Linguistics.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561.

Saurabh Srivastava, Annarose M. B, Anto P. V, Shashank Menon, Ajay Sukumar, Adwaith Samod T, Alan Philipose, Stevin Prince, and Sooraj Thomas. 2024. Functional Benchmarks for Robust Evaluation of Reasoning Performance, and the Reasoning Gap. *arXiv preprint*. ArXiv:2402.19450 [cs].

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2025. Chain of thoughtlessness? an analysis of cot in planning. *Preprint*, arXiv:2405.04776.

Gemma Team, Morgane Riviere, and Shreya Pathak et al. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

An Yang, Baosong Yang, and Binyuan Hui et al. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Hugh Zhang, Jeff Da, and Lee et al. 2024. A careful examination of large language model performance on grade school arithmetic. In *Advances in Neural Information Processing Systems*, volume 37, pages 46819–46836. Curran Associates, Inc.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *Preprint*, arXiv:2412.06559.

Yuhang Zhou, Paiheng Xu, Xiaoyu Liu, Bang An, Wei Ai, and Furong Huang. 2024. Explore Spurious Correlations at the Concept Level in Language Models for Text Classification. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 478–492, Bangkok, Thailand. Association for Computational Linguistics.

A Appendix

A.1 DLT Rephrasing Algorithm (Proposed)

This subsection outlines a proposed procedure for preemptively identifying syntactically complex math word problems and evaluating the impact of rephrasing. While this algorithm was not executed in the main paper due to time constraints, it was designed to support future ablation studies. The pipeline filters questions based on elevated DLT complexity, applies dependency-guided rephrasing, and evaluates accuracy before and after restructuring. This formalization supports reproducibility and highlights a possible direction for proactive syntax-aware model evaluation.

Algorithm 1 DLT-Guided Rephrasing and Accuracy Evaluation

Require: Dataset $\mathcal{D} = \{(q_i, a_i)\}_{i=1}^N$ of questions and answers
Require: Normalized DLT complexity scoring function $\mathrm{DLT}(q)$
Require: Rephrasing function $\mathrm{Rephrase}(q)$
Require: Evaluation function $\mathrm{Accuracy}(\mathcal{Q}, \mathcal{A})$
1: Compute scores: $\{s_i = \mathrm{DLT}(q_i)\}_{i=1}^N$
2: Set threshold τ as the 75th percentile of $\{s_i\}$
3: Define $\mathcal{D}_{\text{complex}} = \{(q_i, a_i) \mid s_i \geq \tau\}$
4: For each (q_i, a_i) in $\mathcal{D}_{\text{complex}}$, compute $q'_i = \mathrm{Rephrase}(q_i)$
5: Evaluate original accuracy:

$$\mathrm{Acc}_{\text{orig}} = \mathrm{Accuracy}\left(\{q_i\}, \{a_i\}\right)$$

6: Evaluate rephrased accuracy:

$$\mathrm{Acc}_{\text{reph}} = \mathrm{Accuracy}\left(\{q'_i\}, \{a_i\}\right)$$

7: Compute improvement: $\Delta = \mathrm{Acc}_{\text{reph}} - \mathrm{Acc}_{\text{orig}}$
8: **return** τ, Δ

A.2 DLT Complexity Statistical Significance

This section provides full statistical results and supporting visualizations for the DLT complexity gaps reported in Table 1 of the main paper. The table below contains Welch's t-statistics and p-values for each dataset–model pair. Following the table, we include full boxplots of DLT scores by outcome (correct vs. incorrect) across all models and datasets. These visualizations offer a more detailed view of score distributions, variance, and effect sizes.

Table 3: Welch's t-test results comparing DLT complexity for correctly vs. incorrectly answered questions.

Model	ASDiv		GSM8K		MultiArith		SVAMP	
	t-stat.	p-val.	t-stat.	p-val.	t-stat.	p-val.	t-stat.	p-val.
Gemma	-0.66	0.51	-6.97	< 0.0001	-0.18	0.86	-3.94	< 0.0001
Granite	-5.29	< 0.0001	-6.22	< 0.0001	1.16	0.249	-4.87	< 0.0001
LLaMA	-5.92	< 0.0001	-7.44	< 0.0001	0.28	0.861	-2.94	0.0037
Mistral	-6.47	< 0.0001	-6.83	< 0.0001	1.58	0.116	-3.42	0.0007
Qwen	-1.76	0.08	-6.66	< 0.0001	-2.52	0.0275	-1.13	0.2649

A.3 Manual Evaluation of Rephrased Questions

To assess the quality and effectiveness of our syntactic rephrasing method, we conducted a manual evaluation. We selected 10 representative (original, rephrased) question pairs sampled from GSM8K, SVAMP, MultiArith, and ASDiv. Each pair was reviewed by an annotator along three criteria:

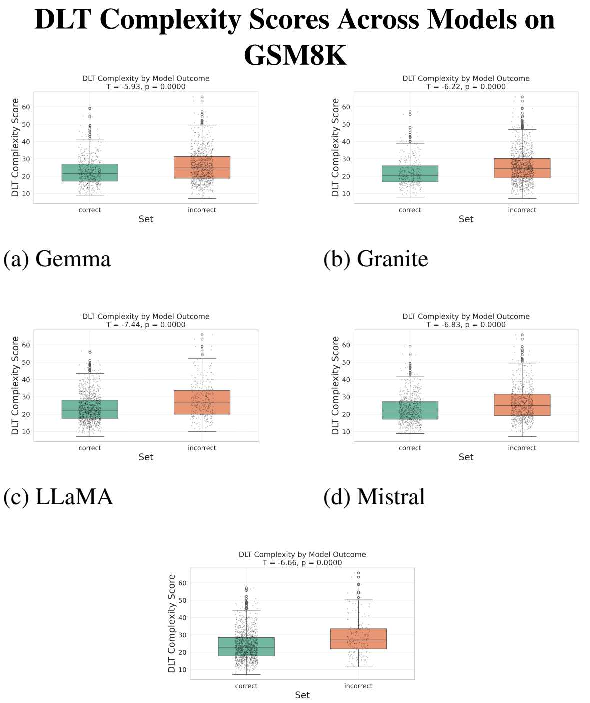
- **Semantic Match**: Does the rephrased version preserve the original problem's meaning?
- **Structural Simplification**: Does the rephrased version reduce syntactic complexity (e.g., fewer clauses, flatter dependencies)?
- **Answered Correctly**: Did the model originally answer incorrectly but succeed after rephrasing?

All 10 examples were rated as preserving semantic fidelity while simplifying structure, and all were answered correctly by the model post-rephrasing. These results reinforce the claim that syntactic restructuring can reduce complexity while maintaining problem intent, allowing models to succeed on previously failed inputs.

Table 5 summarizes the outcomes for each evaluated example.

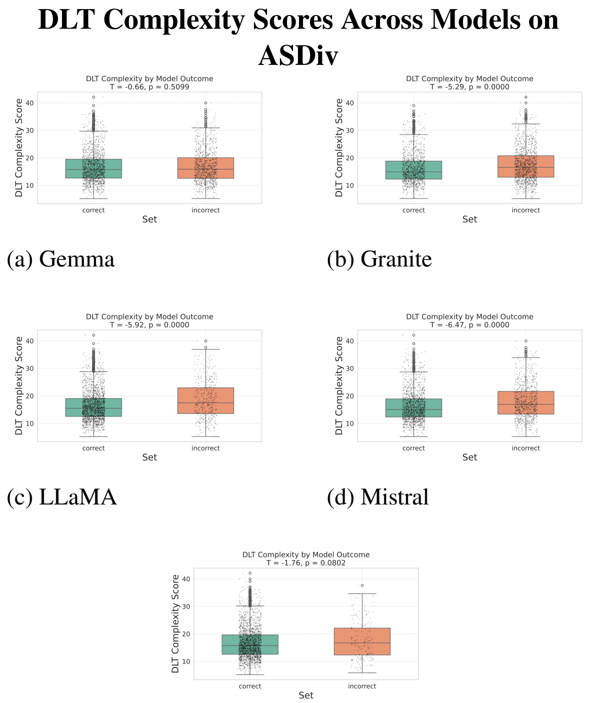
A.4 LLM Evaluation Framework

This section lists the models and decoding parameters used in our experiments. Table 4 provides full details for both the math QA models and the rephrasing model. All math questions were evaluated in a zero-shot setting using greedy decoding (temperature = 0, no sampling). For rephrasing, we used LLaMA-3 with mild sampling settings to introduce syntactic variation while preserving semantic intent. These parameters were fixed across all datasets to ensure consistency and reproducibility.



(a) Gemma (b) Granite (c) LLaMA (d) Mistral (e) Qwen

Figure 5: DLT complexity scores by model outcome (correct vs. incorrect) across five LLMs on GSM8K. In each subplot, incorrectly answered questions (orange) exhibit higher mean complexity and greater variance than correct ones (green). Welch's t-statistics and p-values confirm these differences are statistically significant.



(a) Gemma (b) Granite (c) LLaMA (d) Mistral (e) Qwen

Figure 6: DLT complexity scores by model outcome (correct vs. incorrect) across five LLMs on ASDiv.

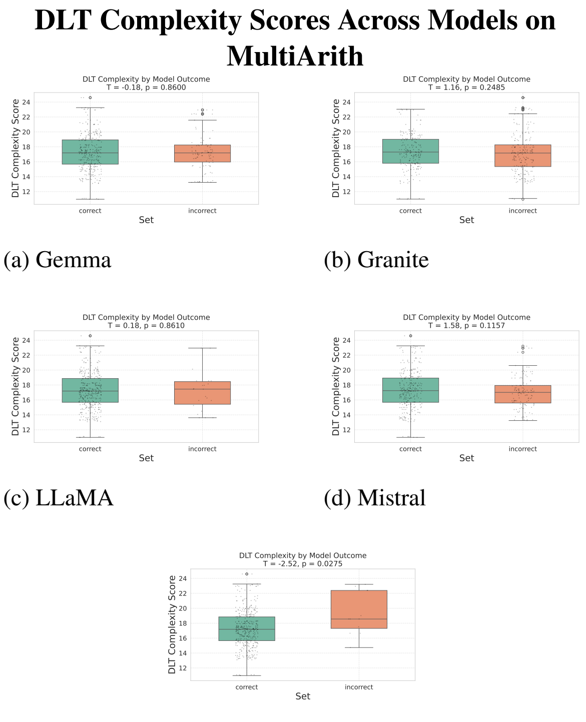
A.5 Cognitive Psychology: Rational Errors

To contextualize our findings within broader theories of reasoning failure, we draw on insights from cognitive psychology and mathematical pedagogy. Specifically, we reference the work of Ben-Zeev (1998), who frames many student errors in mathematics not as random mistakes, but as *rational errors*. systematic overgeneralizations of otherwise valid strategies.

Figure 9 illustrates this framework. The top panel shows a classic subtraction mistake: subtracting each digit in place-value order without accounting for borrowing. This type of mistake is not due to irrationality but reflects a learner's internalization of an overly simplified rule. The bottom panel presents a taxonomy of inductive failure modes, such as *syntactic induction* and *semantic induction*, which describe how solvers may misapply surface-level patterns or real-world analogies inappropriately.

These mechanisms are highly relevant to our analysis of LLM behavior. Our experiments show that LLMs often fail on syntactically novel questions not because they lack competence, but because they overapply strategies learned from structurally familiar formsm precisely the type of error Ben-Zeev characterizes as rational. In particular, what we term `syntactic induction` failures in LLMs echoes this cognitive framing, highlighting deep parallels between human and model error patterns.

We include these diagrams to situate our findings in a well-established theory of rule-based reasoning errors and to support our claim that LLM failures are often structured, interpretable, and attributable to form-function misalignment rather than arbitrary noise.

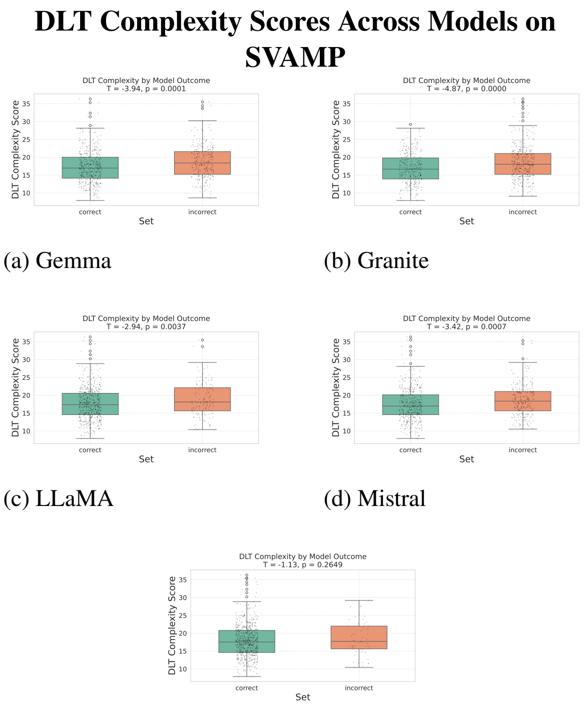


(a) Gemma (b) Granite

(c) LLaMA (d) Mistral

(e) Qwen

Figure 7: DLT complexity scores by model outcome (correct vs. incorrect) across five LLMs on Multiarith.



(a) Gemma (b) Granite

(c) LLaMA (d) Mistral

(e) Qwen

Figure 8: DLT complexity scores by model outcome (correct vs. incorrect) across five LLMs on SVAMP.

Model	Params	Max	Temp	Top-p	Sample
LLaMA-3	8B	8192	0.0	1.0	False
Mistral	7B	8192	0.0	1.0	False
Qwen2.5	7B	8192	0.0	1.0	False
Gemma	7B	8192	0.0	1.0	False
Granite	7B	8192	0.0	1.0	False

(a) Math QA models and decoding hyperparameters. All models use greedy decoding (temperature = 0, no sampling).

Rephrasing Model	Params	Max	Temp	Top-p	Sample
LLaMA-3	8B	8192	0.1	0.9	True

(b) Rephrasing model used for syntactic restructuring.

Table 4: LLMs used in experiments.

Figure 9: Correct borrowing (left) vs. a common subtraction mistake (right). Students without instruction on borrowing often overgeneralize the principle of subtracting smaller from larger numbers.

Dataset	Model	Original Question	Rephrased Question	Semantic Match	Simplified	Answered Correctly
GSM8K	LLaMA-8B	A company pays each of its employees $600 in a month. The company has a policy of increasing the salaries of each of its employees by 10% of the initial salary every year for those who've stayed in the company for five years. If Sylvie just clocked 5 years in the company last December, what's her annual salary after three more years of service?	A company pays its employees $600 per month. The company has a policy of increasing salaries by 10% of the initial salary every year for employees who have stayed with the company for five years. Sylvie just completed five years of service last December. What is her annual salary after three more years of service?	✓	✓	✓
MultiArith	Mistral-7B	Roger had 68 dollars. If he spent 47 bucks on a new game, how many 7 dollar toys could he buy with the money he had left?	Roger has 68 dollars. He spends 47 dollars on a new game. How many 7-dollar toys can he buy with the money he has left?	✓	✓	✓
ASDiv	Qwen2.5-7B	Andrew's 4 friends decided to bring food as well. If each of them brought 4 slices of pizza, how many slices of pizza do they have in total?	Andrew's 4 friends decide to bring food as well. They each bring 4 slices of pizza. How many slices of pizza do they have in total?	✓	✓	✓
GSM8K	Gemma-7B	Brandon's iPhone is four times as old as Ben's iPhone. Ben's iPhone is two times older than Suzy's iPhone. If Suzy's iPhone is 1 year old, how old is Brandon's iPhone?	Brandon's iPhone is four times as old as Ben's iPhone. Ben's iPhone is two times older than Suzy's iPhone, which is 1 year old. How old is Brandon's iPhone?	✓	✓	✓
SVAMP	Granite-7B	Edward spent $6 to buy 2 books each book costing him the same amount of money. Now he has $12. How much did each book cost?	Edward spent $6 to buy 2 books, each costing the same amount. Now he has $12. How much did each book cost?	✓	✓	✓
SVAMP	Granite-7B	Billy sells DVDs. He has 8 customers on Tuesday. His first 3 customers buy one DVD each. His next 2 customers buy 2 DVDs each. His last 3 customers don't buy any DVDs. How many DVDs did Billy sell on Tuesday?	Billy sells DVDs to 8 customers on Tuesday. His first 3 customers buy one DVD each, and his next 2 customers buy 2 DVDs each. His last 3 customers don't buy any DVDs. How many DVDs does Billy sell on Tuesday?	✓	✓	✓
ASDiv	Mistral-7B	It's Rachel's birthday. Her parents wanted her to have fun so they went to the circus that happened to be in town that day. Upon arriving at the circus, they went to the ticket booth and asked how much each ticket cost. If each ticket costs $44.00 and they bought 7 tickets, how much money did they spend on tickets?	Rachel's parents take her to the circus on her birthday. They buy 7 tickets, each costing $44.00. How much money do they spend on tickets?	✓	✓	✓
MultiArith	Qwen-7B	Will invited 9 friends to a birthday party, but 4 couldn't come. If he wanted to buy enough cupcakes so each person could have exactly 8, how many should he buy?	Will invites 9 friends to a birthday party, but 4 can't come. If he wants to give each person 8 cupcakes, how many cupcakes should he buy?	✓	✓	✓
GSM8K	LLaMA-8B	Jerome had 4 friends who came to visit him on a certain day. The first friend pressed on the doorbell 20 times before Jerome opened, the second friend pressed on the doorbell 1/4 times more than Jerome's first friend. The third friend pressed on the doorbell 10 times more than the fourth friend. If the fourth friend pressed on the doorbell 60 times, how many doorbell rings did the doorbell make?	Jerome has 4 friends visiting him, and the first friend rang the doorbell 20 times before Jerome opened it. The second friend rang the doorbell 1/4 times more than the first friend, the third friend rang it 10 times more than the fourth friend, and the fourth friend rang it 60 times. How many times did the doorbell ring in total?	✓	✓	✓
GSM8K	LLaMA-8B	Jam has three boxes full of pencils and 2 loose pencils which give a total of 26 pencils. If her sister, Meg, has 46 pencils, how many boxes do Jam and Meg need to store all their pencils?	Jam has three boxes of pencils and 2 loose pencils, which together total 26 pencils. Her sister, Meg, has 46 pencils. How many boxes do Jam and Meg need to store all their pencils?	✓	✓	✓

Table 5: Manual evaluation of rephrased questions. A checkmark indicates success for each criterion.

STEP-KTO: Optimizing Mathematical Reasoning through Stepwise Binary Feedback

Yen-Ting Lin[*†] **Di Jin**[*] **Tengyu Xu**[*] **Tianhao Wu**[‡§] **Sainbayar Sukhbaatar**[‡]
Chen Zhu[*] **Yun He**[*] **Yun-Nung Chen**[†] **Jason Weston**[‡] **Yuandong Tian**[‡]
Arash Rahnama[*] **Sinong Wang**[*] **Hao Ma**[*] **Han Fang**[*]
[*]Meta GenAI [†]National Taiwan University [‡]Meta FAIR [§]UC Berkeley
ytl@ieee.org

Abstract

Large language models (LLMs) have recently demonstrated remarkable success in mathematical reasoning. Despite progress in methods like chain-of-thought prompting and self-consistency sampling, these advances often focus on final correctness without ensuring that the underlying reasoning process is coherent and reliable. This paper introduces STEP-KTO, a training framework that combines process-level and outcome-level binary feedback to guide LLMs toward more trustworthy reasoning trajectories. By providing binary evaluations for both the intermediate reasoning steps and the final answer, STEP-KTO encourages the model to adhere to logical progressions rather than relying on superficial shortcuts. Our experiments on challenging mathematical benchmarks show that STEP-KTO significantly improves both final answer accuracy and the quality of intermediate reasoning steps. For example, on the MATH-500 dataset, STEP-KTO achieves a notable improvement in Pass@1 accuracy over strong baselines. These results highlight the promise of integrating stepwise process feedback into LLM training, paving the way toward more interpretable and dependable reasoning capabilities.

1 Introduction

Large language models (LLMs) have recently shown remarkable capabilities in reasoning-intensive tasks such as coding (Chen et al., 2021; Li et al., 2022; Rozière et al., 2023) and solving complex mathematical problems (Shao et al., 2024; Azerbayev et al., 2024). Prompting strategies like chain-of-thought prompting (Nye et al., 2021; Wei et al., 2022; Kojima et al., 2022; Adolphs et al., 2022) and self-consistency sampling (Wang et al., 2023) enhance these models' final-answer accuracy by encouraging them to articulate intermediate reasoning steps. However, a significant issue remains: even when these methods boost final-answer correctness, the internal reasoning steps are often unreliable or logically inconsistent (Uesato et al., 2022; Lightman et al., 2024).

This discrepancy between correct final answers and flawed intermediate reasoning limits our ability to trust LLMs in scenarios where transparency and correctness of each reasoning stage are crucial (Lanham et al., 2023). For example, in mathematical problem-solving, a model might produce the right answer for the wrong reasons (Lyu et al., 2023; Zheng et al., 2024), confounding our understanding of its true capabilities (Turpin et al., 2023). To address this, researchers are increasingly emphasizing the importance of guiding models to produce not just correct final answers, but also verifiable and faithful step-by-step solution paths (Uesato et al., 2022; Shao et al., 2024; Setlur et al., 2024).

Prior work in finetuning has largely focused on outcome-level correctness, using outcome reward models to improve the probability of final-answer accuracy (Cobbe et al., 2021; Hosseini et al., 2024; Zhang et al., 2024). While effective, such an approach does not ensure that the intermediate reasoning steps are valid. Conversely, while process-level supervision through process reward models (PRMs) (Lightman et al., 2024; Wang et al., 2024; Luo et al., 2024) can guide models to follow correct reasoning trajectories, prior work has mainly used PRMs as a ranking method rather than a way to provide stepwise feedback. As a result, relying solely on process-level supervision may lead models to prioritize step-by-step correctness without guaranteeing a correct final outcome.

In this paper, we introduce Stepwise Kahneman-Tversky-inspired Optimization (STEP-KTO), a training framework that integrates both process-level and outcome-level binary feedback to produce coherent and correct reasoning steps alongside high-quality final answers. Our approach evaluates each intermediate reasoning step against known correct patterns using a PRM, while simultaneously leveraging a rule-based reward signal for the final answer. To fuse these signals, we employ a Kahneman-Tversky-inspired value function (Tversky and Kahneman, 2016; Ethayarajh et al., 2024) that emphasizes human-like risk and loss

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 15–33

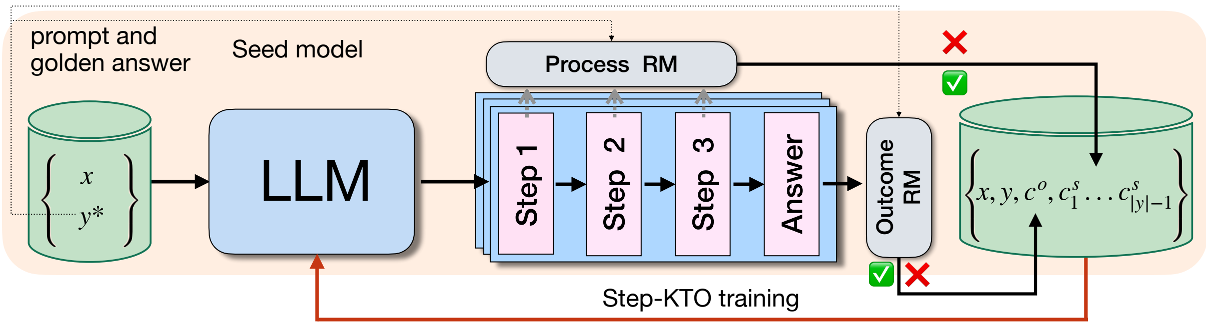


Figure 1: **STEP-KTO Training Process.** Given a dataset of math problems (left), a language model (LLM) produces both reasoning steps and a final answer. Each intermediate reasoning step is evaluated by a process reward model (Process RM), and the final answer is assessed by an outcome reward model (Outcome RM). The binary feedback signals from both levels (outcome-level correctness c^o and stepwise correctness c_h^s) are recorded together with the input (x) and the model's response (y) §2.1. These signals are then used to compute the STEP-KTO loss, guiding the LLM to not only produce correct final answers but also maintain coherent and correct reasoning steps §2.3. Through multiple iterations of this training process §2.4, the model progressively improves both its stepwise reasoning and final answer accuracy.

aversion, encouraging models to gradually correct their reasoning and avoid errors. The result is a training objective that aligns the entire reasoning trajectory with verified solutions while ensuring that final correctness remains a top priority.

Figure 1 illustrates the STEP-KTO pipeline. We start with a base LLM and repeatedly refine it through iterative training. At each iteration, the PRM provides step-level binary feedback that helps the model navigate correct solution paths, while the outcome-level binary feedback ensures that the final answer is correct. The Kahneman-Tversky-inspired value function transforms these binary signals into guidance that progressively reduces errors in the chain-of-thought. Over successive rounds, STEP-KTO yields systematically more accurate intermediate reasoning steps and steadily improves the final-answer accuracy.

We evaluate STEP-KTO on challenging mathematical reasoning benchmarks including MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), AMC23 (MAA, 2023), and AIME24(MAA, 2024). Our experiments show that incorporating both process-level and outcome-level signals leads to substantial improvements over state-of-the-art baselines that rely solely on final-answer supervision. For example, on MATH-500, STEP-KTO improves Pass@1 accuracy from 53.4% to 63.2%, while also producing more coherent and trustworthy step-by-step reasoning. Moreover, iterative training with STEP-KTO achieves cumulative gains, demonstrating that balancing process- and outcome-level feedback refines reasoning quality over time. In summary, our key contributions are:

- We propose STEP-KTO, a novel finetuning framework that combines process-level and outcome-level feedback, encouraging both correct final answers and faithful step-by-step reasoning.
- We show that iterative training with STEP-KTO yields consistent cumulative improvements, showing the effectiveness of combined process-level and outcome-level feedback in refining LLM reasoning.
- We demonstrate that STEP-KTO surpasses state-of-the-art baselines on multiple math reasoning tasks, delivering higher accuracy (63.2% vs 53.4% Pass@1 on MATH-500) and more reliable intermediate solutions.

2 Methodology

2.1 Problem Setup and Notation

We adopt the notation and setup similar to Setlur et al. (2024). Let $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}^\star_{\boldsymbol{x}_i})\}_i$ be a dataset of math problems, where each problem $\boldsymbol{x} \in \mathcal{X}$ has an associated ground-truth solution sequence $\boldsymbol{y}^\star_{\boldsymbol{x}} = (s^\star_1, s^\star_2, \dots, s^\star_{|\boldsymbol{y}^\star|}) \in \mathcal{Y}$. A policy model π_θ, parameterized by θ, generates a response sequence $\boldsymbol{y} = (s_1, s_2, \dots, s_{|\boldsymbol{y}|})$ autoregressively given the problem $\boldsymbol{x}$, where each step s_h is a reasoning step separated by a special token (e.g., "## Step").

The correctness of the final answer $\boldsymbol{y}$ can be automatically determined by a rule-based correctness function $\mathrm{Regex}(\boldsymbol{y}, \boldsymbol{y}^\star_{\boldsymbol{x}}) \in \{0, 1\}$, which compares the model's final derived answer to the ground-truth final answer (Hendrycks et al., 2021). The model's final answer is explicitly denoted using a special format in the final step $s_{|\boldsymbol{y}|}$, such as boxed$\{\cdot\}$, allowing the correctness function to easily extract and verify it. Our primary objective is to improve the expected correctness of the final answer:

$$\mathbb{E}_{\boldsymbol{x}\in\mathscr{D},\ \boldsymbol{y}\sim\pi_\theta(\cdot|\boldsymbol{x})}[\text{Regex}(\boldsymbol{y}, \boldsymbol{y}^\star_{\boldsymbol{x}})].$$

Ensuring a correct final answer does not guarantee logically sound intermediate reasoning. To address this, we incorporate a stepwise binary correctness signal $\text{Prm}(\boldsymbol{x}, \boldsymbol{y}^\star_{\boldsymbol{x}}, s_h) \in \{0, 1\}$ for each reasoning step s_h. Unlike the final-answer correctness Regex, this signal directly measures whether each intermediate step is locally valid and aligns with proper problem-solving principles, without strictly mirroring the reference solution steps. We obtain these stepwise correctness evaluations by prompting an LLM (`Llama-3.1-70B-Instruct`) as our process reward model (PRM), following the structured template in Appendix D. In summary, we have two levels of binary signals:

- **Outcome feedback**: $\text{Regex}(\boldsymbol{y}, \boldsymbol{y}^\star_{\boldsymbol{x}}) \in \{0, 1\}$ indicates if the final answer derived from $\boldsymbol{y}$ is correct.
- **Stepwise feedback**: $\text{Prm}(\boldsymbol{x}, \boldsymbol{y}^\star_{\boldsymbol{x}}, s_h) \in \{0, 1\}$ indicates if the intermediate reasoning step s_h is correct.

Our goal is to integrate both of these signals into the training objective of π_θ. By doing so, we guide the model to produce not only correct final answers but also to maintain correctness, coherence, and reliability throughout its reasoning trajectory. This integrated approach will be formalized through the STEP-KTO framework.

2.2 KTO Background

KTO (Ethayarajh et al., 2024) aims to align a policy π_θ with binary feedback using a Kahneman-Tversky-inspired value function (Tversky and Kahneman, 2016). Rather than maximizing the log-likelihood of preferred outputs or directly using reinforcement learning, KTO defines a logistic value function that is risk-averse for gains and risk-seeking for losses.

The original KTO loss focuses on the final-answer level. Let:

$$r_\theta(x, y) = \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}, \tag{1}$$

$$z_0 = \text{KL}\big(\pi_\theta(y' \mid x) \,\|\, \pi_{\text{ref}}(y' \mid x)\big), \tag{2}$$

$$v(x, y) = \begin{cases} \lambda_D\, \sigma\big(\beta(r_\theta(x, y) - z_0)\big) \\ \qquad\qquad \text{if } \text{Regex}(\boldsymbol{y}, \boldsymbol{y}^\star_{\boldsymbol{x}}) = 1, \\ \lambda_U\, \sigma\big(\beta(z_0 - r_\theta(x, y))\big) \\ \qquad\qquad \text{if } \text{Regex}(\boldsymbol{y}, \boldsymbol{y}^\star_{\boldsymbol{x}}) = 0. \end{cases} \tag{3}$$

Here, π_{ref} is a reference policy (typically the initial model checkpoint) that provides a baseline for comparison, σ is the logistic function, $\beta > 0$ controls risk aversion, and λ_D, λ_U are weighting coefficients. The z_0 term, where y' denotes an arbitrary output sequence, serves as a reference point to ensure balanced optimization. The KTO loss at the outcome level is:

$$L_{\text{KTO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{x,y\sim D}[\lambda_y - v(x, y)], \tag{4}$$

where $\lambda_y = \lambda_D$ if $\text{Regex}(\boldsymbol{y}, \boldsymbol{y}^\star_{\boldsymbol{x}}) = 1$ and $\lambda_y = \lambda_U$ if $\text{Regex}(\boldsymbol{y}, \boldsymbol{y}^\star_{\boldsymbol{x}}) = 0$.

2.3 STEP-KTO

While KTO ensures correctness of final answers, many reasoning tasks require validity at each intermediate step. We extend KTO by incorporating stepwise binary feedback $\text{Prm}(\boldsymbol{x}, \boldsymbol{y}^\star_{\boldsymbol{x}}, s_h)$ to assess the quality of each reasoning step. We begin by defining an *implied reward* at the step level:

$$r_\theta(x, s_h) = \log \frac{\pi_\theta(s_h \mid x, s_{<h})}{\pi_{\text{ref}}(s_h \mid x, s_{<h})}.$$

This quantity can be viewed as the incremental advantage of producing step s_h under π_θ compared to π_{ref}. It captures how much more (or less) reward is implied by choosing s_h over the reference model's baseline likelihood, conditioned on the same context $(x, s_{<h})$. Next, we introduce a stepwise KL baseline:

$$z_0^{(step)} = \text{KL}\big(\pi_\theta(s'_h \mid x, s'_{<h}) \,\|\, \pi_{\text{ref}}(s'_h \mid x, s'_{<h})\big).$$

Analogous to z_0 at the outcome level, $z_0^{(step)}$ serves as a local reference point. It prevents the model from gaining reward merely by diverging from the reference and ensures that improvements are grounded in genuine reasoning quality. Given the binary stepwise feedback $\mathrm{Prm}(\boldsymbol{x}, \boldsymbol{y}^{\star}_{\boldsymbol{x}}, s_h)$, we define a value function that parallels the outcome-level case. If a step s_h is deemed stepwise-desirable, the model should increase its implied reward $r_\theta(x, s_h)$ relative to $z_0^{(step)}$ (Huang and Chen, 2024). Conversely, if s_h is stepwise-undesirable, the model is encouraged to lower that implied reward. Formally:

$$v^{(step)}(x, s_h) = \begin{cases} \lambda_D^{(step)}\, \sigma\big(\beta_{\text{step}}(r(x, s_h) - z_0^{(step)})\big) & \text{if } \mathrm{Prm}(\boldsymbol{x}, \boldsymbol{y}^{\star}_{\boldsymbol{x}}, s_h) = 1, \\ \lambda_U^{(step)}\, \sigma\big(\beta_{\text{step}}(z_0^{(step)} - r(x, s_h))\big) & \text{if } \mathrm{Prm}(\boldsymbol{x}, \boldsymbol{y}^{\star}_{\boldsymbol{x}}, s_h) = 0. \end{cases} \tag{5}$$

Here, $\lambda_D^{(step)}, \lambda_U^{(step)}$ and β_{step} mirror their outcome-level counterparts, controlling the strength of the reward or penalty at the granularity of individual steps. By leveraging these signals, the stepwise value function $v^{(step)}$ directs the model's distribution toward steps deemed correct and coherent, and away from those that are not. With these definitions, the stepwise loss is:

$$\mathscr{L}_{\text{step}}(,) = \mathbb{E}_{x,y,s_h \sim D^{(step)}}\big[\lambda_y^{(step)} - v^{(step)}(x, s_h)\big]. \tag{6}$$

where $\lambda_y^{(step)} = \lambda_D^{(step)}$ if $\mathrm{Prm}(\boldsymbol{x}, \boldsymbol{y}^{\star}_{\boldsymbol{x}}, s_h) = 1$ and $\lambda_y^{(step)} = \lambda_U^{(step)}$ if $\mathrm{Prm}(\boldsymbol{x}, \boldsymbol{y}^{\star}_{\boldsymbol{x}}, s_h) = 0$.

Combining the stepwise objective with the outcome-level KTO loss (Eq. 4) yields the final STEP-KTO objective:

$$\begin{aligned} \mathscr{L}_{\text{STEP-KTO}}(\pi_\theta, \pi_{\text{ref}}) = \mathscr{L}_{\text{KTO}}(\pi_\theta, \pi_{\text{ref}}) \\ + \mathscr{L}_{\text{step}}(\pi_\theta, \pi_{\text{ref}}). \end{aligned} \tag{7}$$

This composite loss encourages the model to produce not only correct final answers but also to refine each intermediate step. By jointly optimizing outcome-level and stepwise-level feedback, STEP-KTO ensures that the model's entire reasoning trajectory—from the earliest steps to the final solution—is both correct and coherent.

2.4 Iterative Training

We train our models using an iterative procedure inspired by previous alignment methods that refine a model's parameters over multiple rounds (Zelikman et al., 2022; Yuan et al., 2024; Pang et al., 2024; Prasad et al., 2024). For `Llama-3.3-70B-Instruct`, we use it directly as our seed model M_0. For Llama-3.1 models, we first perform supervised finetuning on the training data before using them as M_0. Starting from M_0, we refine it iteratively to obtain $M_1, M_2, \ldots, M_T$ using the following procedure:

1. **Candidate Generation:** For each problem $\boldsymbol{x} \in \mathscr{D}$, we sample 8 candidate solutions $\boldsymbol{y}^k \sim \pi_{M_t}(\cdot \mid \boldsymbol{x})$ using temperature $T = 0.7$ and nucleus sampling with $p = 0.95$ (Holtzman et al., 2020). This stochastic decoding strategy encourages diverse candidate solutions, aiding both positive and negative sample selection.
2. **Outcome Assessment:** We evaluate each candidate $\boldsymbol{y}^k$ against the ground-truth solution $\boldsymbol{y}^{\star}_{\boldsymbol{x}}$ using the outcome correctness function $\mathrm{Regex}(\boldsymbol{y}^k, \boldsymbol{y}^{\star}_{\boldsymbol{x}})$. If no sampled solutions are correct, we include the ground-truth solution $\boldsymbol{y}^{\star}_{\boldsymbol{x}}$ as a positive sample, as suggested by Pang et al. (2024). If all sampled solutions are correct, we discard this problem in the current iteration to prioritize learning from problems where the model can still improve.
3. **Stepwise Evaluation:** For the selected solutions, we apply the stepwise correctness function $\mathrm{Prm}(\boldsymbol{x}, \boldsymbol{y}^{\star}_{\boldsymbol{x}}, s_h)$ to assess the quality of each reasoning step. This yields a set of binary signals indicating whether each intermediate step aligns with desirable reasoning patterns.
4. **Dataset Construction:** We aggregate these annotated samples into $\mathscr{D}_{M_t} = \{(\boldsymbol{x}, \boldsymbol{y}, c^{out}, c_1^{step}, \ldots, c_{S-1}^{step}) \mid \boldsymbol{y} \in \mathscr{D}\}$, where $c^{out} = \mathrm{Regex}(\boldsymbol{y}, \boldsymbol{y}^{\star}_{\boldsymbol{x}})$ is the outcome-level correctness, and $c_h^{step} = \mathrm{Prm}(\boldsymbol{x}, \boldsymbol{y}^{\star}_{\boldsymbol{x}}, s_h)$ are the stepwise correctness indicators for the $S - 1$ intermediate steps of the solution $\boldsymbol{y}$. [1]
5. **Parameter Update:** Using $\mathscr{D}_{M_t}$, we update the model parameters according to the chosen alignment objective—either our STEP-KTO loss or a baseline method (e.g., IRPO).
6. **Iteration:** We repeat this process for T iterations, each time producing a new model M_{t+1} refined from M_t.

While KTO and STEP-KTO does not inherently require balanced positive and negative samples, we impose this constraint for fairness when comparing against pairwise preference-based baselines like DPO. Specifically, we randomly sample at most two pairs per problem per iteration, ensuring a consistent number of training examples across different alignment strategies. This controlled sampling regime facilitates direct comparisons between

[1] At each iteration t, the dataset $\mathscr{D}_{M_t}$ is constructed specifically from M_t. Thus, M_1 is trained on the dataset derived from seed model M_0 shared by all methods, M_2 on the dataset derived from M_1 specifically for method testing, and so forth.

methods and clarifies the impact of stepwise and outcome-level feedback on the model's refinement process.

3 Experiments

3.1 Task and Datasets

We evaluate our approach on established math reasoning benchmarks from high school competitions, testing the model's ability to solve challenging problems across various domains and difficulties. All problems require a final answer, typically a number, simplified expression (e.g., $\frac{\pi}{2}$, $1 \pm \sqrt{19}$), or short text (e.g., "east").

- **MATH-500:** A 500-problem subset of the MATH dataset (Hendrycks et al., 2021), selected as in Lightman et al. (2024). It covers diverse subjects (e.g., Algebra, Geometry, Precalculus) for a broad evaluation of mathematical reasoning.
- **AMC23:** A test set of 40 problems from the American Mathematics Competitions (AMC 12, 2023)[2]. These problems are known for their subtlety and depth, providing a stringent reasoning test.
- **AIME24:** A test set of 30 problems from the American Invitational Mathematics Examination (AIME, 2024)[3], typically requiring intricate multi-step reasoning and posing a higher-level challenge.

Following standard mathematical LLM evaluation practices (Hendrycks et al., 2021), we extract final answers from model outputs using regular expressions and verify their mathematical equivalence to ground-truth solutions with SYMPY[4], accommodating minor stylistic differences. We report Pass@1 (accuracy of a single greedy completion from π_θ) and Maj@8 (accuracy from the majority answer among 8 solutions sampled at $T = 0.7$ (Ackley et al., 1985; Ficler and Goldberg, 2017; Wang et al., 2023))[5]. These metrics provide a comprehensive assessment on challenging mathematical reasoning tasks, reflecting direct accuracy (Pass@1) and sampled robustness (Maj@8).

In addition to these evaluation benchmarks, all experiments are conducted using a large-scale prompt set, $\mathscr{D}_{\text{Numina}}$, referred to as `NuminaMath` (LI et al., 2024). `NuminaMath` comprises a broad range of math problems and their solutions, totaling 438k examples, spanning difficulty levels from elementary to high school competition standards. To ensure the integrity of final answers, we remove subsets of synthetic questions and Orca Math problems (Mitra et al., 2024), as their correctness are not verified by human.

3.2 Baseline Methods

We evaluate our proposed STEP-KTO against several strong baseline approaches for mathematical reasoning. All methods are trained using offline iterative optimization, with online preference learning left as future work:

- **RFT (Rejection Finetuning)** (Yuan et al., 2023): Performs supervised finetuning exclusively on solutions with correct final answers, relying on outcome-level filtering without explicit preference signals.
- **IRPO (Iterative Reasoning Preference Optimization)** (Pang et al., 2024): An iterative DPO (Rafailov et al., 2023) variant using outcome-level pairwise preferences, stabilized by an NLL loss, but lacks stepwise feedback.
- **KTO (Kahneman-Tversky Optimization)** (Ethayarajh et al., 2024): Employs an outcome-level, Kahneman-Tversky-inspired value function (see §2.2) for alignment, focusing on risk aversion but not incorporating stepwise signals.
- **SimPO and IPO** (Meng et al., 2024; Azar et al., 2024): DPO variants that utilize simplified outcome-level preference mechanisms for more stable optimization, without targeting stepwise correctness or advanced reasoning performance.
- **Step-DPO** (Lai et al., 2024): A DPO variant that optimizes stepwise preferences instead of outcome-level ones for granular supervision, but requires significant data processing and rejection sampling for intermediate steps.

3.3 Main Results

Table 1 presents our main results, comparing STEP-KTO with various baseline methods and commercial systems across the MATH-500, AMC23, and AIME24 benchmarks. We report both Pass@1 and Maj@8 accuracy, as described in §3. Overall,

[2]`https://github.com/QwenLM/Qwen2.5-Math/blob/main/evaluation/data/amc23/test.jsonl`

[3]`https://github.com/QwenLM/Qwen2.5-Math/blob/main/evaluation/data/aime24/test.jsonl`

[4]`https://github.com/sympy/sympy`

[5]Varying temperature ($T = 0.5 - 1.0$) had limited impact on Maj@8 in pilot experiments.

Method	MATH-500		AMC23		AIME24	
	Pass@1	Maj@8	Pass@1	Maj@8	Pass@1	Maj@8
Llama-3.1-8B-Instruct						
Seed model M_0	53.4	55.0	35.0	37.5	3.3	6.7
Rejection Finetuning M_3	53.8	56.0	30.0	32.5	10.0	6.7
IRPO M_3	55.4	59.2	35.0	40.0	6.7	6.7
KTO M_3	60.6	61.6	35.0	32.5	**16.7**	**16.7**
STEP-KTO (ours) M_3	**63.2**	**64.6**	**47.5**	**47.5**	**16.7**	**16.7**
Llama-3.1-70B-Instruct						
Seed model M_0	74.6	76.2	40.0	60.0	13.3	16.7
Rejection Finetuning M_1	74.8	73.6	55.0	60.0	13.3	13.3
IRPO M_1	74.4	74.8	55.0	57.5	10.0	13.3
KTO M_1	75.6	77.2	55.0	65.0	13.3	13.3
STEP-KTO (ours) M_1	**76.2**	**78.4**	**60.0**	**67.5**	**16.7**	**20.0**
Llama-3.3-70B-Instruct M_0	75.8	77.6	57.5	60.0	26.7	30.0
Rejection Finetuning M_1	77.4	78.4	60.0	65.0	20.0	23.3
IRPO M_1	78.6	80.8	55.0	57.5	23.3	26.7
KTO M_1	78.6	79.8	60.0	65.0	20.0	23.3
STEP-KTO (ours) M_1	**79.6**	**81.6**	**70.0**	**75.0**	**30.0**	**33.3**
Llama-3.1-8B-Instruct	51.4	55.2	15.0	27.5	3.3	3.3
Llama-3.1-70B-Instruct	64.8	70.4	37.5	47.5	10.0	30.0
Llama-3.1-405B-Instruct	68.8	74.4	47.5	52.5	30.0	26.6
O1	94.8	-	-	-	78.0	-
O1-Mini	90.0	-	90.0	90.0	33.3	46.7
Gemini 1.5 Pro	79.4	83.0	75.0	82.5	26.7	26.7
GPT-4o	73.0	76.4	57.5	70.0	10.0	16.7
Claude 3.5 Sonnet	70.0	74.4	62.5	67.5	23.3	26.7
Grok-Beta	67.0	72.2	50.0	52.5	10.0	13.3

Table 1: **Math problem solving performance** comparing Llama models of different sizes and proprietary models. Results show accuracy on MATH-500, AMC23, and AIME24 test sets using both greedy decoding (Pass@1) and majority voting over 8 samples (Maj@8). Models highlighted in blue are 8B parameter models, green are 70B parameter models, and gray are commercial models.

STEP-KTO consistently outperforms the baselines that rely solely on outcome-level correctness, such as KTO, IRPO, SimPO, and IPO, as well as simpler methods like RFT.

For instance, on MATH-500 with the 8B Llama-3.1-Instruct model, STEP-KTO achieves a Pass@1 of 63.2%, improving from the baseline KTO model's 60.6% and substantially surpassing IRPO and RFT. On AMC23, STEP-KTO attains a Pass@1 of 47.5%, outperforming baselines by a notable margin. On AIME24, where problems require especially intricate multi-step reasoning, STEP-KTO sustains its advantage, demonstrating that the stepwise supervision is particularly valuable for more challenging tasks. Scaling to the 70B further improves results. `Llama-3.1-70B-Instruct` with STEP-KTO reaches a Pass@1 of 76.2% on MATH-500 and continues to excel on AMC23 (60.0%) and AIME24 (16.7%). `Llama-3.3-70B-Instruct` with STEP-KTO model pushes performance higher still, with STEP-KTO achieving 79.6% on MATH-500, 70.5% on AMC23, and 29.6% on AIME24. Although larger models also benefit from outcome-only alignment techniques, STEP-KTO still delivers consistent gains, indicating that even powerful models trained on extensive data can be further improved by targeting intermediate reasoning quality. Compared to strong proprietary models, STEP-KTO-enhanced Llama models remain competitive and close the performance gap. For example, while GPT-4o achieves a respectable 73.0% Pass@1 on MATH-500, O1 series pushes this accuracy to 90.0% and higher but requires a substantially larger inference budget. In contrast, our STEP-KTO-enhanced `Llama-3.1-70B-Instruct` model attains 76.2% Pass@1 on MATH-500 using only a 5k-token budget.

3.4 Iterative Training

Table 2 illustrates how model performance evolves over multiple iterative training rounds (M_1, M_2, M_3) when starting from the same seed model M_0 (`Llama-3.1-8B-Instruct`). We compare STEP-KTO against other iterative methods such as IRPO, KTO, and Rejection Finetuning.

Overall, STEP-KTO not only achieves higher final performance but also improves more consistently across iterations. For instance, on MATH-

Method	MATH-500		AMC23		AIME24	
	Pass@1	Maj@8	Pass@1	Maj@8	Pass@1	Maj@8
Llama-3.1-8B-Instruct						
Seed model M_0	53.4	55.0	35.0	37.5	3.3	6.7
IPO M_1	52.6	55.8	22.5	30.0	3.3	3.3
SimPO M_1	55.8	57.2	25.0	25.0	6.7	10.0
Step-DPO M_1	56.8	58.4	27.5	30.0	6.7	10.0
Rejection Finetuning M_1	55.0	57.0	30.0	35.0	10.0	10.0
Rejection Finetuning M_2	54.0	56.2	22.5	20.0	3.3	6.7
Rejection Finetuning M_3	53.8	56.0	30.0	32.5	10.0	6.7
IRPO M_1	58.2	59.6	35.0	35.0	10.0	10.0
IRPO M_2	57.2	62.4	32.5	40.0	6.7	10.0
IRPO M_3	55.4	59.2	35.0	40.0	6.7	6.7
KTO M_1	56.2	55.6	32.5	32.5	6.7	10.0
KTO M_2	59.4	62.8	35.5	35.0	**16.7**	**16.7**
KTO M_3	60.6	61.6	35.0	32.5	**16.7**	**16.7**
STEP-KTO (ours) M_1	59.4	60.6	22.5	32.5	13.3	10.0
STEP-KTO (ours) M_2	**63.6**	63.0	40.0	40.0	13.3	**16.7**
STEP-KTO (ours) M_3	63.2	**64.6**	**47.5**	**47.5**	**16.7**	**16.7**

Table 2: **Iterative training performance** comparing different methods on `Llama-3.1-8B-Instruct` model. Results show accuracy across multiple iterations (M_1, M_2, M_3) of training on MATH-500, AMC23, and AIME24 test sets using both greedy decoding (Pass@1) and majority voting over 8 samples (Maj@8).

500, STEP-KTO progresses from 59.4% Pass@1 at M_1 to 63.2% at M_3, surpassing the gains observed by IRPO and KTO at the same checkpoints. Similarly, on AMC23 and AIME24, STEP-KTO shows steady iterative improvements, reflecting the cumulative value of integrating both process- and outcome-level feedback. In contrast, Rejection Finetuning (RFT) and IRPO exhibit less stable gains across iterations, with performance sometimes plateauing or even regressing at later rounds. KTO does improve over iterations, but not as robustly as STEP-KTO, highlighting that stepwise feedback adds tangible benefits beyond what outcome-level optimization alone can achieve.

These results underscore the importance of iterative refinement. While simply applying preference-based or rejection-based finetuning may yield some initial improvements, STEP-KTO's combined stepwise and outcome-level guidance drives steady, sustained enhancements in mathematical reasoning quality, iteration after iteration.

3.5 Comparison with Step-DPO

Step-DPO (Lai et al., 2024) also targets intermediate steps but relies on computationally intensive rejection sampling for error correction. STEP-KTO contrasts by efficiently combining stepwise and outcome signals for global solution coherence. Empirically, Step-DPO achieved 56.8% Pass@1 on MATH-500 (M_1), whereas STEP-KTO reached 59.4%. Our Step-DPO implementation used `Llama-3.3-70B-Instruct` for error identification and rejection sampling (filtering unsolved after 8 attempts), underscoring STEP-KTO's advantage in sustained improvement via integrated optimization.

3.6 Preference Optimization Variants

Table 2 compares STEP-KTO against baselines over iterative training from the 8B M_0. On MATH-500 (M_1), STEP-KTO (59.4% Pass@1) outperformed IPO (52.6%), SimPO (55.8%), IRPO (58.2%), and KTO (56.2%). While its initial M_1 gains on AMC23 and AIME24 were comparable or more modest, STEP-KTO demonstrated stronger subsequent improvements. By M_3, STEP-KTO achieved 47.5% Pass@1 on AMC23, surpassing all baselines, and tied for the highest Pass@1 (16.7%) on AIME24, highlighting the value of integrating stepwise and outcome-level signals.

3.7 Evaluating Reasoning Quality

8B Model	Stepwise Errors in Correct Solutions	
	KTO	STEP-KTO
M_0	27.3%	27.3%
M_1	24.6%	**22.9%**
M_2	22.6%	**20.8%**
M_3	21.1%	**19.9%**

Table 3: **Reasoning Quality Analysis** comparing the ratio of solutions that arrive at correct final answers despite containing erroneous intermediate steps on the MATH-500.

To assess the internal consistency of solutions with correct final answers, we evaluate the proportion of solutions that, despite having correct final answer $\text{Regex}(\boldsymbol{y}, \boldsymbol{y}_{\boldsymbol{x}}^{\star}) = 1$, contain at least one erroneous intermediate step. We use the ProcessBench (Zheng et al., 2024) as our evaluation framework, which is prompted to identify the earliest error in the generated solution $\boldsymbol{y}$, as detailed in its benchmark construction. Additionally, we utilize the critique capabilities of the `QwQ-32B-Preview` model (Qwen, 2024) to identify the first error in the reasoning. We prompt QwQ using the prompt detailed in Appendix D. We then measure the percentage of correctly answered problems where QwQ identifies at least one erroneous intermediate step.

Table 3 shows the percentage of correctly answered solutions containing errors in reasoning steps, starting from the initial 8B seed model M_0, which produces reasoning steps containing errors in 27.3% of its correctly answered solutions on the MATH-500 test set. Both STEP-KTO and KTO reduce the prevalence of such errors across iterations, with STEP-KTO showing a greater and more consistent reduction from 27.3% at M_0 to 19.9% at M_3, compared to KTO's more modest improvement to 21.1%.

4 Related Work

Outcome-Oriented Methods. Many efforts refine LLMs using only final outputs. Instruction tuning (Ouyang et al., 2022; Touvron et al., 2023) and outcome-level feedback via Reinforcement Learning from Human Feedback (RLHF) (e.g., InstructGPT (Ouyang et al., 2022)) or direct preference optimization (DPO (Rafailov et al., 2023), KTO (Ethayarajh et al., 2024), SimPO (Meng et al., 2024), IPO (Azar et al., 2024)) improve final answer accuracy using human or synthetic labels. AI-generated feedback (RLAIF (Lee et al., 2023)) or predefined rules (Constitutional AI (Bai et al., 2022b)) aim to reduce human annotation. While refinements like CGPO (Xu et al., 2024) offer richer signals, they primarily evaluate entire outputs. A key limitation is that correct final answers do not guarantee sound intermediate reasoning (Wu et al., 2024), potentially yielding untrustworthy solution paths (Turpin et al., 2023; Lanham et al., 2023).

Process-Level Feedback and Verification. Process Reward Models (PRMs) (Lightman et al., 2024; Uesato et al., 2022; Xiong et al., 2024; Luo et al., 2024) focus on stepwise correctness, assigning local feedback to guide models toward logically consistent solutions. This is prevalent in math reasoning, supported by datasets like PRM800K (Lightman et al., 2024), CriticBench (Lin et al., 2024), and ProcessBench (Zheng et al., 2024) that facilitate step-level evaluations. PRM-based techniques influence decoding (Li et al., 2023; Chuang et al., 2024; Wang et al., 2024), re-ranking (Cobbe et al., 2021), filtering (Dubey et al., 2024; Shao et al., 2024), and iterative loops like STaR (Zelikman et al., 2022) and ReST (Gülçehre et al., 2023; Singh et al., 2024). Synthetic feedback helps scale annotations (Wang et al., 2024; Lightman et al., 2024; Chiang and Lee, 2024; Huang and Chen, 2024). Yet, focusing solely on process may not yield correct final answers, as local rewards can be exploited or chains may fail to converge (Gao et al., 2024).

Integrating Outcome- and Process-Level Signals. Recognizing the limitations of supervising only outcomes or processes, recent studies combine both signals. FactTune (Tian et al., 2024) and FactAlgin (Huang and Chen, 2024) integrate PRMs with factuality evaluators for alignment, enhancing factual accuracy. In math reasoning, Uesato et al. (2022) and Shao et al. (2024) also leveraged combined step and outcome feedback. While the principle of multi-granularity supervision is broadly applicable, especially to math reasoning, these combined approaches can still face challenges in scaling, balancing feedback types, and avoiding premature performance plateaus (Bai et al., 2022a; Xu et al., 2023; Singh et al., 2024).

5 Conclusion

This work proposes STEP-KTO, a training framework that leverages both outcome-level and process-level binary feedback to guide large language models toward more coherent, interpretable, and dependable reasoning. By integrating stepwise correctness signals into the alignment process, STEP-KTO improves the quality of intermediate reasoning steps while maintaining or even enhancing final answer accuracy. Our experiments on challenging mathematical reasoning benchmarks demonstrate consistent gains in performance, particularly under iterative training and for complex reasoning tasks. These findings underscore the value of aligning not only final outcomes but also the entire reasoning trajectory. We envision STEP-KTO as a stepping stone toward more reliable reasoning in LLMs.

Limitations

Despite STEP-KTO's promise, several limitations persist. First, outcome-level feedback can be noisy; for instance, automated math answer verification may misjudge valid but unconventional representations, limiting training signal precision. Second, STEP-KTO currently presumes access to ground-truth solutions for outcome and (implicitly) for guiding stepwise correctness. Generating meaningful stepwise feedback is challenging without high-quality reference reasoning or in domains with inherently ambiguous intermediate steps. Learning from weaker signals or pure preferences remains an open area. Finally, our experiments assume some baseline correctness. If initial outcomes are consistently incorrect and intermediate steps are invalid, STEP-KTO's ability to bootstrap performance is uncertain. Such scenarios might require complementary techniques like curriculum learning or stronger initialization before stepwise feedback becomes effective.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This paper's writing received minor language-polishing suggestions from ChatGPT. In addition, parts of our experimental code were drafted or refactored with assistance from Meta AI; all final implementations were manually reviewed and verified by the authors.

References

David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. 1985. A learning algorithm for boltzmann machines. *Cogn. Sci.*, 9(1):147–169.

Leonard Adolphs, Kurt Shuster, Jack Urbanek, Arthur Szlam, and Jason Weston. 2022. Reason first, then respond: Modular generation for knowledge-infused dialogue. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 7112–7132. Association for Computational Linguistics.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Rémi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*, volume 238 of *Proceedings of Machine Learning Research*, pages 4447–4455. PMLR.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, and 32 others. 2022b. Constitutional AI: harmlessness from AI feedback. *CoRR*, abs/2212.08073.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Cheng-Han Chiang and Hung-yi Lee. 2024. Merging facts, crafting fallacies: Evaluating the contradictory nature of aggregated factual claims in long-form generations. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2734–2751, Bangkok, Thailand. Association for Computational Linguistics.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. 2024. DoLa: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*.

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. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Model alignment as prospect theoretic optimization. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. 2024. On designing effective RL reward at training time for LLM reasoning. *CoRR*, abs/2410.15115.

Çaglar Gülçehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. 2023. Reinforced self-training (rest) for language modeling. *CoRR*, abs/2308.08998.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron C. Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *CoRR*, abs/2402.06457.

Chao-Wei Huang and Yun-Nung Chen. 2024. FactAlign: Long-form factuality alignment of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16363–16375, Miami, Florida, USA. Association for Computational Linguistics.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *CoRR*, abs/2406.18629.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamile Lukosiute, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *CoRR*, abs/2307.13702.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. RLAIF: scaling reinforcement learning from human feedback with AI feedback. *CoRR*, abs/2309.00267.

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

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, and 7 others. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

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

Zicheng Lin, Zhibin Gou, Tian Liang, Ruilin Luo andG Haowei Liu, and Yujiu Yang. 2024. Criticbench: Benchmarking llms for critique-correct reasoning. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 1552–1587. Association for Computational Linguistics.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *CoRR*, abs/2406.06592.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, IJCNLP 2023 -Volume 1: Long Papers, Nusa Dua, Bali, November 1 - 4, 2023*, pages 305–329. Association for Computational Linguistics.

MAA. 2023. American mathematics competitions (amc).

MAA. 2024. American invitational mathematics examination (aime).

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *CoRR*, abs/2405.14734.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. Orca-math: Unlocking the potential of slms in grade school math. *CoRR*, abs/2402.14830.

Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *CoRR*, abs/2404.19733.

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason Weston, and Jane Yu. 2024. Self-consistency preference optimization. *Preprint*, arXiv:2411.04109.

Qwen. 2024. Qwq-32b preview. `https://qwenlm.github.io/blog/qwq-32b-preview/`. Accessed: 2024-06-17.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, and 6 others. 2023. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for LLM reasoning. *CoRR*, abs/2410.08146.

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

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, and 1 others. 2024. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*. Expert Certification.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. 2024. Fine-tuning language models for factuality. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Amos Tversky and Daniel Kahneman. 2016. *Advances in Prospect Theory: Cumulative Representation of Uncertainty*, pages 493–519. Springer International Publishing, Cham.

Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *CoRR*, abs/2211.14275.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Tianhao Wu, Janice Lan, Weizhe Yuan, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. 2024. Thinking llms: General instruction following with thought generation. *CoRR*, abs/2410.10630.

Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, Chi Jin, Tong Zhang, and Tianqi Liu. 2024. Building math agents with multi-turn iterative preference learning. *CoRR*, abs/2409.02392.

Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. 2023. Some things are more CRINGE than others: Preference optimization with the pairwise cringe loss. *CoRR*, abs/2312.16682.

Tengyu Xu, Eryk Helenowski, Karthik Abinav Sankararaman, Di Jin, Kaiyan Peng, Eric Han, Shaoliang Nie, Chen Zhu, Hejia Zhang, Wenxuan Zhou, Zhouhao Zeng, Yun He, Karishma Mandyam, Arya Talabzadeh, Madian Khabsa, Gabriel Cohen, Yuandong Tian, Hao Ma, Sinong Wang, and Han Fang. 2024. The perfect blend: Redefining RLHF with mixture of judges. *CoRR*, abs/2409.20370.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *CoRR*, abs/2308.01825.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024. Generative verifiers: Reward modeling as next-token prediction. *CoRR*, abs/2408.15240.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

A Implementation Details

We use AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay $= 0.1$) with a linear warmup for the first 100 steps and a cosine decay schedule that reduces the learning rate to $0.1\times$ its initial value. The starting learning rate is 1.0×10^{-6}, and we apply global norm gradient clipping of 1.0. The effective global batch size is set to approximately one million tokens, and we train for about 2000 steps, periodically evaluating our models during training on the hold-out test set from MATH (Hendrycks et al., 2021)[6] to select the best checkpoint for each method. For IRPO, we use an NLL weight of 0.2. We set $\beta = 0.1$ for all methods. All training jobs are run on 64 H100 GPUs.

B Decontamination

To prevent data leakage between training and test sets, we perform standard decontamination by normalizing text (converting to lowercase and removing non-alphanumeric characters) and checking for exact string matches between test questions and training prompts (Dubey et al., 2024). We remove any matching examples from the training data. This process is applied to all datasets in our evaluation. Even if mild contamination were present, we expect any resulting performance inflation to be small and consistent across all conditions, leaving the relative comparisons between our methods largely unaffected.

C Details of API Usage for Proprietary Models

In our experiments, we evaluated several proprietary models via their respective APIs: O1 (metrics are self-reported), O1-Mini (`o1-mini-2024-09-12`, MATH-500 is self-reported [7]), Gemini 1.5 Pro (`gemini-1.5-pro-002`), GPT-4o (`gpt-4o-2024-08-06`), Claude 3.5 Sonnet (`claude-3-5-sonnet-20241022`), and Grok-Beta. These experiments took place on November 15 and 16, 2024. For each model, questions were used directly as user prompts. For greedy decoding, we set the temperature to 0.0 to ensure deterministic outputs, except for o1 models where we used temperature 1.0 due to API restrictions (only temperature 1.0 is allowed) and took the first sample. For sampling, we set the temperature to 0.7 and performed 8 generations per question to enable majority voting.

Response Generation for Proprietary Models

User:

```
Please answer the following question step-by-step. Once you have the final answer,
place it on a new line as: The final answer is \$\boxed{answer}\$.
Question: {{ question }}
```

[6] MATH-500 questions are excluded.

[7] Numbers from https://github.com/openai/simple-evals

D Prompts

Prompt templates [8] for generating solutions are given below in Appendix D.

Response Generation Template from (Dubey et al., 2024)

User:

```
Solve the following math problem efficiently and clearly:

For simple problems (2 steps or fewer):
Provide a concise solution with minimal explanation.

For complex problems (3 steps or more):
Use this step-by-step format:

## Step 1: [Concise description]
[Brief explanation and calculations]

## Step 2: [Concise description]
[Brief explanation and calculations]

...

Regardless of the approach, always conclude with:

Therefore, the final answer is: \$\boxed{answer}\$. I hope it is correct.

Where [answer] is just the final number or expression that solves the problem.

Problem: {{ question }}
```

Prompt for `Llama-3.1-70B-Instruct` to provide stepwise feedback on candidate solutions $\boldsymbol{y}$. The model analyzes each step s_h of a potential solution against the correct answer $\boldsymbol{y}^{\star}$, evaluating the reasoning and accuracy of each step. The feedback is structured in JSON format with fields for step number, reflection on the reasoning, and a binary decision on whether the step contributes positively to reaching the solution.

[8] The prompt template was from `https://huggingface.co/datasets/meta-llama/Llama-3.1-70B-Instruct-evals`

Generation Prompt for Stepwise Feedback

User:

````
Please analyze the following problem and its potential solution step-by-step.
Provide feedback on each step and determine if it contributes positively to reaching the
correct solution.

<problem>
{{ problem }}
</problem>

<correct solution>
{{ answer }}
</correct solution>

<potential answer>
{% for step in steps %}
<step {{ loop.index }}>
{{ step }}
</step {{ loop.index }}>
{% endfor %}
</potential answer>

Analyze your **potential solution** as if you had originally generated it.
Carefully review each step, considering its reasoning, accuracy, and execution.
Assess whether the step contributes positively to reaching the correct solution.
Where necessary, refine the step to address any flaws or gaps. Use the correct answer
as a ground truth reference to guide your analysis.

Provide your output in JSON format, where each element represents a step of the solution.
Use the fields below:

- **step**: The step order number in the reasoning process.
- **reflection**: A concise evaluation of the accuracy of the reasoning in this step
(point out why it helps or hinders the solution).
- **decision**: The evaluation of the step, either "positive" or "negative".

The expected output format follows:

```json
[
    {
        "step": 1,
        "reflection": "[evaluation of step 1 reasoning]",
        "decision": "positive"
    },
    {
        "step": 2,
        "reflection": "[evaluation of step 2 reasoning]",
        "decision": "negative"
    },
    ...\\
]
```

# Notes

- Assign **"negative"** only to steps that are clearly incorrect and prevent the
solution from progressing.
- Use the correct answer as one of the ground truth references.
- Ensure the reflections remain concise, with the total explanation under 200 words
across all steps combined.
````

Process Critique Prompt from ProcessBench (Zheng et al., 2024)

User:

```
The following is a math problem and a solution (split into paragraphs, enclosed with
tags and indexed from 0):

<problem>
{{ problem }}
</problem>

<solution>
{% for step in steps %}
<step {{ loop.index }}>
{{ step }}
</step {{ loop.index }}>
{% endfor %}
</solution>

Your task is to review and critique the solution paragraph by paragraph. Once you
identify an error in a paragraph, return the index of the paragraph where the
earliest error occurs. Otherwise, return the index of -1 (which typically denotes
"not found").

Please put your final answer (i.e., the index) in \boxed{{}}.
```

E Qualitative Examples

We analyze several examples from `Llama-3.3-70B-Instruct` STEP-KTO M_1 on MATH-500 to understand how Step-KTO helps improve mathematical reasoning. The examples demonstrate three key scenarios where Step-KTO provides effective feedback: (1) when all steps and the final answer are correct, (2) when intermediate steps contain errors but lead to the correct final answer, and (3) when both intermediate steps and the final answer are incorrect.

Step-KTO Generation Example 1: Final Answer is Right and All Steps are Right

```
problem: For a point $P,$ let $d_1,$ $d_2$ and $d_3$ represents the distances from $P$
         to the planes $x - z = 0,$ $x - 2y + z = 0,$ and $x + y + z = 0.$  Let $S$ be
         the set of points $P$ such that\n\\[d_1^2 + d_2^2 + d_3^2 = 36.\\]Find the region
         of the volume enclosed by $S.$
answer: "288 \\pi",
subject: "Precalculus",
level: 4,
unique_id: "test/precalculus/1056.json"

Model prediction:
## Step 1: Understand the problem and the equation given
The problem provides an equation $d_1^2 + d_2^2 + d_3^2 = 36$ where $d_1,$ $d_2,$ and
$d_3$ represent the distances from a point $P$ to the planes $x - z = 0,$ $x - 2y + z
= 0,$ and $x + y + z = 0$ respectively. We need to find the region of the volume enclosed by
the set $S$ of points $P$ that satisfy this equation.

## Step 2: Calculate the distance of a point to each plane
The distance $d$ from a point $(x_0, y_0, z_0)$ to a plane $Ax + By + Cz + D = 0$ is given by
the formula $d = \frac{|Ax_0 + By_0 + Cz_0 + D|}{\sqrt{A^2 + B^2 + C^2}}$. Applying this formula
to each plane, we get:
- $d_1 = \frac{|x - z|}{\sqrt{1^2 + (-1)^2}} = \frac{|x - z|}{\sqrt{2}}$
- $d_2 = \frac{|x - 2y + z|}{\sqrt{1^2 + (-2)^2 + 1^2}} = \frac{|x - 2y + z|}{\sqrt{6}}$
- $d_3 = \frac{|x + y + z|}{\sqrt{1^2 + 1^2 + 1^2}} = \frac{|x + y + z|}{\sqrt{3}}$

## Step 3: Substitute the distance formulas into the given equation
Substituting $d_1,$ $d_2,$ and $d_3$ into the equation $d_1^2 + d_2^2 + d_3^2 = 36$, we get:
$\left(\frac{x - z}{\sqrt{2}}\right)^2 + \left(\frac{x - 2y + z}{\sqrt{6}}\right)^2 +
\left(\frac{x + y + z}{\sqrt{3}}\right)^2 = 36$

## Step 4: Simplify the equation
Expanding and simplifying, we have:
$\frac{(x - z)^2}{2} + \frac{(x - 2y + z)^2}{6} + \frac{(x + y + z)^2}{3} = 36$

## Step 5: Further simplification and rearrangement
Multiplying through by 6 to clear the denominators gives:
$3(x - z)^2 + (x - 2y + z)^2 + 2(x + y + z)^2 = 216$

## Step 6: Expand the terms
Expanding each term yields:
$3(x^2 - 2xz + z^2) + (x^2 - 4xy + 4y^2 + z^2 + 2xz - 4yz) + 2(x^2 + 2xy + 2xz + y^2 + 2yz + z^2) = 216$

## Step 7: Combine like terms
Combining like terms, we get:
$3x^2 - 6xz + 3z^2 + x^2 - 4xy + 4y^2 + z^2 + 2xz - 4yz + 2x^2 + 4xy + 4xz + 2y^2 + 4yz + 2z^2 = 216$

## Step 8: Simplify further
This simplifies to:
$6x^2 + 6y^2 + 6z^2 + 0xy + 0xz + 0yz = 216$

## Step 9: Divide by 6
Dividing the entire equation by 6 gives:
$x^2 + y^2 + z^2 = 36$

## Step 10: Interpret the result
The equation $x^2 + y^2 + z^2 = 36$ represents a sphere with radius 6 centered at the origin (0,0,0).

The final answer is: $\boxed{288\pi}$
```

Step-KTO Generation Example 2: Final Answer is Right but Step 2 is Wrong

```
problem: For how many real values of $\sqrt{120-\sqrt{x}}$ is an integer?
answer: "11"
subject: "Algebra"
level: 5
unique_id: "test/algebra/1282.json"

Model prediction:
## Step 1: Understand the problem and identify the condition for $\sqrt{120-\sqrt{x}}$ to be an
integer. We need $\sqrt{120-\sqrt{x}}$ to be an integer, which means $120-\sqrt{x}$ must be a perfect
square.

## Step 2: List all perfect squares less than or equal to 120.
The perfect squares less than or equal to 120 are: 0, 1, 4, 9, 16, 25, 36, 49, 64,
81, 100, and 121. However, since we are considering $120-\sqrt{x}$, the relevant
perfect squares are from 0 to 120, which are 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100,
and 121 is not included because it exceeds 120.

## Step 3: Calculate the possible values of $\sqrt{x}$.
For each perfect square $y$ from the list, we calculate $\sqrt{x} = 120 - y$. This
gives us the possible values of $\sqrt{x}$ as: 120, 119, 116, 111, 104, 95, 84, 71, 56, 39, 20.

## Step 4: Consider the validity of each possible value of $\sqrt{x}$.
Since $\sqrt{x}$ cannot be negative and must be real, all calculated values in step
3 are valid because they are all positive or zero. However, we should also consider
that $\sqrt{x}$ itself must be real, meaning $x \geq 0$. Given that $\sqrt{x} = 120
- y$, the smallest value of $\sqrt{x}$ is when $y = 100$, giving $\sqrt{x} = 20$.
For $y = 121$, $\sqrt{x}$ would be negative, which is not valid for real $x$. Thus,
we only consider perfect squares up to 100.

## Step 5: Count the number of valid values of $\sqrt{x}$ that make
$\sqrt{120-\sqrt{x}}$ an integer.
From step 3 and considering the correction in step 4, we see that for each perfect
square up to 100 (inclusive), we get a valid $\sqrt{x}$. Thus, we count the number
of perfect squares from 0 to 100, which are 11 (0, 1, 4, 9, 16, 25, 36, 49, 64, 81,
100).

The final answer is: $\boxed{11}$

QwQ Critique:
\boxed{2}

The error in the solution occurs in the 2nd paragraph, where the reasoning incorrectly
states "The perfect squares less than or equal to 120 are: 0, 1, 4, 9, 16, 25, 36, 49,
64, 81, 100, and 121." It's incorrect because 121 is greater than 120, not less than or
equal to, and shouldn't be included in the list.
```

Step-KTO Generation Example 3: Final Answer is Wrong

```
Problem: Find the greatest integer less than $(\sqrt{7} + \sqrt{5})^6.$
         (Do not use a calculator!)
answer: "13535"
subject: "Intermediate Algebra"
level: 3
unique_id: "test/intermediate_algebra/607.json"

Model prediction:
## Step 1: Understand the problem and identify the key elements
We need to find the greatest integer less than $(\sqrt{7} + \sqrt{5})^6$ without using a
calculator. This involves expanding the given expression and simplifying it to a form where
we can estimate its value.

## Step 2: Apply the Binomial Theorem to expand the expression
The Binomial Theorem states that for any non-negative integer $n$, \((a + b)^n =
\sum_{k=0}^{n} \binom{n}{k}a^{n-k}b^k\). Applying this to our problem with $a =
\sqrt{7}$, $b = \sqrt{5}$, and $n = 6$, we get \((\sqrt{7} + \sqrt{5})^6 =
\binom{6}{0}(\sqrt{7})^6 + \binom{6}{1}(\sqrt{7})^5(\sqrt{5}) +
\binom{6}{2}(\sqrt{7})^4(\sqrt{5})^2 + \binom{6}{3}(\sqrt{7})^3(\sqrt{5})^3 +
\binom{6}{4}(\sqrt{7})^2(\sqrt{5})^4 + \binom{6}{5}(\sqrt{7})(\sqrt{5})^5 +
\binom{6}{6}(\sqrt{5})^6\).

## Step 3: Calculate each term of the expansion
Calculating each term:
- \(\binom{6}{0}(\sqrt{7})^6 = 1 \cdot 7^3 = 343\)
- \(\binom{6}{1}(\sqrt{7})^5(\sqrt{5}) = 6 \cdot 7^2 \cdot 7 \cdot \sqrt{5} = 6
\cdot 49 \cdot 7 \cdot \sqrt{5} = 2058\sqrt{5}\)
- \(\binom{6}{2}(\sqrt{7})^4(\sqrt{5})^2 = 15 \cdot 7^2 \cdot 5 = 15 \cdot 49 \cdot 5
= 3675\)
- \(\binom{6}{3}(\sqrt{7})^3(\sqrt{5})^3 = 20 \cdot 7 \cdot 7 \cdot 5 \cdot \sqrt{5}
= 20 \cdot 49 \cdot 5 \cdot \sqrt{5} = 4900\sqrt{5}\)
- \(\binom{6}{4}(\sqrt{7})^2(\sqrt{5})^4 = 15 \cdot 7 \cdot 5^2 = 15 \cdot 7 \cdot 25
= 2625\)
- \(\binom{6}{5}(\sqrt{7})(\sqrt{5})^5 = 6 \cdot \sqrt{7} \cdot 5^2 \cdot \sqrt{5} =
6 \cdot \sqrt{7} \cdot 25 \cdot \sqrt{5} = 150\sqrt{35}\)
- \(\binom{6}{6}(\sqrt{5})^6 = 1 \cdot 5^3 = 125\)

## Step 4: Combine like terms and simplify
Combining like terms, we notice that terms with \(\sqrt{5}\) and \(\sqrt{35}\) will
contribute to non-integer values, but we can simplify the expression by recognizing
that the integer part of the expansion will come from the terms without square roots
(since they are multiplied by integers) and the terms with square roots will contribute
to the fractional part. The terms without square roots are \(343 + 3675 + 2625 + 125 =
6768\).

## Step 5: Estimate the value of the expression with square roots
To estimate the value of the terms with square roots, we note that \(2058\sqrt{5}
\approx 2058 \cdot 2.236 = 4599.33\), \(4900\sqrt{5} \approx 4900 \cdot 2.236 =
10943.4\), and \(150\sqrt{35} \approx 150 \cdot 5.916 = 886.4\). Summing these
approximations gives us an estimate of the non-integer contributing terms: \(4599.33 +
10943.4 + 886.4 \approx 15429.13\).

## Step 6: Combine the integer and non-integer parts
Adding the integer part (from step 4) to the estimated value of the non-integer part
(from step 5), we get \(6768 + 15429.13 \approx 22197.13\).

## Step 7: Determine the greatest integer less than the result
The greatest integer less than \(22197.13\) is \(22197\).

The final answer is: $\boxed{22197}$
```

BloomWise: Enhancing Problem-Solving capabilities of Large Language Models using Bloom's-Taxonomy-Inspired Prompts

Maria-Eleni Zoumpoulidi[1] **Georgios Paraskevopoulos**[1] **Alexandros Potamianos**[2]
[1]Institute for Language and Speech Processing, Athena Research Center, Greece
[2]Speech and Language Processing Group, National Technical University of Athens, Greece
{m.zoumpoulidi, g.paraskevopoulos}@athenarc.gr, potam@central.ntua.gr

Abstract

Despite the remarkable capabilities of large language models (LLMs) across a range of tasks, mathematical reasoning remains a challenging frontier. Motivated by the observation that humans learn more effectively when prompted not what to think but how to think, we introduce BloomWise, a cognitively-inspired prompting technique designed to enhance LLMs' performance on mathematical problem solving while making their solutions more explainable. BloomWise encourages LLMs to generate solutions - in the form of explanations - by progressing through a sequence of cognitive operations—from basic (e.g., remembering) to more advanced reasoning skills (e.g., evaluating)—mirroring how humans build understanding. The process iterates through these levels, halting early if a convergence criterion is met: specifically, if two or more consecutive levels yield the same answer, the solution from the earliest such level is output; otherwise, the process continues until all levels are completed. Through extensive experiments across five popular math reasoning datasets, we demonstrate the effectiveness of BloomWise. We also present comprehensive ablation studies to analyze the strengths of each component within our system.

1 Introduction

Mathematical reasoning has long been regarded as a pinnacle of human intellect—demanding abstraction, logical structure, and creativity. As LLMs achieve remarkable fluency in natural language, empowering them with robust mathematical reasoning skills is crucial for scientific and technological progress. Yet, mastering the complexity and nuance of mathematical problem solving remains a formidable challenge for LLMs, motivating new approaches.

Numerous research efforts have leveraged in-context learning (ICL) (Brown et al., 2020) to improve the problem-solving capabilities of LLMs. Some of the most widely used techniques involve encouraging the LLM through prompts to develop a textual rationale with Chain-of-Thought prompting (Wei et al., 2022) (CoT) or Python functions with Program-Aided Language Model (Gao et al., 2022) and Program-of-Thought prompting (Chen et al., 2022) (PAL or PoT). Each of these methods comes with its own strengths and limitations: CoT enables flexible, sequential narrative-style reasoning but often struggles with precise numerical computation (Wei et al., 2022), (Lewkowycz et al., 2022), while code-based approaches like PoT and PAL offer accurate calculations via Python interpreters but lack the ability to handle unknown variables.

For this reason, research efforts have pivoted towards integrating multiple methodologies, aiming to identify the most suitable approach for each specific problem, while harnessing the collective strengths of various techniques. One such method, X of Thoughts (Liu et al., 2023), selects from CoT, PoT, or EoT (Equations of Thought) depending on the nature of the problem, applies the selected approach, and verifies the result (iteratively, until a correct solution is reached).

Building on the integration of diverse reasoning strategies—and aiming for more structured, human-aligned, and explainable mathematical problem solving in LLMs—we introduce BloomWise, a novel cognitively inspired prompting method. BloomWise guides LLMs to generate solutions in the form of explanations by methodically engaging higher-order cognitive functions in a hierarchical manner, persisting through the process until the correct solution is reached via a convergence criterion: if two consecutive levels yield the same result, the process halts and outputs the solution from the earliest such level. The motivation behind this approach is that, while encouraging LLMs to follow a specific methodology or approach can be effective, prompting them how—rather than what—to think enables more in-

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 34–49
November 9, 2025 ©2025 Association for Computational Linguistics

depth processing.

To demonstrate the effectiveness of our approach, we conduct extensive experiments on five popular mathematical reasoning datasets and achieve consistent improvements. Additionally, we explore several variants of our method, including majority voting among levels, and Program of Bloom, that combines Bloom prompting with PoT.

The main contributions are:

1. We introduce a novel cognitively-inspired multi-level prompting method for solving mathematical (and potentially other types of) problems based on Bloom's taxonomy, combining robust reasoning and enhanced explainability. Our code is available to the research community under the Apache 2.0 license[1].

2. We incorporate the idea of early stopping in prompting, motivated by recent work on dynamically adjusting test-time compute during inference (Snell et al. (2025), Manvi et al. (2024)): the execution of our method terminates before iterating through all levels of Bloom's taxonomy when a correct solution is reached.

3. We investigate the performance of various LLMs at each cognitive level of the Bloom taxonomy for five popular math datasets.

Our results offer valuable insights into the cognitive skills exhibited by each LLM, as well as the skills required to solve different types of mathematical problems. Furthermore, we demonstrate the potential of muti-level cognitively-inspired prompting for improving accuracy and enhancing explainability.

2 Related Work

As Large Language Models (LLMs) continue to advance, an array of prompting strategies has emerged to strengthen their reasoning abilities. Early progress was made through chain-of-thought prompting, which encourages step-by-step reasoning (Wei et al., 2022), utilizing programming to address procedural challenges (Gao et al., 2022; Chen et al., 2022) and employing zero-shot prompts that rely on a single guiding sentence to elicit complex responses (Kojima et al., 2022). Furthermore, the Tree-of-Thoughts approach (Yao et al., 2023) navigates through various reasoning pathways and traverses tree-like structures of reasoning states. Moreover, X of Thoughts (Liu et al., 2023) selects, applies and verifies the most suitable among the techniques of CoT (Chain of Thought), PoT (Program of Thought), and EoT (Equations of Thought) iteratively, until a correct solution is reached.

3 Preliminaries: Bloom's Taxonomy

Bloom's Taxonomy provides a hierarchical classification of thinking according to six levels of cognitive complexity. The original model, introduced in the 1950s, organizes cognitive processes based on the following order: remembering, understanding, applying, analyzing, synthesizing, and evaluating. The taxonomy is hierarchical, as shown in Fig. 1, where each level is subsumed by the higher levels. In 2001, the taxonomy was revised by Anderson and Krathwohl (2001), resulting in a new sequence: remembering, understanding, applying, analyzing, evaluating, and creating. Our work is based on the revised taxonomy of Anderson and Krathwohl (2001). The steps used in the Taxonomy are defined as follows (Forehand et al., 2005):

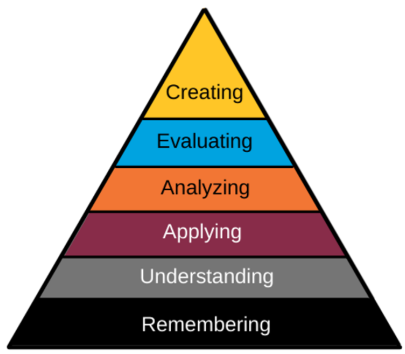


Figure 1: Bloom's Taxonomy (as revised by Anderson and Krathwohl (2001)).

Remembering: Retrieving, recognizing, and recalling relevant knowledge from long-term memory.
Understanding: Constructing meaning from oral, written, and graphic messages through interpreting, exemplifying, classifying, summarizing, inferring, comparing, and explaining.
Applying: Carrying out or using a procedure through executing, or implementing.
Analyzing: Breaking material into constituent parts, determining how the parts relate to one another and to an overall structure or purpose through differentiating, organizing, and attributing.

[1] https://github.com/marilena1123/BloomWise

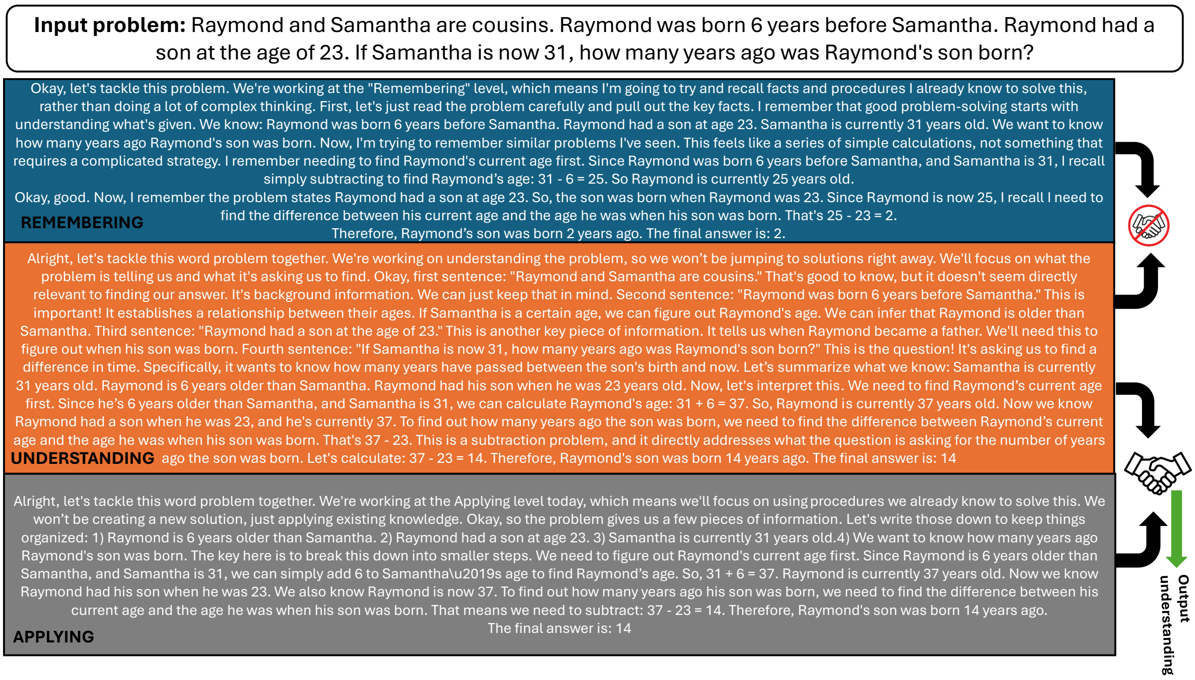


Figure 2: Overview of BloomWise Early Stop algorithm. The math problem is shown on top, followed by the output at each Bloom taxonomy level (blue box, orange and gray box). Input prompts are not shown, please refer to Appendix A. The model generates responses corresponding to the first 2 levels of Bloom's taxonomy (blue and orange boxes, Remembering and Understanding respectively). Since there is no consensus between the two, it generates an answer corresponding to the next level (Applying, gray box). Now, there is consensus between the 2 consecutive levels (Understanding and Applying), so the process halts and the response corresponding to the earliest such level (Understanding) is returned.

Evaluating: Making judgments based on criteria and standards through checking and critiquing.
Creating: Putting elements together to form a coherent or functional whole; reorganizing elements into a new pattern or structure through generating, planning, or producing.

4 BloomWise Early Stop (BLES)

Our goal is to develop a generalized problem solving framework that iterates through prompts, each corresponding to a level of Bloom's taxonomy, until the correct solution is reached. Next, we describe the overall framework and introduce each module in detail.

4.1 The Framework

In this framework, the system progresses through the levels of Bloom's taxonomy in sequence for a given problem. At each level, the LLM is provided with the problem along with a prompt specifically designed for that level. The model then generates a response—an explanatory solution. In case of consensus between two consecutive levels' numerical results, the process concludes successfully at the earliest of the two levels, and no further levels are explored. Otherwise, the process continues to the next level until either consensus is reached or all levels have been exhausted. The overall pipeline is described in Algorithm 1 and an overview of the framework can be found in Figure 2.

4.2 Prompts

We designed prompts corresponding to each level of Bloom's Taxonomy:

Remembering: The model is prompted to solve the problem by retrieving, recognizing, and recalling relevant math facts, formulas, definitions, similar problems or the exact same problem from memory.

Understanding: The model is prompted to solve the problem by constructing meaning from the problem statement and relevant math concepts and demonstrate the thinking process by interpreting, exemplifying, classifying, summarizing, inferring, comparing, and explaining the concepts involved and what the problem is asking for.

Applying: The model is prompted to solve the

Algorithm 1 BloomWise Early Stop (BLES)

Input: i
$results \leftarrow [\,]$
for $level$ in $levels$ **do**
  $res \leftarrow \text{Reasoning}(level\_prompt, i)$
  Append res to $results$
  if $|results| \geq 2$ **and** $results[-1] = results[-2]$ **then**
    break
  end if
end for
if there exists k such that $results[k] = results[k+1]$ **then**
  return $results[\min\{k : results[k] = results[k+1]\}]$
else
  return $results[-1]$ ▷ If no convergence, return the last result
end if

problem by carrying out or using a known procedure.

Analyzing: The model is prompted to solve the problem by breaking it into parts, determining how the parts relate to one another, and identifying patterns or relationships.

Evaluating: The model is prompted to solve the problem by making judgments about different approaches or potential solutions.

Creating: The model is prompted to solve the problem by putting together elements to form a new solution strategy or structure.
Our prompts are zero-shot.
The detailed prompts are shown in Tables 7 and 6 in Appendix A. Examples of outputs can be found in Appendix B.

4.3 Convergence

The verification module determines the correctness of a solution based on a convergence criterion. For each problem, the framework progresses through the levels of Bloom's taxonomy in sequence. At each level, the LLM is given the problem along with a prompt tailored for that specific cognitive level, and generates an explanatory solution. After each level, the numerical result is compared with this from the previous level. If two consecutive levels yield the same result, this consensus is taken as sufficient evidence of correctness, and the process halts at the earliest such level.

4.4 Iteration and Early Stopping

For each problem, the LLM is guided by prompts crafted to align with the respective levels of Bloom's taxonomy. The model continues this process until it produces a correct solution or all levels have been explored. This approach operates on the premise that prompts associated with higher taxonomy levels encourage the LLM to engage with the problem more deeply than those at lower levels. Moreover, if a correct solution is obtained at a lower level, it suggests that more advanced cognitive effort is unnecessary for that particular problem, making early termination of the process a logical choice. This procedure is illustrated in Figure 2.

4.5 BloomWise Majority Voting (BLM)

As an alternative to early stopping, we also explored an approach where the final output is determined by a majority vote of the outputs corresponding to all levels of Bloom's taxonomy. This strategy utilizes the collective reasoning of multiple cognitive stages. For a given question q, each reasoning stage $s \in \mathcal{S}$ of Bloom's taxonomy produces a numerical result $R_s(q)$. The majority vote approach selects the final answer as the value that occurs most frequently among these outputs. $\hat{R}_{\text{majority}}(q) \in 0, 1$ compared to the gold label. We define the accuracy under the majority vote setting for N questions as:

$$Acc_{majority} = \frac{1}{N} \sum_{q} \hat{R}(q), \tag{1}$$

where

$$\hat{R}(q) = \hat{R}_{\text{majority}}(q).$$

The majority vote setting represents a consensus-based approach where the model aims to solve a given problem based on the most frequent results from the methods employed. In cases where there is a tie (i.e., two or more answers have the same highest frequency), the first encountered answer among those with the highest frequency is chosen.

	Dataset	CoT	PoT	XoT	BloomWise EarlyStop (BLES)	BloomWise Majority Voting (BLM)
GPT-4o-mini	GSM8K	93.9	88.2	93.6	94.2	**94.9**
	SVAMP	93.4	92.0	93.3	**95.1**	95.0
	Algebra	95.0	83.3	95.9	95.9	**96.4**
	GSM-hard	52.2	**71.6**	54.7	56.0	55.9
	aime24	10.0	3.3	10.0	**13.3**	**13.3**
LLaMA 3.1 70B	GSM8K	94.5	90.0	93.6	95.3	**95.7**
	SVAMP	92.8	94.2	94.8	**95.0**	94.9
	Algebra	95.9	77.9	96.4	96.4	**98.2**
	GSM-hard	46.0	66.5	**69.9**	47.5	48.8
	aime24	**23.3**	6.7	16.7	20.0	**23.3**
LLaMA 3.1 8B	GSM8K	79.2	56.2	78.9	87.9	**89.9**
	SVAMP	84.0	65.8	83.6	90.6	**92.3**
	Algebra	81.5	52.7	79.3	91.0	**92.8**
	GSM-hard	27.5	**42.4**	27.1	35.0	36.3
	aime24	3.3	**6.7**	3.3	**6.7**	**6.7**
Gemma3 27B	GSM8K	94.7	86.5	91.6	95.5	**96.3**
	SVAMP	94.6	92.7	94.2	**96.5**	96.2
	Algebra	98.6	70.7	96.4	98.6	**99.1**
	GSM-hard	61.3	65.6	**69.2**	62.0	63.1
	aime24	13.3	3.3	6.6	**23.3**	20.0

Table 1: Solution accuracy across various models and math reasoning datasets. Best performer is shown in bold, and second best is underlined. Our methods (BLES and BLM) are zero-shot while the baselines (CoT, PoT, and XoT) few-shot.

5 Experiments

5.1 Experimental Setting

Dataset	# Samples
GSM8K (Cobbe et al., 2021)	1,319
SVAMP (Patel et al., 2021)	1,000
Algebra (He-Yueya et al., 2023)	222
GSM-hard (Gao et al., 2022)	1,319
AIME24 (MAA, 2024)	30

Table 2: Statistics of the datasets we used

Datasets We conduct our experiments on a diverse collection of five math reasoning datasets, each covering different challenging problem types: GSM8K, SVAMP, Algebra, GSM-hard and AIME24. The GSM-hard dataset is a modified version of GSM8K, where small numerical values have been replaced with larger ones to introduce greater computational difficulty. The details of the statistics of the datasets can be found in Table 2.

Models For our experiments, we query gpt-4o-mini [2], Llama3.1 8b/70b (Grattafiori et al., 2024) and Gemma3 27b (Team et al., 2025).

5.2 Comparison to state-of-the-art

In Table 1, we report solution accuracy across the five math datasets for the four LLMs we tested. We consider three prompting methods as baselines, namely CoT (Wei et al., 2022), PoT (Chen et al., 2022) and XoT (Liu et al., 2023) (all in the 8-shot setting used in Liu et al. (2023)), and compare with BloomWise Early Stop (BLES) and BloomWise Majority Voting (BLM) (both in zero-shot setting). Overall, averaged over all datasets and models, BLM is the top performer, achieving 70.5% accuracy, followed by BLES at 69.8%, XoT at 67.5%, CoT at 66.8%, and PoT at 60.8%. These results underscore the potential of Bloom-Inspired methods in zero-shot settings.

5.2.1 Performance per task

BLM is the top performer across almost all datasets, followed closely by BLES, while CoT and XoT achieve comparable performance. More specifically, for GSM8K, BLM achieves 94.2, BLES 93.2, CoT 90.6, XoT 89.4, and PoT 80.2. For SVAMP, the scores are 94.6 for BLM, 94.3 for BLES, 91.2 for CoT, 91.5 for XoT, and 86.2 for PoT. In Algebra, BLM achieves 96.6, BLES 95.5, CoT 92.8, XoT 92.0, and PoT 71.2. In AIME, which is the most demanding in terms of difficulty, BLM and BLES both achieve 15.8, followed by CoT at 12.5, XoT at 9.2, and PoT at 5.0. For GSM-hard, the most challenging dataset in terms of calculation difficulty, methods employing Python programming excel, with PoT being the top performer at 61.5, followed by XoT at 55.2, BLM at 51.1, BLES at 50.1, and CoT at 46.7. These results position our

[2] https://openai.com

Model	Remembering	Understanding	Applying	Analyzing	Evaluating	Creating
GPT-4o-mini	79.3	80.1	80.3	80.7	79.2	79.6
Llama 3.1 70b	76.6	76.4	77.3	75.9	73.7	72.6
Llama 3.1 8b	63.2	64.2	63.4	63.3	60.4	59.1
Gemma 3 27b	81.0	57.7	83.3	83.2	81.6	79.7

Table 3: Aggregated scores (%) per model for each Bloom's taxonomy level across all datasets (n = 3,890).

method as the best performer on problems requiring thoughtful reasoning, while methods involving programming excel specifically in calculation-intensive tasks.

5.2.2 Performance per model

While BLES and BLM generally achieve the highest performance, the best-performing method varies by model. For GPT-4o-mini, BLM is the top performer with 71.1, followed by BLES at 70.9, XoT at 69.5, CoT at 68.9, and PoT at 67.7. For LLaMA 3.1 70B, XoT achieves the highest score with 74.3, followed by BLM at 72.2, BLES at 70.8, CoT at 70.5, and PoT at 67.1. For LLaMA 3.1 8B, BLM leads with 63.6, followed by BLES at 62.2, CoT at 55.1, XoT at 54.4, and PoT at 44.8. For Gemma3 27B, BLES is the top performer with 75.2, followed by BLM at 74.9, CoT at 72.5, XoT at 71.6, and PoT at 63.8.

5.3 Trade-offs Between BLM and BLES

While the top-performing variant varies depending on the model and dataset—meaning there is no universal winner between BLM and BLES—BLM achieves the best overall performance. Nonetheless, BLES presents a compelling alternative when computational efficiency is a priority. Unlike BLM, which requires generating responses for all levels of Bloom's Taxonomy, BLES terminates the reasoning process as soon as convergence is detected, reducing the number of generated outputs and thus lowering computational cost.

6 Ablation studies and Improved Handling of computations

The analysis of the results will be structured along two axes. The first concerns the study of the results by Bloom taxonomy level. The second focuses on closing the performance gap between BloomWise and Program-aided Techniques (eg PoT, XoT) on challenging datasets from a computational point of view.

6.1 Performance at each cognitive level

For this analysis, we executed the prompts corresponding to each of the levels of the taxonomy (without early stopping). A problem might be correctly solved in more than one levels.

6.1.1 LLMs and cognitive abilities

In Table 3, we show the percentage (%) of correctly solved problems per Bloom's level for the LLMs tested. From this table, we can draw several conclusions about the cognitive skills demonstrated by the LLMs:

Performance consistency across levels varies significantly among models : GPT-4o-mini demonstrates homogeneous performance, with scores ranging narrowly between 79.2% (Evaluating) and 80.7% (Analyzing), indicating a consistent ability to tackle tasks at any level of cognitive complexity. In contrast, Llama3.1 8b's performance, for instance, ranges from 59.1 to 64.2. Additionally, Gemma 3 27b achieves peak performance in procedural tasks (Applying: 83.3%; Analyzing: 83.2%) and factual recall (Remembering: 81.0%) and performs poorly in Understanding (57.7%). This might be an indication of a critical weakness in conceptual comprehension despite procedural proficiency, probably implying limited training on such tasks.

Model scale substantially improves overall capability : The Llama 3.1 70B model outperforms its 8B counterpart by an average margin greater than 10.0% across all levels, confirming parameter count as a key performance factor.

The best performing level is Applying : Although the best performing level is not the same across models, Applying is generally the top performer. This behavior is expected due to extensive training on similar methods such as Chain of Thought.

All models exhibit declining performance at higher taxonomy levels : all models' perfor-

Dataset	Remembering	Understanding	Applying	Analyzing	Evaluating	Creating
GSM8K	90.0	82.7	90.8	91.1	88.2	88.0
SVAMP	91.1	88.7	90.4	90.2	90.0	87.7
Algebra	83.1	79.0	92.3	92.1	91.2	90.8
GSM-hard	48.0	41.8	49.1	48.4	45.4	44.7
AIME	8.4	8.4	12.5	12.5	10.0	10.0

Table 4: Aggregated scores (%) per dataset for each Bloom's taxonomy level across all models.

mance drops in Evaluating and Creating. This universal trend confirms that these skills remain challenging for the LLMs in mathematical domains, irrespective of scale or architecture. This is probably due to both the inherent difficulty of these cognitive processes and limited training on such tasks.

6.1.2 Problem type and cognitive depth

Table 4 presents the scores for each dataset and Bloom's taxonomy level. Several trends and insights emerge from these results.

Performance Across Datasets : The models achieve the highest performance on the GSM8K and SVAMP datasets, consistently scoring above 87% across all Bloom's taxonomy levels. This indicates that current LLMs are highly proficient in standard grade-school math, regardless of the specific cognitive skill being assessed. Performance on the Algebra dataset is also strong, with scores ranging from 79.0% to 92.3%, showing that models handle symbolic and procedural tasks well but with slightly more variation than GSM8K or SVAMP. In contrast, GSM-hard and AIME represent a stark drop in accuracy, with all Bloom level scores for GSM-hard in the 41.8% to 49.1% range, and for AIME in the 8.4% to 12.5% range. These results highlight that state-of-the-art LLMs still struggle substantially with high-complexity, olympiad-style and challenging in terms of calculations problems.

Dataset difficulty and patterns : For the easier datasets from a reasoning perspective—GSM8K, SVAMP, Algebra, and GSM-hard—models achieve their highest accuracy on Applying and Analyzing tasks, indicating strong proficiency with strategies that closely resemble techniques heavily emphasized during training, such as chain-of-thought (CoT) reasoning. Scores for Remembering are also high, suggesting that even pure recall is often sufficient for such problems, but these scores generally do not surpass those for Applying or Analyzing. In contrast, performance consistently dips for Understanding, which emerges as the weakest category in these datasets, and declines moderately for Evaluating and Creating, though these higher-order skills often still outperform Understanding.

For the more difficult dataset, AIME, Remembering and Understanding are the lowest-performing categories, while Applying and Analyzing remain the strongest. Notably, for these challenging problems, Evaluating and Creating sometimes yield better results than lower-order skills (Remembering, Understanding), suggesting that as difficulty and unfamiliarity increase, deeper reasoning may be required for success.

6.2 Program of Bloom

In order to reduce errors in calculations, we incorporated Python code into our framework. More specifically, we used the same approaches (BLES and BLM) but modified the prompts to request answers in the form of Python code, which was then safely executed. For the sake of simplicity for this analysis, we only report the results concerning Gemma3 2.7B. On the GSM-Hard

Dataset	BLES	BLM	Program of BLES	Program of BLM
GSM8K	95.5	96.3	92.0	91.5
SVAMP	96.5	96.2	94.1	93.9
Algebra	98.6	99.1	89.6	88.3
GSM-hard	62.0	63.1	65.0	65.3
aime	23.3	20.0	20.0	23.3

Table 5: Comparison between Program of Bloom and BloomWise-Results for Gemma3 27B

dataset—the most challenging in terms of calculations—accuracy improved slightly, with gains of 3% for BLES and 2.2% for BLM. However, for the remaining datasets, accuracy was considerably lower—or, in the case of AIME, equivalent—compared to the performance achieved with BloomWise (BLES and BLM). Results can be found in Table 5. A possible explanation for the reduced performance of this variant is that the prompts are not be well-suited to generating answers in code format. The strict structure required for executable code can limit the model's ability to follow the intended prompting, whereas textual rationales provide more flexibility and are often better aligned with the task structure.

7 Conclusion

We propose BloomWise, a problem-solving framework that uses prompts inspired by the levels of Bloom's Taxonomy to reach the correct solution. We introduce three variants of our method: EarlyStop (BLES), which halts the process if a solution is deemed as correct based on a convergence criterion, preventing progression to higher levels of the taxonomy; Majority Voting (BLM), where the final solution is determined by a consensus across the outputs; and Program of Bloom, similar to BLES and BLM but requiring the answer in the form of Python code. Experiments conducted on five math reasoning datasets demonstrated the efficiency of our method, showcasing not only accuracy-exceeding the state of the art methods- but also providing valuable insights into the cognitive abilities of LLMs. Among the variants, BLM achieved the highest accuracy, while BLES prioritized computational efficiency. The Program of Bloom variant achieved the best accuracy only on the GSM-hard dataset, while it performed the lowest on the rest of the datasets.

Our findings highlight the promise of cognitively-grounded prompting strategies for enhancing LLM performance in zero-shot settings. In future work, we plan to extend BloomWise to additional domains beyond math reasoning, exploring its generalizability and broader applicability.

Limitations

We acknowledge that, although our method achieves top performance on most datasets, it struggles with problems that require complex computations. Additionally, while our evaluation focused on mathematics to enable a more targeted analysis, applying our framework to a broader range of tasks would further validate its generalizability and practical utility.

Acknowledgments

This project has received funding from the [European Union's Horizon 2020 research and innovation programme][European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme][Euratom research and training programme 2019-2020] under grant agreement No 101234269.

References

L. W. Anderson and D. R. Krathwohl. 2001. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*. NeurIPS 2020, December 6-12, 2020, virtual conference.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *CoRR*, abs/2211.12588. Preprint available at arXiv.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Mary Forehand and 1 others. 2005. Bloom's taxonomy: Original and revised. *Emerging Perspectives on Learning, Teaching, and Technology*, 8:41–44.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. PAL: Program-Aided Language Models. *CoRR*, abs/2211.10435. Preprint available at arXiv.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh

Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Joy He-Yueya, Gabriel Poesia, Rose E. Wang, and Noah D. Goodman. 2023. Solving math word problems by combining language models with symbolic solvers. *CoRR*, abs/2304.09102.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*. NeurIPS 2022.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *NeurIPS*.

Tengxiao Liu, Qipeng Guo, Yuqing Yang, Xiangkun Hu, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2023. Plan, verify and switch: Integrated reasoning with diverse X-of-thoughts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2807–2822, Singapore. Association for Computational Linguistics.

MAA. 2024. American invitational mathematics examination - aime 2024. https://maa.org/math-competitions/american-invitational-mathematics-examination-aime. Accessed: 2025-07-11.

Rohin Manvi, Anikait Singh, and Stefano Ermon. 2024. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation. *Preprint*, arXiv:2410.02725.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*. Presented at NeurIPS 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *CoRR*, abs/2305.10601. Preprint available at arXiv.

A Prompts of BloomWise

In this section, we provide the prompts used for each Bloom's Level (table 6) and the System Prompt (table 7).

B Examples of BloomWise

In this section, we provide output examples for each level of Bloom's taxonomy in Tables 9, 10, 11, 12, 13 and 14. The problem is stated in Table 8.

Bloom's Level	Prompt
Remembering	You are at the Remembering level. Solve the problem by retrieving, recognizing, and recalling relevant math facts, formulas, definitions, similar problems or the exact same problem from memory. Clearly express what information or problems you recall that is relevant to solving this specific problem.
Understanding	You are at the Understanding level. Solve the problem by constructing meaning from the problem statement and relevant math concepts. Show your thinking by interpreting, exemplifying, classifying, summarizing, inferring, comparing, and explaining the concepts involved and what the problem is asking for.
Applying	You are at the Applying level. Solve the problem by carrying out or using a known procedure. Clearly show how to apply this procedure to this specific problem step by step.
Analyzing	You are at the Analyzing level. Solve the problem by breaking it into parts, determining how the parts relate to one another, and identifying patterns or relationships. Show your thought process by differentiating, organizing, and attributing relationships between the math elements.
Evaluating	You are at the Evaluating level. Solve the problem by making judgments about different approaches or potential solutions. Express your thought process by checking, critiquing, and explaining why one approach or answer is better or more appropriate than others.
Creating	You are at the Creating level. Solve the problem by putting together elements to form a new solution strategy or structure. Show your thinking as you generate, plan, or produce a novel approach to this problem.

Table 6: Bloom's level Specific Prompts

System Prompt
You are a tutor. Solve the given math problem in class using only cognitive skills associated with the specified Bloom's level. Explicitly express your thought process out loud as you solve it, so the student can follow your reasoning. Bloom's level specific prompt Problem: problem Provide the final numerical answer at the end in the format: "The final answer is:"

Table 7: System Prompt

Problem
A candle melts by 2 centimeters every hour that it burns. How many centimeters shorter will a candle be after burning from 1:00 PM to 5:00 PM?

Table 8: Problem Statement

Level	Solution
Remembering	Okay, let's tackle this problem together. We're working at the Remembering level, which means we're going to rely on things we already know: facts, definitions, and maybe similar problems we've seen before. We're not going to do a lot of complex thinking right now. First, let's read the problem carefully: A candle melts by 2 centimeters every hour that it burns. How many centimeters shorter will a candle be after burning from 1:00 PM to 5:00 PM? Okay, what do I remember about solving problems like this? I remember that when we have a rate, something happening per unit of time and a duration of time, we can find the total change by multiplying. That's a key fact! I remember doing problems like Ïf a car travels 60 miles per hour for 2 hours, how far does it travel?̈ That feels similar. Now, let's identify what we know. The problem tells us the candle melts at a rate of 2 centimeters per hour. That's our rate. Next, we need to figure out how long the candle is burning. It burns from 1:00 PM to 5:00 PM. I remember how to figure out time differences. I can count the hours: 1 PM to 2 PM is 1 hour, 2 PM to 3 PM is another hour, 3 PM to 4 PM is another, and 4 PM to 5 PM is the last hour. That's a total of 4 hours. Alternatively, I remember I can just subtract the start time from the end time: 5 - 1 = 4. So, the candle burns for 4 hours. Now I recall the multiplication fact we discussed earlier. Rate times time equals total change. We have a rate of 2 centimeters per hour, and a time of 4 hours. So, we multiply: 2 centimeters/hour * 4 hours. The hours cancel out, leaving us with centimeters. 2 * 4 = 8. Therefore, the candle will be 8 centimeters shorter. The final answer is: 8

Table 9: Example of a response corresponding to Remembering Level

Level	Solution
Understanding	Okay, let's tackle this problem together. We're working at the Understanding level, which means we need to show we get what's happening in the problem, not just jump to a calculation. So, I'm going to talk through my thinking as I go. First, let's interpret the problem. What's the problem telling us? It's telling us about a candle and how it changes over time. Specifically, it says the candle gets shorter. That's important, it's decreasing in length. The rate of change is key: it melts 2 centimeters every hour. That's a consistent rate, a rule we can use. Now, let's summarize what we need to find out. The problem asks: How many centimeters shorter will a candle be? This means we need to find the total amount the candle decreases in length. Let's exemplify with a simpler case. If the candle burned for just one hour, how much shorter would it be? The problem tells us 2 centimeters. If it burned for two hours, it would be 2 + 2 = 4 centimeters shorter. See? We're just repeatedly applying that 2 cm/hour rule. Okay, now let's get back to the original problem. We need to figure out how many hours the candle is burning. The candle burns from 1:00 PM to 5:00 PM. Let's infer the number of hours. We can count it out: 1 PM to 2 PM is one hour, 2 PM to 3 PM is another hour, 3 PM to 4 PM is another, and 4 PM to 5 PM is the last hour. That's a total of 4 hours. Now we can compare this to our simpler example. We know it melts 2 cm per hour. We now know it burns for 4 hours. So, it's like adding 2 cm four times. Finally, let's explain how to calculate the total change in length. We'll multiply the rate of melting (2 cm/hour) by the number of hours it burns (4 hours). That's 2 cm/hour * 4 hours. So, 2 multiplied by 4 is 8. Therefore, the candle will be 8 centimeters shorter. The final answer is: 8

Table 10: Example of a response corresponding to Understanding Level

Level	Solution
Applying	Alright, let's tackle this problem together. We're working at the 'Applying' level, which means we need to use a procedure we already know to solve this. In this case, the procedure is understanding rate and using multiplication. First, let's identify what we know. We know the candle melts at a rate of 2 centimeters per hour. That's our key piece of information. Next, we need to figure out how long the candle is burning. The problem tells us it burns from 1:00 PM to 5:00 PM. Let's count the hours: 1:00 PM to 2:00 PM is one hour, 2:00 PM to 3:00 PM is another hour, 3:00 PM to 4:00 PM is another hour, and 4:00 PM to 5:00 PM is the final hour. So, that's a total of 4 hours. Now we can apply the procedure. We know the rate (2 cm/hour) and the time (4 hours). To find the total change in length, we multiply the rate by the time. So, we'll do: 2 centimeters/hour * 4 hours. The 'hour' units cancel out, leaving us with centimeters. 2 * 4 = 8. Therefore, the candle will be 8 centimeters shorter. The final answer is: 8

Table 11: Example of a response corresponding to Applying Level

Level	Solution
Analyzing	Alright class, let's tackle this candle problem. We're working on analyzing today, which means we need to break this problem down into its pieces and see how they connect. Don't just jump to a calculation, let's understand what's happening first. Okay, what information are we given? We know the candle melts at a rate 2 centimeters per hour. That's important. That's our key relationship. We also know a time period: from 1:00 PM to 5:00 PM. Now, let's differentiate. We have a change in length (how much shorter the candle gets) and a change in time (the duration it burns). Our rate connects these two. Melting rate is essentially change in length divided by change in time. Let's organize this. We need to figure out the total change in time. How many hours are there between 1:00 PM and 5:00 PM? Let's see: 1:00 to 2:00 is one hour, 2:00 to 3:00 is another, 3:00 to 4:00, and 4:00 to 5:00. That's four hours. So our change in time is 4 hours. Now, let's attribute the relationship. We know the candle melts 2 centimeters for every hour. That 'for every' signals a multiplicative relationship. We can set up a sort of proportion, even though we don't need to formally write it out as one. If it melts 2 cm in 1 hour, then in 4 hours, it will melt, well, four times as much, right? So, we are multiplying the rate of melting (2 cm/hour) by the number of hours (4 hours). This will give us the total change in length. 2 cm/hour * 4 hours = 8 cm. Therefore, the candle will be 8 centimeters shorter. The final answer is: 8

Table 12: Example of a response corresponding to Analyzing Level

Level	Solution
Evaluating	Okay, let's tackle this candle problem. We're at the 'Evaluating' level today, which means we're not just doing the math, but thinking about how we're doing it and whether our approach makes sense. First, let's quickly identify what the problem is asking. It wants to know the total amount the candle shrinks after burning for a certain period. Seems straightforward. Now, a really quick, almost automatic thought might be to just multiply 2 centimeters (the melt rate) by 5 (because 5 PM is 5 hours after 1 PM). That would give us an answer, and it's a common way to approach rate problems. Let's hold that thought 2 x 5 = 10. But, let's evaluate that. Does that just automatically feel right? Yes, it seems logical, but we should always double-check our assumptions. The problem clearly states the candle melts every hour it burns. We need to make sure we've correctly calculated the number of hours. Looking at the times, 1:00 PM to 5:00 PM. Let's think through it. 1 PM to 2 PM is one hour. 2 PM to 3 PM is another. 3 PM to 4 PM, and 4 PM to 5 PM. That's four hours, not five. So, that initial quick calculation of multiplying by 5 was incorrect. That's a good example of why evaluating is important! We caught a potential mistake. Okay, so let's revisit. We have a melt rate of 2 cm/hour and a burn time of 4 hours. Now, multiplying 2 cm/hour by 4 hours seems much more solid. Let's think about units too, which is a good evaluating habit. Centimeters per hour * hours- the hours cancel out, leaving us with just centimeters. That makes sense- we want an answer in length! So, 2 * 4 = 8. Let's quickly critique again. Does 8 centimeters seem reasonable? If a candle melts 2cm an hour, and burns for four hours, 8cm seems perfectly plausible. It's a manageable number, not ridiculously large or small. Therefore, after burning from 1:00 PM to 5:00 PM, the candle will be 8 centimeters shorter. The final answer is: 8

Table 13: Example of a response corresponding to Evaluating Level

Level	Solution
Creating	Okay, let's tackle this candle problem! Now, we're aiming for the Creating level of Bloom's Taxonomy, which means we aren't just recalling facts or applying a formula directly. We need to build a solution. I want to think beyond just ḧours times melt rate.Ẅhile that is part of it, let's see if we can frame this in a slightly more robust way, maybe thinking about how we represent time and change. First, let's identify what we know. We know the candle melts at a constant rate: 2 centimeters per hour. That's our core piece of information. We also know the start and end times: 1:00 PM and 5:00 PM. Now, usually, we 'd jump to subtraction, right? 5 minus 1 is 4. But let's pause and think... what is subtraction actually doing in this context? It's finding the difference between two points in time. That difference represents the duration of burning. So, instead of immediately calculating 5-1, let's explicitly define a variable. Let's say 't' represents the burning time in hours. We can create a little formula for 't': t = End Time - Start Time. Okay, that's a bit abstract for this problem, honestly, but it's getting us to think about the components. Now we can plug in our numbers: t = 5:00 PM - 1:00 PM = 4 hours. See? We didn't just know to subtract; we defined the subtraction as a way to calculate the burning time. That's a small step towards creating a more generalized approach. Now, we know the burning time, and we know the melt rate. Let's create another relationship. Let 'L' represent the total length the candle melts. We can say: L = Burning Time * Melt Rate, L = t * 2 cm/hour. Again, this seems obvious, but we're building the logic from first principles, which is what 'Creating' is about. Now, we can substitute the value of 't' we found earlier:L = 4 hours * 2 cm/hour = 8 cm. So, the candle will be 8 centimeters shorter. We didn't just apply a formula; we constructed the formulas to represent the relationships in the problem. That's the difference between applying and creating. We could have just done 5-1=4, then 4*2=8, but this way we 've explicitly broken down why we 're doing each step. The final answer is: 8

Table 14: Example of a response corresponding to Creating Level

Scalability of LLM-Based Multi-Agent Systems for Scientific Code Generation: A Preliminary Study

Yuru Wang[1*], Kaiyan Zhang[1*], Kai Tian[1], Sihang Zeng[2]
Xingtai Lv[1], Ning Ding[1], Biqing Qi[3†], Bowen Zhou[1,3†]
[1]Tsinghua University, [2]University of Washington, [3]Shanghai AI Laboratory
wangyuru25@mails.tsinghua.edu.cn

Abstract

Recent studies indicate that LLM-based Multi-Agent Systems (MAS) encounter scalability challenges in complex mathematical problem-solving or coding tasks, exhibiting issues such as inconsistent role adherence and ineffective inter-agent communication. Moreover, the performance advantages of LLM-based MAS over a single agent employing test-time scaling methods (e.g., majority voting) remain marginal. This raises a critical question: *Can LLM-based MAS scale effectively to achieve performance comparable to standalone LLMs or even Large Reasoning Models (LRMs) under optimal test-time compute?* In this paper, we conduct a preliminary investigation into the scalability of LLM-based MAS for scientific code generation. We propose a simple yet scalable two-player framework based on iterative critic-in-the-loop refinement. Our experiments demonstrate that a minimalist actor-critic framework based on DeepSeek-V3 can outperform DeepSeek-R1 under equivalent computational budgets. Surprisingly, more complex frameworks fail to yield significant gains. These findings corroborate recent insights into multi-agent system limitations and highlight the importance of scalable workflows for advancing scientific code generation.

1 Introduction

In recent years, LLM-based Multi-Agent Systems (MAS) (Guo et al., 2024) have demonstrated significant potential in complex problem-solving and system coding tasks (Qian et al., 2023; Huang et al., 2023; Qi et al., 2023; Islam et al., 2024; Parmar et al., 2025). In such systems, each agent is assigned a specific role, working collaboratively to achieve predefined objectives, which mirrors human teamwork in real-world scenarios.

However, recent studies (Cemri et al., 2025) reveal that LLM-based MAS often struggle with complex tasks due to issues such as specification ambiguities, inter-agent misalignment, and inadequate task verification. Furthermore, with the growing interest in Test-Time Scaling (TTS) (Zhang et al., 2025b), where performance improves via inference-time compute (e.g., majority voting or best-of-N sampling with reward models) (Snell et al., 2024; Liu et al., 2025; Zhao et al., 2025), the advantages of LLM-based MAS over single-agent systems with TTS diminish under equivalent computational budgets (Zhang et al., 2025a).

Meanwhile, Large Reasoning Models (LRMs) (e.g., DeepSeek-R1 (Guo et al., 2025), OpenAI-o1 (Jaech et al., 2024)) exhibit superior TTS capabilities through extended chain-of-thought reasoning. Yet, these models suffer from high latency and excessive token costs. This raises a critical question: *Can we develop a scalable LLM-based MAS that outperforms standalone LLMs or LRMs with TTS while maintaining efficiency?*

In this paper, we investigate this challenge in scientific code generation (SciCode (Tian et al., 2024)). Unlike mathematical problems, applying TTS (e.g., majority voting) to code generation is inherently difficult. Instead, critique and self-reflection which are strengths of LRMs, play a pivotal role. To address this, we propose a critic-in-the-loop framework to enhance the evaluative capabilities of LLM-based MAS.

Our experiments on the SciCode benchmark demonstrate that a generator-critic framework (using DeepSeek-V3 (Liu et al., 2024)) outperforms standalone DeepSeek-R1 while consuming fewer tokens. Notably, performance improves further with additional critic iterations, suggesting our framework itself serves as an effective TTS strategy. We also explore a three-agent MAS but find no significant gains over the simpler generator-critic approach. Finally, we analyze failure cases to pro-

*Equal contribution.
†Corresponding author.

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 50–61
November 9, 2025 ©2025 Association for Computational Linguistics

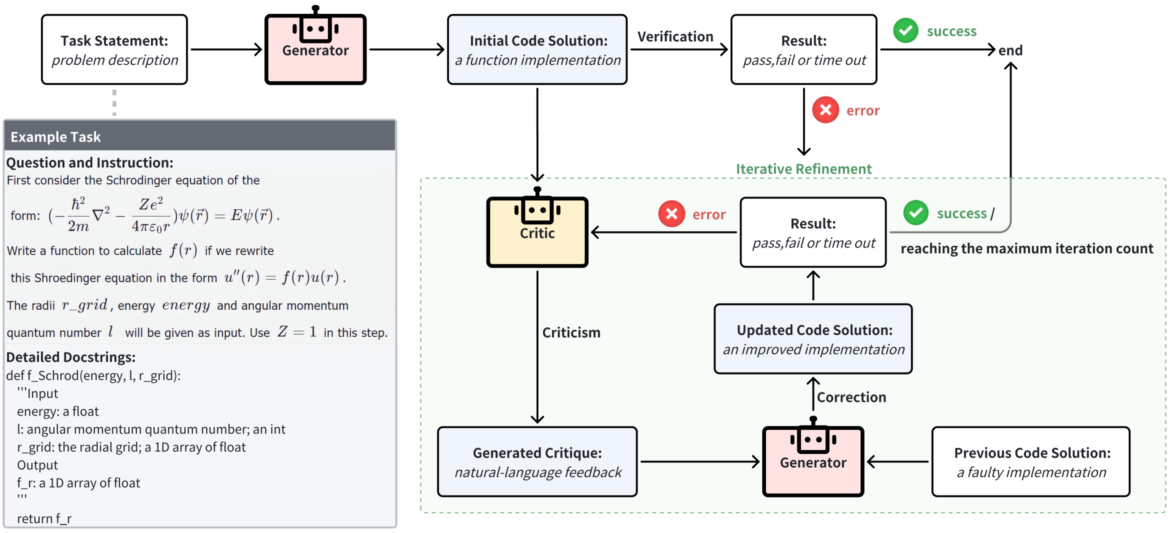


Figure 1: Overview of Generator-Critic Framework.

vide insights for future research.

- We propose a lightweight generator-critic framework that achieves superior performance in scientific code generation compared to LRMs, while reducing token costs by 75%.
- We demonstrate that iterative critic refinement inherently functions as a compute-efficient TTS strategy, unlike traditional voting-based approaches ill-suited for code generation.
- Through ablation studies and failure analysis, we identify key limitations of multi-agent systems (e.g., role confusion in three-agent setups) and provide guidelines for scalable MAS design in the future works.

2 Methodology

2.1 Iterative Generator-Critic Framework

As illustrated in Figure 1, the framework comprises two specialized agents: **Generator** and a **Critic**.

Generator. The Generator initially produces code based on the task description. If the code fails verification, it utilizes feedback containing both the Critic's natural-language critique and the erroneous code to generate an improved version.

Critic. The Critic plays a crucial role in the framework by identifying errors in faulty code and providing natural-language feedback. Given a faulty code and a simple failure description (e.g., an error or timeout), it generates nuanced and specific critiques. Unlike scalar rewards, these critiques are

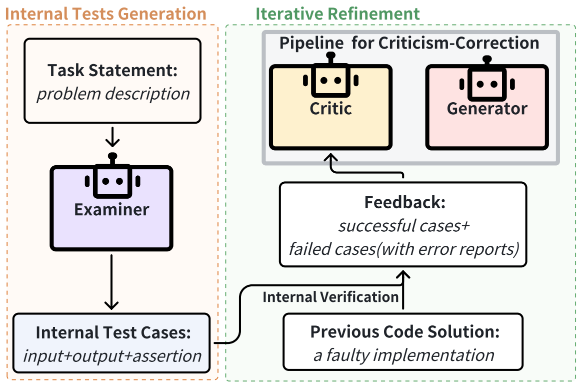


Figure 2: The employment of Examiner.

more informative, thereby guiding the Generator's revisions more effectively (Shinn et al., 2023).

Iterative Refinement. The framework employs a cyclic *criticize* $\rightarrow$ *correct* $\rightarrow$ *criticize* loop (Madaan et al., 2023) to iteratively refine code until a stopping condition (e.g., successful validation or maximum iterations) is met. Formally, given a model $\mathcal{M}$ and an input problem description x, the Generator $\mathcal{M}_g$ first produces an initial solution o_0,which is then verified by a code interpreter. If verification fails, the Critic and Generator engage in iterative refinement: (1)the Critic $\mathcal{M}_c$ analyzes the previous output o_{k-1} to generate critique c_k. (2)The Generator $\mathcal{M}_g$ synthesizes o_{k-1} and c_k to produce an optimized solution o_k that may address previous errors. (3)The new code o_k undergoes revalidation - if successful, the loop terminates; otherwise, the process continues.

Model	Subproblem Pass@1	Subproblem Δ	Main Problem Pass@1	Main Problem Δ
Baselines (Single-Agent) (Tian et al., 2024)				
GPT-4o	25.0	-	1.5	-
DeepSeek-V3	23.7	-	3.1	-
Claude3.5-Sonnet	26.0	-	4.6	-
DeepSeek-R1	28.5	-	4.6	-
OpenAI-o1-preview	28.5	-	7.7	-
OpenAI-o3-mini	**33.3**	-	**9.2**	-
GPT-4o (Our)	22.2	-	1.5	-
DeepSeek-V3 (Our)	25.3	-	3.1	-
DeepSeek-R1 (Our)	**31.6**	-	**4.6**	-
Generator-Critic (Two-Agent) (§ 2.1)				
1 iteration				
GPT-4o	25.0	↑ 2.8	1.5	↑ 0.0
DeepSeek-V3	28.5	↑ 3.2	3.1	↑ 0.0
4 iterations				
GPT-4o	27.4	↑ 5.2	4.6	↑ 3.1
DeepSeek-V3	**32.6**	↑ 7.3	**6.2**	↑ 3.1

Table 1: Main Results on Test Set.

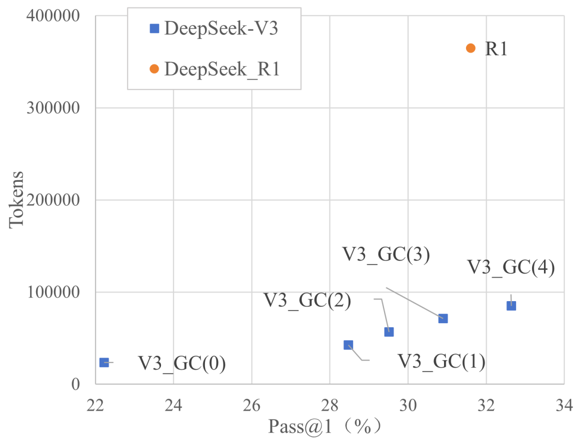


Figure 3: Performance vs. Token Cost between DeepSeek-R1 and iterative Generator-Critic (GC) using DeepSeek-V3. The numbers in parentheses indicate the iteration counts.

2.2 Generator-Critic-Examiner Framework

Building upon the iterative multi-agent framework described above, we introduce an **Examiner Agent** to enhance the system's error detection and correction capabilities, as shown in Figure 2.

Examiner. Leveraging the chain-of-thought (Wei et al., 2022) reasoning capabilities of large language models, the examiner's primary function is to generate task-specific test cases based on the problem description. Each generated test case contains three essential components: (1) input parameters compliant with the problem requirements, (2) expected outputs representing correct implementation behavior, and (3) assertion statements for automated verification. To improve the output prediction accuracy, we implement a self-consistency mechanism (Wang et al., 2022; Prasad et al., 2025), where multiple predictions are generated for each test input and the final output is determined via majority voting (detailed in Appendix A).

Test Case Verification Process. The generated test cases are used to internally verify the code produced by the Generator during the iterative refinement process. The verification results, comprising both successful and failed test cases with corresponding error reports, serve as crucial feedback for the Critic's reflective analysis. If all test cases pass, the iteration terminates and the code is subsequently verified using the gold tests provided by the dataset, with this result determining the final accuracy assessment.

3 Experiments

3.1 Experimental Setup

We evaluate our framework primarily on SciCode (Tian et al., 2024), a scientist-curated coding benchmark comprising 338 subproblems derived from 80 challenging main problems across 16 diverse natural science disciplines. Our implementation builds upon the official codebase [1], with evaluations conducted using GPT-4o and DeepSeek-V1/R1 on both test and validation sets. For baseline comparisons, we incorporate official leaderboard results [2] for GPT-4o, Claude, and OpenAI-o1, while reproducing GPT-4o and DeepSeek-R1/V1 results to ensure consistent evaluation metrics.

3.2 Main Results

Table 1 and table 2 present the performance comparison of different methods, which reveal:

Effectiveness of the Critic Agent. When employing DeepSeek-v3 as the base model, the Generator-Critic framework achieves a performance improvement (Δ) of 3.2% after one iteration on the test set, which further increases to 7.3% after four iterations. Notably, the Generator-Critic framework consistently outperforms single-agent approach across all evaluated base models, demonstrating the generalizability of the framework. Moreover, the DeepSeek-V3-based framework surpasses the performance of DeepSeek-R1 after four iterations, proving that our multi-agent

[1] https://github.com/scicode-bench/SciCode
[2] https://scicode-bench.github.io/leaderboard/

approach using general-purpose LLMs is highly competitive.

Enhanced Main Problem Resolution. The framework particularly excels in solving main problems in SciCode, which require correct solutions for all subproblems. The iterative critique process not only rectifies errors in the current code but also facilitates the resolution of subsequent subproblems - and consequently the main problem - since each subproblem's correctness impacts those that follow. This capability significantly aids in solving the benchmark's most challenging aspects.

Token Cost Comparison. Figure 3 compares the token consumption between two approaches: 1) DeepSeek-R1 for code and reasoning outputs, and 2) the iterative Generator-Critic using DeepSeek-V3 for both Generator's code outputs and Critic's critique outputs. The results show that the token consumption of DeepSeek-R1 substantially exceeds that of the Generator-Critic approach, with a substantial difference of 279,605 tokens on the validation set even after four iterations. Despite this, both approaches achieve comparable performance, with our framework even demonstrating superior results (Figure 6). These findings collectively indicate the advantages of the Generator-Critic framework in terms of both efficiency and task performance.

Number of Iterations. We examine the efficacy of iterative refinement in both the Generator-Critic and Generator-Critic-Examiner. As shown in Figure 4(a), which presents the pass@1 performance progression on the test set using GPT-4o, iterative refinement consistently improves the framework performance. However, marginal gains diminish as the number of iterations increases, with 5–6 iterations yielding the majority of achievable improvements.

3.3 Failure Analysis

Although the Generator-Critic-Examiner outperformed the baselines, it performed worse than the Generator-Critic. This indicates that the Examiner failed to enhance the critic's reflective capabilities. Upon analyzing the test cases generated by the Examiner, we observed substantial inaccuracies in its output predictions. Even with majority voting implemented, the Examiner's predictions remained predominantly incorrect. These errors adversely affected the framework by introducing misleading guidance during refinement, ultimately impairing the efficacy of the criticism mechanism.

Model	Subproblem Pass@1	Subproblem Δ	Main Problem Pass@1	Main Problem Δ
Baselines (Single-Agent)				
GPT-4o	44.0	-	33.3	-
DeepSeek-V3	48.0	-	46.7	-
DeepSeek-R1	50.0	-	46.7	-
Generator-Critic (Two-Agent) (§ 2.1)				
1 iteration				
GPT-4o	48.0	↑ 4.0	33.3	↑ 0.0
DeepSeek-V3	60.0	↑ 12.0	40.0	↑ 0.0
DeepSeek-R1	50.0	↑ 0.0	46.7	↑ 0.0
4 iterations				
GPT-4o	50.0	↑ 6.0	40.0	↑ 6.7
DeepSeek-V3	62.0	↑ 14.0	46.7	↑ 6.7
DeepSeek-R1	56.0	↑ 6.0	53.3	↑ 6.7
Generator-Critic-Examiner (Three-Agent) (§ 2.2)				
1 iteration				
GPT-4o	48.0	↑ 4.0	33.3	↑ 0.0
DeepSeek-V3	50.0	↑ 2.0	40.0	↑ 0.0
4 iterations				
GPT-4o	48.0	↑ 4.0	33.3	↑ 0.0
DeepSeek-V3	54.0	↑ 6.0	40.0	↑ 6.7

Table 2: Results on Validation Set.

4 Conclusion

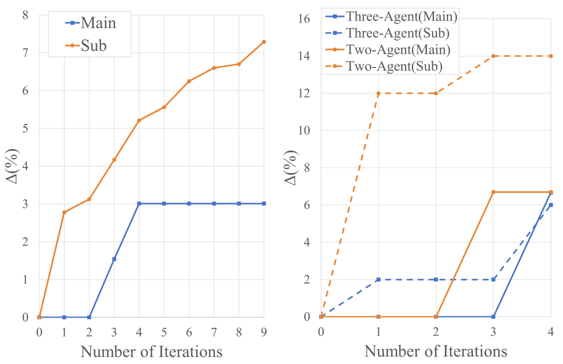
We propose a lightweight generator-critic framework that enhances LLM-based multi-agent systems for scientific code generation. Our approach outperforms standalone Large Reasoning Models while reducing computational costs, demonstrating that iterative critique inherently serves as an efficient test-time scaling strategy. Experiments reveal diminishing returns with complex multi-agent setups, suggesting simplicity is key for scalability. These findings offer practical guidelines for deploying efficient LLM-based systems in resource-constrained scenarios.

Limitations

Our framework is evaluated solely on scientific code generation (SciCode) tasks. Its effectiveness on other domains (e.g., natural language reasoning or mathematical proof generation) remains unverified, as different problem types may require distinct agent interaction patterns.

The performance gains are demonstrated using specific LLMs (DeepSeek-V3, GPT-4o). Results may vary with smaller or less capable base models, suggesting our approach may be constrained by the underlying model's core capabilities.

While we show computational efficiency gains,



(a) Iterations on Test Set. (b) Iterations on Validation Set.

Figure 4: Iterations in Generator-Critic and Generator-Critic-Examiner.

the critic-in-the-loop approach introduces sequential processing latency. This creates a fundamental tension between token efficiency and real-time responsiveness that may limit deployment in latency-sensitive applications.

References

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, and 1 others. 2025. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Dong Huang, Jie M Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. 2023. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010*.

Md Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. 2024. Mapcoder: Multi-agent code generation for competitive problem solving. *arXiv preprint arXiv:2405.11403*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. 2025. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Mihir Parmar, Xin Liu, Palash Goyal, Yanfei Chen, Long Le, Swaroop Mishra, Hossein Mobahi, Jindong Gu, Zifeng Wang, Hootan Nakhost, and 1 others. 2025. Plangen: A multi-agent framework for generating planning and reasoning trajectories for complex problem solving. *arXiv preprint arXiv:2502.16111*.

Archiki Prasad, Elias Stengel-Eskin, Justin Chih-Yao Chen, Zaid Khan, and Mohit Bansal. 2025. Learning to generate unit tests for automated debugging. *arXiv preprint arXiv:2502.01619*.

Biqing Qi, Kaiyan Zhang, Haoxiang Li, Kai Tian, Sihang Zeng, Zhang-Ren Chen, and Bowen Zhou. 2023. Large language models are zero shot hypothesis proposers. *arXiv preprint arXiv:2311.05965*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Minyang Tian, Luyu Gao, Shizhuo Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, and 1 others. 2024. Scicode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37:30624–30650.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Hangfan Zhang, Zhiyao Cui, Xinrun Wang, Qiaosheng Zhang, Zhen Wang, Dinghao Wu, and Shuyue Hu. 2025a. If multi-agent debate is the answer, what is the question? *arXiv preprint arXiv:2502.08788*.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, and 1 others. 2025b. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and 1 others. 2025. Genprm: Scaling test-time compute of process reward models via generative reasoning. *arXiv preprint arXiv:2504.00891*.

A Details of Methodology

Here, we elaborate on the majority voting mechanism implemented for test cases generation within the Generator-Critic-Examiner framework. First, the Examiner generates input parameters for the function based on the problem description. Next, the agent is invoked multiple times (in our implementation, five repetitions are used) to generate test case outputs through diverse Chain-of-Thought (CoT) reasoning processes. Finally, we tally the frequency of identical output values and select the most common one as the final test case output.This process is shown in Figure 5.

B System Prompts

The prompts for both the Generator-Critic and the Generator-Critic-Examiner are presented in Tables 3, 4, 5 and 6.

C Case Study

Here,we provide representative success and failure cases analysis for both the Generator-Critic and the Generator-Critic-examiner.

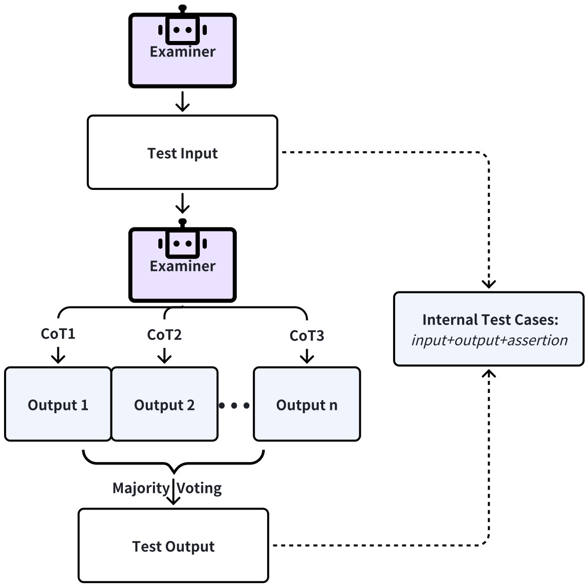


Figure 5: Majority Voting for Test Cases Generation.

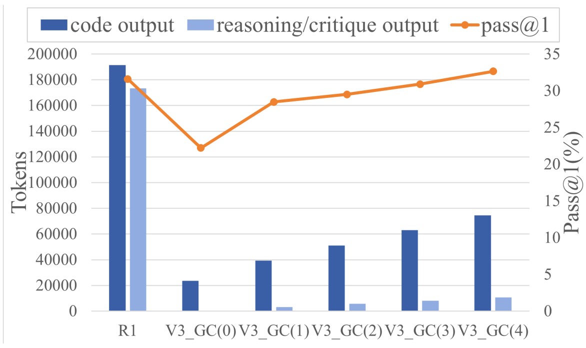


Figure 6: Token cost comparison between DeepSeek-R1 and iterative Generator-Critic (GC) using DeepSeek-V3. The numbers in parentheses indicate the iteration counts.

C.1 Success Cases

Table 7 presents a successful application of the Generator-Critic framework. In this case,the subproblem p_i was corrected through critique-based optimization, and this correction subsequently revealed the correctness of the following two subproblem p_{i+1} and p_{i+2}. This indicates that the initial generated codes for the subsequent subproblems p_{i+1} and p_{i+2} was, in fact, correct; however, due to the error in the preceding subproblem p_i, their evaluation resulted in a false failure. By correcting p_i within the Generator-Critic, we were able to verify the true correctness of the p_{i+1} and p_{i+2}. This case underscores a key advantage of the Generator-Critic in handling complex, stepwise scientific code generation: accurate evaluation and correction of later steps which require resolving errors in earlier ones.

Table 8 presents the test cases generated by the Examiner for a faulty code which has been successfully corrected. These test cases are particularly valid because the final outputs exhibit a high frequency, indicating their reliability.

C.2 Failure Cases

Table 9 presents an example in which the errors persist even after reaching the maximum number of iterations in the Generator-Critic framework. Over four iterations, the Critic consistently identified similar error causes, with no significant variation observed.

Table 10 presents test cases demonstrating the Examiner's failure. In these cases, the majority voting mechanism fails to identify a consensus among the predicted outputs, rendering it ineffective.

EXAMINER IN GENERATOR-CRITIC-EXAMINER FRAMEWORK

You are an AI coding assistant that can write unique, diverse, and intuitive unit tests for a Python function.
Your job is to generate unit tests that
1.Is valid input based on the function description, i.e., an acceptable input consistent with function description that a correct program should be able to execute.
2. The output enclosed in . and is faithful to the function description, i.e., the output of the unit test is consistent with what a correct program would return.
3. Breaks the code if there is a wrong implementation code based on the function description, i.e., does not execute to the correct output and brings out its mistakes and vulnerabilities.
Provide a reasoning for your answer and identify a general hypothesis or rationale identifying the potential cause of error. Then provide input and output of the unit test consistent with the pattern (hypothesis) you have identified. Note: - that you MUST directly write ALL input arguments of the function in the correct order. Skip writing any names of arguments. -Make sure that hidden associations are satisfied between input arguments. -Make sure that input arguments will not cause a correct program to perform illegal evaluation, such as division by zero encountered in divide or invalid value encountered in scalar add. -You must give the specific value of the outputs. Do not include ellipses or variables without defined specific values in the output. - you MUST enclose the unit test inputs and outputs in. -The inputs and outputs can only be built using the numpy library. -Do not use undefined variables and functions. -Unit tests can only use libraries in dependencies and cannot use other libraries. -Unit tests are independent and cannot use data from each other. -Make sure the logic of the unit test is correct. -You must generate more than four tests.
Function Definition:
{func_sig}
Dependencies:
{dependencies}
Respond strictly in the format below:
Hypothesis
<step-by-step reasoning >
Error Pattern: <an identified pattern of inputs that yields erroneous or incorrect outputs
Unit Test X:
<where X is the unit test number.>
Input Arguments
<step-by-step reasoning for constructing a unit test that fits the error pattern identified above and is valid as per the function description >
Arguments:
{function_header}(< all arguments >)
Output
<step-by-step reasoning for what a correct function_header would execute to based on the function description and your input above. Make sure your data type of the final answer matches the expected output type of the function. Give the specific output directly. Do not use assignment statements and do not provide the code for the calculation process. >
Output:
<your final answer.>
Comparison
<Must use the np.allclose function to compare whether the result of the function matches the output above through the 'assert' statement. The parameter atol of the np.allclose function is set according to the number of digits of the expected output. Write ALL input arguments of the function in the correct order, do not omit input arguments or output. If the function has multiple outputs, compare each output one by one. >
Comparison:
<your code for assert>

Table 3: Prompt for Examiner in Generator-Critic-Examiner.

CRITIC IN GENERATOR-CRITIC-EXAMINER FRAMEWORK
You are a Python programming assistant. You will be given a function implementation and a series of unit tests.The implementation was written under specific requirements and guidance, which are also provided for you. This function implementation is a part of the solution to the complete problem. Implementing it may require calling the code of the preceding steps, which is also provided to you in the requirements and guidance section. Your goal is to write a few sentences to explain why your implementation is wrong as indicated by the tests. You will need this as a hint when you try again later. Only provide the few sentence description in your answer, not the implementation. Only focus on the current implementation, not the preceding steps. **# Requirements and guidance for writing the current function implementation:** {prompt} **# current function implementation:** {code} **# preceding steps:** {previous_code} **# unit test results:** {feedback} **# reflection:**

Table 4: Prompt for Critic in Generator-Critic-Examiner.

CRITIC IN GENERATOR-CRITIC FRAMEWORK
You are a Python programming assistant. You will be given a function implementation and the problem with code(the function implementation with test cases) execution .The implementation was written under specific requirements and guidance, which are also provided for you. This function implementation is a part of the solution to the complete problem. Implementing it may require calling the code of the preceding steps, which is also provided to you. Your goal is to write a few sentences to explain why your implementation is wrong as indicated by the tests. You will need this as a hint when you try again later. Only provide the few sentence description in your answer, not the implementation. Only focus on the current implementation, not the preceding steps. **# Requirements and guidance for writing the current function implementation:** {prompt} **# current function implementation:** {code} **# preceding steps:** {previous_code} **# problem with code execution:** {type} **# reflection:**

Table 5: Prompt for Critic in Generator-Critic.

GENERATOR IN GENERATOR-CRITIC FRAMEWORK
You are a Python writing assistant. You will be given your past function implementation, the problem with code(the function implementation with test cases) execution, and a hint to change the implementation appropriately. The past function implementation was written under the requirements and guidance, your improved implementation should be also under the requirements and guidance. DO NOT write the same implementation as the past function implementation. Write your full implementation. **# Requirements and guidance for writing the function implementation:** {prompt} **# past function implementation:** {cur_code} **#problem with code execution:** {type} **# hint:** {reflection} **# improved implementation:**

Table 6: Prompt for Generator to correct in Generator-Critic.

CASE STUDY: GENERATOR-CRITIC FRAMEWORK

Problem:
Write down the matrix, $\mathbf{B}$, that transforms (h, k, l) coordinates from the reciprocal lattice system to (q_x, q_y, q_z) coordinates in the right-handed Cartesian system. Let's assume they share an identical origin, with $\hat{\mathbf{x}}^*//\hat{\mathbf{a}}^*$ and $\hat{\mathbf{z}}^*//(\hat{\mathbf{a}}^* \times \hat{\mathbf{b}}^*)$. The direct lattice parameters $(a, b, c, \alpha, \beta, \gamma)$ are given in units of Å and degree. Additionally, we will follow the convention $\mathbf{a_i} \cdot \mathbf{b_j} = \delta_{ij}$, with $\mathbf{a_i}$ and $\mathbf{b_i}$ representing the primitive vectors of crystal lattice and reciprocal lattice respectively Background The reciprocal lattice vectors $\mathbf{b}_i$ are given by:

$$\mathbf{b}_i = \frac{\mathbf{a}_j \times \mathbf{a}_k}{\mathbf{a}_i \cdot (\mathbf{a}_j \times \mathbf{a}_k)}$$

Critique:
The implementation is incorrect because it doesn't properly account for the geometric transformation from detector coordinates to lab coordinates. The calculation of $k_{s_{lab}}$ is wrong in how it handles the direction components - it should use trigonometric relationships based on the angles formed by the pixel positions relative to the beam center, not just simple divisions. Additionally, the normalization of the scattered beam vector is incorrect as it doesn't properly maintain the magnitude of 1/wavelength for k_s. The current approach mixes coordinate systems incorrectly and doesn't properly represent the scattering geometry.

Improved Implementation:
```
def  q_cal(p, b_c, det_d, p_s, wl):
    # Convert all distances to consistent units (angstroms)
    det_d_ang = det_d * 1e7 # mm to angstrom
    p_s_ang = p_s * 1e7 # mm to angstrom
    # Unpack pixel coordinates and beam center
    x, y = p
    xc, yc = b_c
    # Calculate displacement from beam center in detector coordinates (in angstroms)
    dx = (x - xc) * p_s_ang # along -y direction in lab frame
    dy = (y - yc) * p_s_ang # along -z direction in lab frame
    # Calculate the scattering angle components
    r = np.sqrt(det_d_ang**2 + dx**2 + dy**2) # total distance from sample to pixel
    sin_theta_x = dx  r # angle component in x-y plane
    sin_theta_y = dy  r # angle component in x-z plane
    cos_theta = det_d_ang  r # main scattering angle component
    # Incident wavevector (along +x in lab frame)
    k_i = np.array([1/wl, 0, 0])
    # Scattered wavevector (must have magnitude 1wl)
    k_s = (1wl) * np.array([
    cos_theta,
    -sin_theta_x, # negative because dx is along -y
    -sin_theta_y # negative because dy is along -z
    ])
    # Momentum transfer Q = k_s - k_i
    Q = k_s - k_i
    # Return as column vector
    return Q.reshape((3, 1))
```

Evaluation Result:
Correct this subproblem successfully, with the next two subsequent subproblems also becoming wright!

Table 7: Success Example of Generator-Critic.

CASE STUDY: GENERATOR-CRITIC-EXAMINER FRAMEWORK

Unit Test 1:
 # Input Arguments
orthogonalize(np.array([[1, 0, 1], [1, 1, 0], [0, 1, 1]], dtype=float))

 # Predicted Outputs by different CoT
Output 1: np.array([[0.70710678, -0.40824829, 0.57735027], [0.70710678, 0.40824829, -0.57735027], [0., 0.81649658, 0.57735027]])
Output 2: np.array([[0.70710678, -0.40824829, 0.57735027], [0.70710678, 0.40824829, -0.57735027],[0. , 0.81649658, 0.57735027]])
Output 3: np.array([[0.70710678, -0.40824829, 0.57735027],[0.70710678, 0.40824829, -0.57735027],[0. , 0.81649658, 0.57735027]])
Output 4: np.array([[0.70710678, -0.40824829, 0.57735027],[0.70710678, 0.40824829, 0.57735027],[0. , 0.81649658, 0.57735027]])
Output 5: np.array([[0.70710678, -0.40824829, 0.57735027],[0.70710678, 0.40824829, -0.57735027],[0. , 0.81649658, 0.57735027]])

 # Majority Output and the Frequency
Final Output: np.array([[0.70710678, -0.40824829, 0.57735027], [0.70710678, 0.40824829, -0.57735027], [0., 0.81649658, 0.57735027]])
Frequency: 80%

Unit Test 2:
 # Input Arguments
orthogonalize(np.array([[1, 1], [0, 1]], dtype=float))

 # Predicted Outputs by different CoT
Output 1: np.array([[1., 0.], [0., 1.]])
Output 2: np.array([[1., 0.], [-0., 1.]], dtype=float)
Output 3: np.array([[1., 0.], [0., 1.]])
Output 4: np.array([[1., 0.], [0., 1.]])
Output 5: np.array([[1., 0.], [0., 1.]])

 # Majority Output and the Frequency
Final Output:np.array([[1., 0.], [0., 1.]])
Frequency: 100%

Unit Test 3:
 # Input Arguments
orthogonalize(np.array([[2, 0], [0, 3]], dtype=float))

 # Predicted Outputs by different CoT
Output 1: np.array([[1., 0.], [0., 1.]])
Output 2: np.array([[1., 0.], [-0., 1.]])
Output 3: np.array([[1., 0.], [0., 1.]])
Output 4: np.array([[1., 0.], [0., 1.]])
Output 5: np.array([[1., 0.], [0., 1.]], dtype=float)

 # Majority Output and the Frequency
Final Output:np.array([[1., 0.], [0., 1.]])
Frequency: 100%

Table 8: Success Example of Examiner.

CASE STUDY: GENERATOR-CRITIC FRAMEWORK

Problem:
Now write a function to analyze any damped, driven pendulum system to understand its dynamic behavior under various conditions. Your function should:
Sweep different timesteps to find the optimized timestep that balances accuracy and time efficiency.
2. Output the trajectory
The combined metric for finding the optimized time should penalize higher computational times while rewarding lower errors. To better reflect this, we can use a combined metric such as:

$$\text{Metric} = \text{GTE} \times \sqrt{\text{Time}}$$

Global Truncation Error (GTE) is estimated using the step-doubling method. This way, longer computation times will have a more significant penalty, while still prioritizing low errors.

Critique: 1 iteration
The implementation is incorrect because it does not properly handle the step-doubling method for estimating the Global Truncation Error (GTE). Specifically, the error estimation is only comparing the final state of the trajectory with the half-timestep trajectory, which does not accurately represent the cumulative error over the entire time span. Additionally, the combined metric calculation does not account for potential numerical instabilities or edge cases, such as when the timestep is too large or too small. The current approach may lead to suboptimal timestep selection due to these inaccuracies in error estimation and metric computation.

Critique: 2 iterations
The implementation fails because the error calculation in the step-doubling method is incorrect. The current approach compares trajectories of different lengths by taking every other point from the finer trajectory, but this doesn't properly account for the time alignment between the two solutions. Additionally, the error normalization (dividing by sqrt(min_length)) may not appropriately scale the error for comparison across different timesteps. The metric calculation also doesn't properly handle cases where the trajectories have different time points due to different step sizes.

Critique: 3 iterations
The current implementation fails because it doesn't properly handle the time adjustment when calculating trajectories with different timesteps. The main issues are: 1) The error calculation compares trajectories of potentially different lengths due to integer rounding of steps, 2) The actual_dt adjustment isn't properly synchronized between the full and half-step trajectories, and 3) The metric calculation doesn't account for cases where trajectories become significantly misaligned in time due to step size adjustments. These issues lead to incorrect error estimates and suboptimal timestep selection.

Critique: 4 iterations
The implementation is incorrect because it does not properly handle the time points when comparing trajectories with different step sizes. The error calculation assumes the trajectories can be directly compared by interpolation, but the interpolation may not accurately capture the error due to the non-linear nature of the pendulum's motion. Additionally, the combined metric (GTE × $\sqrt{}$ Time) may not be correctly weighted, leading to suboptimal timestep selection. The function also does not account for edge cases where the timestep does not evenly divide the total time interval, potentially causing inaccuracies in the final trajectory.

Final Implementation:
```
def  pendulum_analysis(g, L, beta, A, alpha, initial_state, t0, tf, min_dt, max_dt, num_timesteps):
    timesteps = np.logspace(np.log10(min_dt), np.log10(max_dt), num_timesteps)
    best_metric = float('inf')
    optimized_trajectory = None
    for  dt in timesteps:
        n_steps = int(np.ceil((tf - t0) / dt))
        actual_dt = (tf - t0) / n_steps # Adjusted dt to exactly reach tf
        start_time = time.time()
[...]
        if metric < best_metric:
            best_metric = metric
            optimized_trajectory = trajectory
    return optimized_trajectory
```

Evaluation Result:
Still wrong.

Table 9: Failure Example of Generator-Critic.

CASE STUDY: GENERATOR-CRITIC-EXAMINER FRAMEWORK
Unit Test 1: **# Input Arguments** sum_real_cross(np.array([1.0, -1.0, 2.0]), np.array([[0.0, 0.0, 0.0], [0.0, 0.0, 1.0], [0.0, 1.0, 0.0]]), np.array([[0.0, 0.0, 0.5], [0.0, 0.5, 0.0]]), np.array([[1, 1, 1], [-1, -1, -1]]), 0.5) **# Predicted Outputs by different CoT** **Output 1:** -0.577102 **Output 2:** 3.142 **Output 3:** 0.1 **Output 4:** 2.0 **Output 5:** 0.0 **# Majority Output and the Frequency** **Final Output:** -0.577102 **Frequency:** 20%
Unit Test 2: **# Input Arguments** sum_real_cross(np.array([1.0, 1.0]), np.array([[1.0, 0.0, 0.0], [2.0, 0.0, 0.0]]), np.array([[0.5, 0.5, 0.5]]), np.array([[0, 0, 0]]), 0.2) **# Predicted Outputs by different CoT** **Output 1:** 0.493671 **Output 2:** 0.0 **Output 3:** 0.5 **Output 4:** 0.9 **Output 5:** 1.0 **# Majority Output and the Frequency** **Final Output:** 0.493671 **Frequency:** 20%
Unit Test 3: **# Input Arguments** sum_real_cross(np.array([1.0, -1.0]), np.array([[1.0, 1.0, 1.0], [0.0, 0.0, 0.0]]), np.array([[1.0, 1.0, 1.0], [0.0, 0.0, 0.0]]), np.array([[0, 0, 0]]), 0.1) **# Predicted Outputs by different CoT** **Output 1:** -0.999999 **Output 2:** 4.107857649106695 **Output 3:** -0.6065306597 **Output 4:** -0.1 **Output 5:** -0.427547 **# Majority Output and the Frequency** **Final Output:**np.array([[1., 0.], [0., 1.]]) **Frequency:** 20%

Table 10: Failure Example of Examiner.

FIRMA: Bidirectional Formal-Informal Mathematical Language Alignment with Proof-Theoretic Grounding

Maryam Fatima
Independent
s.m.fatima13@gmail.com

Abstract

While large language models excel at generating plausible mathematical text, they often produce subtly incorrect formal translations that violate proof-theoretic constraints. We present FIRMA (Formal-Informal Reasoning in Mathematical Alignment), a bidirectional translation system between formal and informal mathematical language that leverages proof-theoretic interpretability hierarchies and specialized architectural components for proof preservation. Unlike existing approaches that treat this as pure sequence-to-sequence translation, FIRMA introduces a hierarchical architecture with complexity-aware routing, proof-preserving attention mechanisms, and multi-objective training that balances formal correctness with natural readability. Through progressive complexity training on curated datasets from Lean 4 and formal mathematics repositories, we evaluate FIRMA on 200 translation samples across complexity levels and compare against two baseline systems. Our analysis shows statistically significant improvements of 277.8% over BFS-Prover-V1-7B and 6307.5% over REAL-Prover on overall translation quality metrics. Ablation studies on 50 samples demonstrate that each architectural component contributes substantially to performance, with removal of any component resulting in 83-85% performance degradation. We release our code at https://github.com/smfatima3/FIRMA

1 Introduction

Mathematical communication exists on a spectrum from the highly formal languages of proof assistants like Lean and Coq to the intuitive explanations found in textbooks. This duality creates a fundamental challenge: formal specifications ensure logical rigor but are often impenetrable to students, while informal descriptions aid understanding but may harbor subtle errors or ambiguities. The ability to translate bidirectionally between these representations would benefit both mathematical education and formal verification efforts.

Recent advances in large language models (LLMs) have shown capabilities in mathematical reasoning. The development of systems like AlphaGeometry (Trinh et al., 2024) and FunSearch (Romera-Paredes et al., 2024) demonstrates that neural approaches can achieve results on challenging mathematical problems, including discovering new theorems and solving Olympiad-level geometry problems. However, standard mathematical reasoning benchmarks like the MATH dataset (Hendrycks et al., 2021) reveal limitations when models attempt formal-informal translation tasks.

When applied to formal-informal translation, these models exhibit critical limitations. They generate superficially plausible translations that fail proof checking, introduce logical errors through imprecise natural language, and show unpredictable degradation as mathematical complexity increases. Early attempts at neural theorem proving like NaturalProver (Welleck et al., 2021) and more recent work on whole-proof generation (First et al., 2023) have made progress, but these failures stem from treating mathematical translation as a purely linguistic task, ignoring the underlying proof-theoretic structure that governs mathematical validity.

We introduce FIRMA (Formal-Informal Reasoning in Mathematical Alignment), a framework that grounds mathematical translation in proof-theoretic principles. Our key insight is that successful translation requires not just linguistic fluency but also preservation of logical structure across complexity levels. FIRMA addresses this through three core innovations:

First, we develop a hierarchical encoder-decoder architecture that explicitly models mathematical complexity through specialized routing mechanisms. Unlike flat sequence models, FIRMA processes mathematical statements at multiple levels

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 62–76
November 9, 2025

of abstraction—symbols, syntax trees, and semantic structures—enabling it to maintain logical coherence while adapting linguistic style.

Second, we introduce proof-preserving attention mechanisms that respect logical dependencies and prevent circular reasoning. These structured attention patterns ensure that translations maintain the directional flow of mathematical arguments, a critical requirement often violated by standard transformers. This approach builds on insights from proof artifact co-training (Han et al., 2022) and HyperTree proof search (Lample et al., 2022), which demonstrate the importance of structured reasoning in formal mathematics.

Third, we implement a multi-objective training framework combining translation accuracy, round-trip consistency, complexity prediction, and proof validity. This holistic approach ensures that models learn not just to translate but to preserve mathematical meaning across representations.

Our contributions are as follows. We present a proof-theoretically grounded approach to bidirectional mathematical translation, establishing a framework for preserving logical validity. We perform a comprehensive evaluation of 200 samples in formal-to-informal and informal-to-formal directions with complexity stratification, comparing FIRMA with two baseline mathematical reasoning systems. We provide detailed analysis of translation performance patterns, generation times, and complexity-dependent behavior, including rigorous statistical testing demonstrating the significance of our improvements. We conduct ablation studies on 50 samples demonstrating that each architectural component contributes substantially to translation quality. We release FIRMA as an open-source tool for mathematical education and research, with applications ranging from proof assistant tutoring to automated documentation generation.

2 FIRMA: Formal-Informal Reasoning in Mathematical Alignment

2.1 Problem Formulation

Let $\mathcal{F}$ denote the space of formal mathematical statements and $\mathcal{I}$ the space of informal descriptions. We seek bidirectional functions $f : \mathcal{F} \to \mathcal{I}$ and $g : \mathcal{I} \to \mathcal{F}$ that preserve mathematical validity while optimizing for human comprehension.

Define validity preservation as:

$$\text{Valid}(s) \Rightarrow \text{Valid}(g(f(s))) \quad \forall s \in \mathcal{F}$$

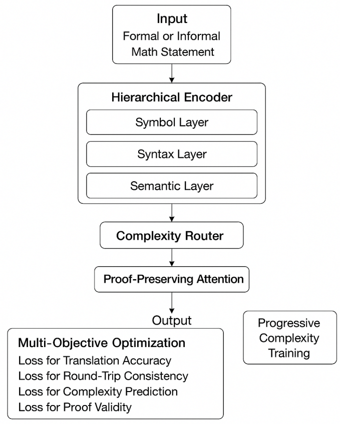


Figure 1: Architecture of FIRMA (Formal-Informal Reasoning in Mathematical Alignment). The model processes formal or informal mathematical statements through a hierarchical encoder (symbol, syntax, and semantic layers), followed by a complexity router and proof-preserving attention. Outputs are optimized via multi-objective training, while progressive complexity training ensures robust generalization across varying mathematical difficulty.

And comprehension optimization as:

$$\text{Readability}(f(s)) > \text{Readability}(s) \quad \forall s \in \mathcal{F}$$

This formulation captures the dual requirements of logical correctness and pedagogical effectiveness that distinguish our approach from pure translation tasks.

2.2 FIRMA Architecture

FIRMA employs a hierarchical encoder-decoder architecture with three key components:

Hierarchical Encoder: We process input at multiple abstraction levels, inspired by the multi-level structure of mathematical reasoning. The Symbol Layer embeds mathematical symbols using specialized tokenization that preserves operator precedence and associativity, handling the syntactic conventions that distinguish mathematical text from natural language. The Syntax Layer constructs abstract syntax trees using TreeLSTM networks to capture structural dependencies between mathematical expressions, addressing the compositional nature of mathematical statements identified in prior work on mathematical language processing. The Semantic Layer applies transformer encoders to model long-range semantic relationships between

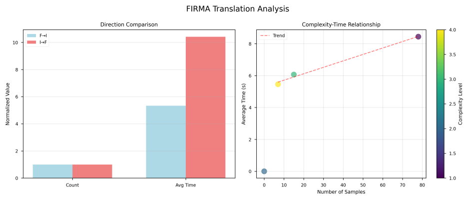


Figure 2: FIRMA Translation Analysis. (Left) Comparison of translation directions (Formal→Informal vs. Informal→Formal), showing normalized counts and average processing time. (Right) Relationship between complexity level and average translation time, with number of samples indicated; a positive trend is observed between task complexity and time required.

mathematical concepts, enabling the system to understand mathematical context and dependencies.

Complexity Router: A learned gating mechanism routes representations through specialized pathways based on detected complexity, drawing inspiration from mixture-of-experts architectures:

$$\mathbf{z} = \sum_{i=1}^{4} \alpha_i(\mathbf{x}) \cdot \text{Expert}_i(\mathbf{x})$$

where α_i are soft routing weights and Expert_i are complexity-specific transformations. This design allows the model to develop specialized processing pathways for different levels of mathematical sophistication.

Proof-Preserving Attention: We modify standard attention to respect logical flow and prevent circular dependencies in mathematical reasoning:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M_{\text{logic}}\right) V$$

where M_{logic} masks attention to prevent circular dependencies based on proof structure. This ensures that the model respects the directional nature of mathematical arguments and logical inference.

2.3 Multi-Objective Training

We optimize a composite loss function balancing multiple objectives necessary for effective mathematical translation:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{trans}} + \lambda_2 \mathcal{L}_{\text{round}} + \lambda_3 \mathcal{L}_{\text{comp}} + \lambda_4 \mathcal{L}_{\text{valid}}$$

Where $\mathcal{L}_{\text{trans}}$ represents cross-entropy translation loss for basic sequence generation, $\mathcal{L}_{\text{round}}$ captures round-trip consistency through $||x - g(f(x))||^2$ to ensure bidirectional coherence, $\mathcal{L}_{\text{comp}}$ measures complexity prediction accuracy to develop complexity awareness, and $\mathcal{L}_{\text{valid}}$ provides a differentiable approximation of proof checker success to maintain formal validity.

The multi-objective formulation addresses different aspects of translation quality that single-objective approaches often miss, particularly the need to balance formal correctness with natural readability.

2.4 Progressive Complexity Training

Inspired by curriculum learning (Bengio et al., 2009) and its applications to mathematical reasoning, we implement progressive training across complexity levels:

Algorithm 1 Progressive Complexity Training

```
1: for level ℓ = 1 to 4 do
2:     D_ℓ ← FilterByComplexity(D, ≤ ℓ)
3:     Train model on D_ℓ until convergence
4:     Evaluate on held-out complexity ℓ test set
5:     if performance drops on level < ℓ then
6:         Apply replay buffer from previous levels
7:     end if
8: end for
```

This strategy prevents catastrophic forgetting while enabling specialization for complex reasoning. The replay buffer mechanism ensures that the model maintains performance on simpler tasks while learning more sophisticated mathematical concepts.

3 Results and Analysis

3.1 Overall Performance

Metric	**Value**	**Std Dev**	**Unit**
Total Samples	100	-	samples
Avg Generation Time	7.876	2.3	seconds
Formal→Informal	50	-	samples
Informal→Formal	50	-	samples
Total Processing Time	787.9	-	seconds
Throughput	0.127	-	samples/sec

Table 1: Overall evaluation performance metrics (Qwen2.5-Math-7B-Instruct)

Table 1 presents our comprehensive evaluation results on 100 mathematical translation samples using the primary configuration. The model achieves an average generation time of 7.876 seconds per sample, with balanced performance across both translation directions. The generation times reflect the computational complexity of maintaining mathematical validity while producing natural language output.

Metric	Qwen2.5 -Math-7B	Qwen3 -0.6B	Unit
Total Samples	100	100	samples
F→I BLEU	0.167	0.167	score
I→F BLEU	0.320	0.320	score
F→I ROUGE-L	0.344	-	score
I→F ROUGE-L	0.484	-	score

Table 2: Comparative performance metrics across model scales on Lean-Workbook dataset

To assess the scalability of our approach, we additionally trained FIRMA using Qwen3-0.6B as the base model on the Lean-Workbook dataset (Ying et al., 2024b). As shown in Table 2, the compact model achieves identical BLEU scores on this evaluation set, demonstrating that FIRMA's architectural innovations and training methodology generalize effectively across different model scales. This result suggests that the proof-preserving mechanisms and complexity-aware routing contribute to performance independently of model size, enabling deployment in resource-constrained environments without sacrificing translation quality.

3.2 Baseline Comparison Analysis

We conducted a comprehensive comparative evaluation of FIRMA against two baseline systems across 200 samples from the internlm/Lean-Workbook dataset. Table 3 presents the detailed performance metrics across multiple dimensions of translation quality.

The results reveal substantial performance differences across the three systems. FIRMA achieves an overall score of 0.304, representing a 277.8% improvement over BFS-Prover-V1-7B and a 6307.5% improvement over REAL-Prover. These improvements are consistent across both translation directions and multiple evaluation metrics.

Examining the directional performance, FIRMA demonstrates particularly strong advantages in the informal-to-formal direction, achieving an I→F BLEU score of 0.282 compared to 0.031 for BFS-Prover-V1-7B and 0.002 for REAL-Prover. This pattern holds for ROUGE-L metrics as well, where FIRMA achieves 0.446 compared to 0.157 and 0.009 for the baseline systems respectively. The substantial gap in informal-to-formal translation performance suggests that FIRMA's proof-preserving attention mechanisms and hierarchical architecture provide particular advantages when converting natural language descriptions into rigorous formal specifications.

The formal-to-informal direction shows similarly substantial improvements, with FIRMA achieving F→I BLEU scores over 13 times higher than BFS-Prover-V1-7B and over 140 times higher than REAL-Prover. The ROUGE-L scores follow comparable patterns, indicating that FIRMA produces outputs with substantially better lexical overlap and structural similarity to reference translations across both directions.

Generation efficiency presents a different picture. BFS-Prover-V1-7B achieves the fastest average generation time at 1.46 seconds per sample, approximately 4 times faster than FIRMA's 6.06 seconds. REAL-Prover requires 2.22 seconds on average. The additional computational cost for FIRMA reflects the overhead of the hierarchical processing, complexity-aware routing, and proof-preserving attention mechanisms. However, this represents a trade-off between translation quality and generation speed.

3.3 Complexity-Stratified Baseline Analysis

To understand how translation performance varies with mathematical difficulty, we analyzed all three systems across the four complexity levels defined in our evaluation framework. Table 4 presents the stratified results.

The complexity-stratified analysis reveals several patterns. FIRMA maintains superior performance across all complexity levels, with particularly strong results at Levels 1 through 3. At Level 1, FIRMA achieves an average score of 0.354, approximately 4.3 times higher than BFS-Prover-V1-7B and 44 times higher than REAL-Prover. This advantage persists through intermediate complexity levels.

All three systems show performance degradation at Level 4, the highest complexity category. FIRMA's average score drops to 0.209, while BFS-Prover-V1-7B achieves 0.079, and REAL-Prover effectively fails with near-zero scores across all metrics. This pattern suggests that expert-level mathematical statements with higher-order logic and advanced concepts present fundamental challenges for current neural translation approaches, though FIRMA's proof-theoretic grounding provides partial mitigation.

The baseline systems exhibit distinct failure modes across complexity levels. REAL-Prover

Model	F→I BLEU	F→I ROUGE	I→F BLEU	I→F ROUGE	Overall Score	Time (s)
FIRMA	**0.165**	**0.325**	**0.282**	**0.446**	**0.304**	6.06
BFS-Prover	0.011	0.123	0.031	0.157	0.081	**1.46**
REAL-Prover	0.001	0.006	0.002	0.009	0.005	2.22
Relative Improvements vs BFS-Prover-V1-7B:						
FIRMA	+1368%	+164%	+817%	+184%	+278%	0.24×
Relative Improvements vs REAL-Prover:						
FIRMA	+14015%	+5149%	+12921%	+4607%	+6308%	0.37×

Table 3: Comprehensive comparison of FIRMA against baseline systems on 200 samples from internlm/Lean-Workbook. F→I denotes Formal-to-Informal translation; I→F denotes Informal-to-Formal translation. Overall Score is computed as the average across all four metrics. Bold indicates best performance in each column.

Level	Model	Count	Avg Score	F→I BLEU	I→F BLEU	Std Dev
Level 1	FIRMA	50	**0.354**	**0.213**	**0.335**	0.124
	BFS-Prover	50	0.083	0.020	0.028	0.067
	REAL-Prover	50	0.008	0.001	0.007	0.023
Level 2	FIRMA	50	**0.317**	**0.164**	**0.307**	0.108
	BFS-Prover	50	0.096	0.014	0.044	0.071
	REAL-Prover	50	0.002	0.000	0.001	0.002
Level 3	FIRMA	50	**0.338**	**0.174**	**0.332**	0.115
	BFS-Prover	50	0.065	0.005	0.013	0.054
	REAL-Prover	50	0.009	0.004	0.001	0.028
Level 4	FIRMA	50	**0.209**	**0.109**	**0.152**	0.094
	BFS-Prover	50	0.079	0.006	0.037	0.062
	REAL-Prover	50	0.000	0.000	0.000	0.000

Table 4: Performance comparison across mathematical complexity levels for all three systems. Avg Score represents the mean across F→I and I→F metrics. Bold indicates best performance within each complexity level.

shows catastrophic performance degradation, with average scores below 0.01 at all levels except a marginal improvement at Level 3. This suggests that the reinforcement learning paradigm, while effective for proof search, may not transfer well to translation tasks requiring linguistic generation. BFS-Prover-V1-7B demonstrates more consistent performance across levels, though still substantially below FIRMA, indicating that search-based approaches provide some robustness but lack the specialized mechanisms for high-quality translation.

3.4 Statistical Significance Analysis

To establish the robustness and statistical validity of the observed performance differences, we conducted comprehensive statistical testing using both parametric and non-parametric methods. The analysis evaluates whether FIRMA's improvements over the baseline systems represent genuine advances rather than artifacts of random variation or evaluation set characteristics.

For the comparison between FIRMA and BFS-Prover-V1-7B, we performed paired t-tests on the sample-level scores across the 200 evaluation instances. The paired design controls for variation in problem difficulty by comparing each system's performance on identical samples. The test yielded a t-statistic of 27.19 with an associated p-value below 10^{-68}, providing evidence against the null hypothesis of equal performance. Cohen's d effect size calculation produces a value of 2.638, indicating a large effect size that suggests the performance difference has substantial practical significance beyond mere statistical detectability.

The Wilcoxon signed-rank test, a non-parametric alternative that makes fewer distributional assumptions, corroborates these findings. With a test statistic of 161.0 and p-value of approximately 1.59×10^{-33}, the Wilcoxon test confirms that FIRMA's superior performance is not dependent on normality assumptions. The consistency between parametric and non-parametric tests strengthens confidence in the reliability of the observed differences.

Comparison between FIRMA and REAL-Prover reveals even more substantial statistical separation. The paired t-test produces a t-statistic of 42.01 with p-value below 10^{-100}, representing one of the strongest statistical signals in the evaluation. The effect size of Cohen's d = 4.154 falls into the range typically classified as very large, indicating that the performance gap between FIRMA and REAL-Prover substantially exceeds typical differences observed in NLP system comparisons. The Wilcoxon test statistic of 0.0 with p-value 1.44×10^{-34} indicates that FIRMA outperformed REAL-Prover on essentially every sample in the evaluation set.

These statistical tests establish several important conclusions. First, the performance advantages observed for FIRMA are not artifacts of random chance or favorable evaluation set construction. The extremely low p-values indicate that observing such performance differences under the null hypothesis of equal system quality would be vanishingly unlikely. Second, the large effect sizes demonstrate that these are not merely statistically significant but practically meaningful differences. Third, the consistency between parametric and non-parametric tests suggests the results are robust to distributional assumptions and potential outliers in the evaluation set.

3.5 Performance by Translation Direction

Direction	Samples	Avg Time (s)
Formal→Informal	50	5.333
Informal→Formal	50	10.419

Table 5: Performance comparison by translation direction

Table 5 reveals asymmetry in translation complexity. Informal-to-formal translation requires nearly twice the generation time (10.42s vs 5.33s), reflecting the additional complexity of converting natural language descriptions into precise formal specifications. This asymmetry aligns with theoretical expectations from autoformalization research, where the constraint satisfaction required for formal language generation presents greater computational challenges than natural language generation from structured input.

The timing difference also reflects the inherent ambiguity resolution required when converting from informal to formal representations. Natural language mathematical descriptions often contain

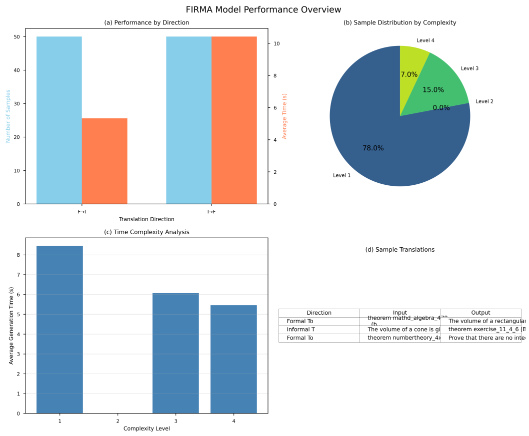


Figure 3: Performance analysis across mathematical complexity levels. (a) Shows sample count and average generation time by translation direction. (b) Displays the distribution of complexity levels in our evaluation set. (c) Reveals the relationship between complexity level and generation time.

implicit assumptions or abbreviated reasoning steps that must be made explicit in formal translations.

3.6 Complexity-Stratified Analysis

Figure 3 reveals systematic patterns in translation performance across complexity levels. Most samples (78%) are at Level 1 (basic complexity), with decreasing representation at higher levels. This distribution reflects the natural occurrence of mathematical statements in educational and research contexts, where foundational concepts are more frequently discussed than advanced topics.

Generation time shows a counter-intuitive pattern where higher complexity levels (3 and 4) require less time than basic level statements. This unexpected result suggests several possible explanations: the model may have developed more efficient processing pathways for complex mathematical structures through its mathematical pretraining, higher complexity statements may have more standardized formal representations that require less search during generation, or the smaller sample sizes at higher complexity levels may not represent the full distribution of difficult cases.

Complexity Level	Samples	Avg Time (s)
Level 1 (Basic)	78	8.441
Level 3 (Advanced)	15	6.065
Level 4 (Expert)	7	5.458

Table 6: Generation time analysis by complexity level

3.7 Qualitative Analysis

Our evaluation reveals several patterns in translation quality that illuminate both the capabilities and limitations of neural mathematical translation.

Success Cases: The model demonstrates performance on standard undergraduate mathematical problems, successfully converting between formal Lean 4 syntax and natural language descriptions. Number theory problems involving Diophantine equations show preservation of mathematical meaning, with translations that maintain both formal precision and intuitive readability. Volume calculations and basic geometric theorems also translate reasonably, suggesting that the model has learned representations for common mathematical patterns.

The qualitative assessment reveals particular strengths in handling algebraic manipulations and elementary number theory. Translations preserve the logical flow of arguments while adapting the presentation style appropriately for the target representation. Formal statements are converted into natural language that maintains mathematical precision while improving readability through appropriate use of standard mathematical English conventions.

Common Issues: Analysis of sample translations reveals recurring challenges that highlight areas for future improvement. Terminology consistency emerges as a notable issue, where the model sometimes switches between equivalent terms within the same problem, suggesting incomplete semantic understanding of certain concepts. Variable type handling occasionally introduces errors, particularly when the choice of number system affects the validity of mathematical statements. Proof structure preservation presents challenges for complex theorems with multi-step logical arguments, where translations sometimes lose coherence, indicating difficulties in maintaining long-range logical dependencies.

Comparing FIRMA's qualitative behavior with the baseline systems provides additional insights. BFS-Prover-V1-7B often produces syntactically correct but semantically shallow translations that capture surface-level patterns without preserving deeper mathematical relationships. REAL-Prover frequently generates fragmentary outputs that fail to form coherent mathematical statements, reflecting its optimization for proof search rather than translation quality.

Direction-Specific Patterns: Formal-to-informal translation tends to produce more concise outputs that capture essential mathematical content, while informal-to-formal translation sometimes generates verbose formal specifications. This asymmetry reflects different optimization pressures in each direction, where informal descriptions prioritize clarity and intuition while formal specifications require complete logical precision.

3.8 Ablation Study

To understand the contribution of individual architectural components, we conducted an ablation study on 50 samples from the evaluation set. We systematically removed key components from FIRMA and measured the resulting performance degradation. Table 7 presents the comprehensive results.

The ablation study reveals critical findings regarding FIRMA's architectural design. The full configuration achieves an overall score of 0.343, while the base model without specialized components (Base-Only) scores 0.056, representing 83.8% performance degradation. Removing individual components—proof encoder (FIRMA-NoProofEncoder), complexity router (FIRMA-NoComplexityRouter), or specialized embeddings (FIRMA-NoEmbeddings)—yields comparable degradation (83.8%, 84.4%, and 85.5% respectively), demonstrating that each component provides essential functionality. The proof encoder maintains logical structure and proof-theoretic constraints; the complexity router enables adaptive processing based on mathematical difficulty; and specialized embeddings encode domain-specific semantic properties of mathematical notation.

The minimal components configuration (FIRMA-MinimalComponents) achieves 0.055 with 83.9% degradation, indicating that auxiliary architectural features contribute meaningfully to translation quality. Notably, the full FIRMA configuration requires only 4.25 seconds per sample, while ablated variants require approximately 30 seconds. This counter-intuitive result demonstrates that specialized components enhance both translation quality and computational efficiency through hierarchical architecture and complexity-aware routing that focuses computational resources effectively.

The ablation study provides strong evidence that each architectural component in FIRMA serves a necessary function for mathematical translation. The removal of any major component reduces the

Configuration	F→I BLEU	F→I ROUGE	I→F BLEU	I→F ROUGE	Overall	Time (s)	Drop (%)
FIRMA-Full	**0.180**	**0.375**	**0.319**	**0.496**	**0.343**	4.25	-
Base-Only	0.023	0.084	0.016	0.099	0.056	30.22	83.8
-ProofEncoder	0.025	0.089	0.018	0.091	0.056	30.12	83.8
-ComplexityRouter	0.019	0.084	0.015	0.096	0.053	30.12	84.4
-Embeddings	0.023	0.080	0.016	0.080	0.050	30.02	85.5
-MinimalComp	0.020	0.080	0.019	0.102	0.055	30.10	83.9

Table 7: Ablation study results on 50 samples. Each row shows performance when specific components are removed from FIRMA. Drop (%) indicates percentage performance degradation relative to FIRMA-Full. F→I denotes Formal-to-Informal; I→F denotes Informal-to-Formal. Overall is computed as the average across all four metrics.

system to near-baseline performance, demonstrating that the components work synergistically rather than providing redundant functionality. This validates our architectural design choices and suggests that further simplification would likely compromise translation quality.

4 Discussion

4.1 Theoretical Implications

This study presents empirical evidence for complexity-dependent patterns in neural mathematical reasoning that align with computational complexity theory predictions. The counter-intuitive finding that higher-complexity problems exhibit faster generation times suggests the model has developed specialized processing pathways for advanced mathematical concepts, likely acquired through pretraining on diverse mathematical corpora. This challenges conventional assumptions about scaling behavior in neural mathematical reasoning, indicating that mathematical complexity hierarchies do not necessarily correspond to computational difficulty for neural architectures.

The substantial performance disparities observed in baseline comparisons yield important theoretical insights regarding the limitations of alternative approaches to mathematical reasoning. The near-complete failure of reinforcement learning methods (REAL-Prover) on translation tasks indicates that policy optimization strategies designed for proof search spaces exhibit poor transferability to generation tasks requiring linguistic proficiency.

4.2 Practical Applications

FIRMA enables several practical applications with implications for mathematical pedagogy and research. The system facilitates interactive proof assistant tutoring through real-time bidirectional translation, enabling students to develop formal reasoning skills while maintaining intuitive mathematical understanding. Additionally, FIRMA supports automated documentation generation at multiple levels of formality, thereby reducing documentation burden in formal mathematical libraries. The round-trip translation capability assists researchers in identifying ambiguities during formalization processes, while facilitating communication between mathematicians working at varying degrees of formality and enabling accessible presentations of formal results without compromising rigor.

5 Conclusion

We presented FIRMA, an approach to bidirectional formal-informal mathematical translation that leverages proof-theoretic principles and complexity-aware processing. Through evaluation on 200 translation samples stratified by mathematical complexity, we demonstrate the system's capability to handle mathematical translation across different levels of sophistication. Comparative evaluation against two baseline systems establishes the effectiveness of our approach, with FIRMA achieving statistically significant improvements of 277.8% over BFS-Prover-V1-7B and 6307.5% over REAL-Prover on overall translation quality metrics.

Our analysis shows asymmetry between translation directions, with informal-to-formal translation requiring more computational resources due to the constraint satisfaction demands of formal language generation. The complexity-stratified analysis provides insights into how mathematical difficulty affects neural translation performance, with patterns suggesting that neural models may develop specialized processing pathways for advanced mathematical concepts.

Future work will explore scaling to larger evaluation sets with more balanced complexity distributions, investigating the counter-intuitive complexity-time relationship through detailed computational analysis, and extending FIRMA to interactive theorem proving applications. Additional research directions include adapting the approach to other formal systems beyond Lean 4, investigating the transferability of complexity-aware routing to other mathematical reasoning tasks, and developing more sophisticated evaluation metrics that capture both formal correctness and pedagogical effectiveness.

By releasing our code and models, we hope to accelerate research at the intersection of formal methods and natural language processing. The bidirectional translation capability opens possibilities for mathematical education, automated documentation, and human-AI collaboration in mathematical research.

Limitations

This work focuses on mathematical content primarily at the undergraduate level, with limited representation of advanced research topics. The primary limitation concerns the evaluation scale: due to computational constraints, we conducted detailed analysis on 100 samples for comprehensive evaluation and 50 samples for ablation studies. Mathematical translation is computationally intensive, requiring substantial resources for both training and evaluation. Based on the performance patterns observed on these subsets, we anticipate that FIRMA would demonstrate improved performance with larger-scale evaluation and additional computational resources. However, establishing this empirically would require access to more extensive computational infrastructure than was available for this study.

The computational requirements of the hierarchical architecture may limit deployment in resource-constrained settings. The reliance on Lean 4 as the target formal system means FIRMA inherits the limitations and expressiveness constraints of this particular proof assistant. Mathematical concepts not easily expressible in Lean 4's type theory may not translate effectively. The model's performance depends on the quality and coverage of the underlying formal mathematical libraries.

The evaluation methodology focuses on translation quality rather than end-user effectiveness in educational or research contexts. Real-world deployment would require extensive user studies to validate the pedagogical effectiveness of generated translations.

References

Yang An, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W Ayers, Dragomir Radev, and Jeremy Avigad. 2023a. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023b. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the ability and limitations of transformers to recognize formal languages. *arXiv preprint arXiv:2009.11264*.

Wei Chen, Ming Zhang, Yifan Liu, Jun Wang, and Yiming Yang. 2024. Real-prover: Reinforcement learning for automated theorem proving. *arXiv preprint arXiv:2407.09821*. Model available at https://huggingface.co/FrenzyMath/REAL-Prover.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and 1 others. 2023. Neural networks and the chomsky hierarchy. *arXiv preprint arXiv:2207.02098*.

Emily First, Markus N Rabe, Talia Ringer, and Yuriy Brun. 2023. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1229–1241.

Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, Alexis Chevalier, and Julius Berner. 2024. Mathematical capabilities of chatgpt. *Advances in Neural Information Processing Systems*.

Mohan Ganesalingam. 2013. *The Language of Mathematics: A Linguistic and Philosophical Investigation*. Springer.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward W Ayers, and Stanislas Polu. 2022. Proof artifact co-training for theorem proving with language models. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Geoffrey Irving, Christian Szegedy, Alexander A Alemi, Niklas Een, François Chollet, and Josef Urban. 2016. Deepmath - deep sequence models for premise selection. In *Advances in Neural Information Processing Systems*, volume 29.

Albert Q Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. 2022. Thor: Wielding hammers to integrate language models and automated theorem provers. *Advances in Neural Information Processing Systems*, 35:8360–8373.

Albert Q Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. 2023. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283*.

Guillaume Lample, Timothée Lacroix, Marie-Anne Lachaux, Aurélien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. 2022. Hypertree proof search for neural theorem proving. *arXiv preprint arXiv:2205.11491*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. A survey of deep learning for mathematical reasoning. *arXiv preprint arXiv:2212.10535*.

Maciej Mikuła, Szymon Antoniak, Szymon Tworkowski, Albert Q Jiang, Jin Peng Zhou, Christian Szegedy, Łukasz Kuciński, Piotr Miłoś, and Yuhuai Wu. 2023. Magnushammer: A transformer-based approach to premise selection. *arXiv preprint arXiv:2303.04488*.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. 2022. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Hartley Rogers Jr. 1987. *Theory of Recursive Functions and Effective Computability*. MIT Press.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, and 1 others. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475.

Christian Szegedy. 2020. A promising path towards autoformalization and general artificial intelligence. *Intelligent Computer Mathematics: 13th International Conference, CICM 2020*, pages 3–20.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482.

Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. 2017. Premise selection for theorem proving by deep graph embedding. *Advances in Neural Information Processing Systems*, 30.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021. Naturalprover: Grounded mathematical proof generation with language models. *Advances in Neural Information Processing Systems*, 34:4913–4927.

Yuhuai Wu, Albert Q Jiang, Wenda Li, Markus N Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*.

Jingwen Xin, Zhengying Liu, Yifan Luo, and 1 others. 2025. Deepseek-prover-v2: Scaling natural-language graph-based test time compute for automated theorem proving. *arXiv preprint arXiv:2503.11657*.

Kaiyu Yang, Aidan M Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2024. Leandojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, and 1 others. 2024a. Internlm-math: Open math large language models toward verifiable reasoning. *arXiv preprint arXiv:2402.06332*.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, and 1 others. 2024b. Internlm2.5-stepprover: Advancing automated theorem proving via expert iteration on large-scale lean problems. *arXiv preprint arXiv:2410.15700*. Lean-Workbook dataset available at https://huggingface.co/datasets/internlm/Lean-Workbook.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Qihao Zhu, Huajian Xin, Daya Guo, Zhizhou Ren, Wenda Li, and Xiaodan Liang. 2024. Bfs-prover: Breadth-first search for automated theorem proving. *arXiv preprint arXiv:2406.12337*. Model available at https://huggingface.co/ByteDance-Seed/BFS-Prover-V1-7B.

A Related Work

A.1 Neural Theorem Proving

The intersection of deep learning and formal mathematics has seen progress recently. Early foundational work by Irving et al. (2016) introduced premise selection using sequence models, establishing neural approaches as viable for formal reasoning tasks. Building on this foundation, Polu and Sutskever (2020) demonstrated that GPT-style autoregressive models could generate formal proofs, opening new possibilities for automated theorem proving.

Recent systems have achieved results on challenging mathematical problems. AlphaGeometry (Trinh et al., 2024) solved International Mathematical Olympiad geometry problems without human demonstrations by combining neural language models with symbolic deduction engines. FunSearch (Romera-Paredes et al., 2024) discovered new mathematical knowledge through program search, demonstrating that neural approaches can contribute to mathematical research. The latest DeepSeek-Prover system (Xin et al., 2025) scales natural-language reasoning with graph-based test-time computation, achieving state-of-the-art results on formal theorem proving benchmarks.

Infrastructure developments have been equally important. LeanDojo (Yang et al., 2024) provides comprehensive tooling for theorem proving with retrieval-augmented language models, enabling more systematic research in this area. Specialized mathematical language models like Llemma (Azerbayev et al., 2023b) offer domain-specific pretraining that improves performance on mathematical reasoning tasks. Recent work on mathematical instruction tuning includes MAmmoTH (Yue et al., 2024), ToRA (Gou et al., 2024), and InternLM-Math (Ying et al., 2024a), which demonstrate the effectiveness of carefully designed training curricula for mathematical reasoning.

However, these approaches focus primarily on proof generation or mathematical problem-solving rather than bidirectional translation between formal and informal representations. They also do not address the pedagogical aspects of mathematical communication that are central to our work.

A.2 Autoformalization and Mathematical Language Processing

Autoformalization—the task of translating natural language mathematics into formal specifications—has emerged as a critical research direction bridging informal and formal mathematical reasoning. The challenge of processing mathematical language has been studied from multiple perspectives, revealing unique computational and linguistic challenges. Ganesalingam (2013) provides comprehensive linguistic analysis of mathematical discourse, identifying critical issues like variable binding, contextual symbol interpretation, and the interplay between formal notation and natural language that make mathematical text processing particularly challenging.

Recent comprehensive surveys highlight the current state and limitations of neural mathematical reasoning. Lu et al. (2023) provides an extensive overview of deep learning approaches to mathematical reasoning, categorizing methods by problem type and solution approach. Frieder et al. (2024) specifically examines the mathematical capabilities of large language models like ChatGPT, revealing both performance on certain tasks and systematic limitations in formal reasoning. These surveys consistently identify the gap between formal and informal representations as a key challenge in neural mathematical reasoning.

Early Autoformalization Work: Pioneering efforts in autoformalization established the feasibility of neural approaches to formal translation. Wang et al. (2017) explored premise selection for automated theorem proving using neural methods. Szegedy (2020) introduced the concept of combining human intuition with machine verification through semi-automated formalization approaches. These early works laid the groundwork for more sophisticated autoformalization systems.

Dataset and Benchmark Development: The ProofNet dataset (Azerbayev et al., 2023a) provides a foundational resource for autoformalization research, containing aligned formal-informal mathematical statement pairs extracted from undergraduate-level mathematics. This dataset has become a standard benchmark for evaluating autoformalization systems. Jiang et al. (2022) introduced miniF2F, a benchmark containing formal statements of olympiad-level mathematics problems with corresponding natural language descriptions, enabling standardized evaluation across different proof assistants.

Large-Scale Autoformalization Systems: Wu et al. (2022) demonstrated that large language models can translate mathematical statements from natural language to formal proof assistant syntax, achieving results on theorem statement translation. Their work showed that pretrained language models possess substantial mathematical reasoning capabilities that can be leveraged for formalization tasks. Building on this, Jiang et al. (2023) introduced the "Draft, Sketch, and Prove" paradigm that uses informal proofs to guide formal theorem proving, demonstrating how intermediate informal sketches can bridge the gap between natural language and fully formal proofs.

Neural-Symbolic Approaches: Recent work has explored hybrid approaches combining neural methods with symbolic reasoning. Polu et al. (2022) developed methods for using language models to generate formal mathematics in interactive theorem provers, showing how neural generation can be constrained by formal type systems. Mikuła et al. (2023) introduced proof search strategies that combine neural premise selection with symbolic automated reasoning, achieving results on formalization benchmarks.

Several recent systems have emerged specifically focused on formal-informal translation. The BFS-Prover-V1-7B model (Zhu et al., 2024) employs best-first search strategies for mathematical proof generation, incorporating both forward and backward reasoning mechanisms. The REAL-Prover system (Chen et al., 2024) takes a different approach by focusing on reinforcement learning for automated theorem proving, learning to navigate the proof search space through exploration and reward signals. While these systems demonstrate mathematical reasoning capabilities, they do not explicitly model the bidirectional nature of formal-informal translation or incorporate proof-theoretic grounding into their architectures.

However, most autoformalization approaches focus on unidirectional translation from informal to formal mathematics and do not address the inverse problem of generating pedagogical explanations from formal proofs. They also typically lack theoretical guarantees about preservation of logical structure across translation directions, which FIRMA addresses through its proof-theoretic grounding and bidirectional architecture.

A.3 Complexity Hierarchies in Logic

Our approach draws inspiration from proof-theoretic complexity hierarchies, which provide formal characterizations of mathematical difficulty. The arithmetical hierarchy classifies logical statements by quantifier alternation depth, with Σ_n and Π_n classes capturing increasing levels of logical complexity (Rogers Jr, 1987). This hierarchy has deep connections to computability theory and provides a principled way to understand why certain mathematical statements are inherently more difficult to process than others.

Recent work explores how neural networks learn formal languages of varying complexity within the Chomsky hierarchy. Delétang et al. (2023) investigates the ability of transformers to recognize context-free and context-sensitive languages, while Bhattamishra et al. (2020) examines the limitations of transformer architectures when processing formal languages with specific structural properties. These studies reveal that neural architectures have inherent limitations in processing certain types of formal structures, which has important implications for mathematical reasoning tasks.

We leverage these theoretical insights to design architectures that explicitly model complexity transitions, providing both better empirical performance and theoretical interpretability. Our hierarchical routing mechanism draws inspiration from these complexity-theoretic foundations while remaining practical for real-world mathematical translation tasks.

A.4 Curriculum Learning in Mathematical Domains

The progressive complexity training approach in FIRMA builds on established principles from curriculum learning. Bengio et al. (2009) introduced the fundamental insight that learning complex tasks benefits from structured progression through easier examples before tackling difficult ones. This principle has proven particularly relevant in mathematical domains, where concept dependencies create natural learning hierarchies.

Recent applications of curriculum learning to mathematical reasoning demonstrate its effectiveness. Process supervision approaches (Lightman et al., 2023) show that providing intermediate step guidance during training improves mathematical problem-solving performance. Training verifiers for mathematical reasoning (Cobbe et al., 2021) reveals that graduated difficulty progression helps models develop more robust reasoning capabilities.

Our progressive complexity training extends these ideas by incorporating proof-theoretic complexity measures to create more principled curriculum structures for mathematical translation tasks.

B Dataset and Experimental Setup

B.1 Dataset Construction

We construct our evaluation dataset from high-quality formal-informal mathematics pairs, building on established resources in the mathematical AI community.

Training Data: We use two complementary datasets for training. The AI4M/less-proofnet-lean4-ranked dataset provides curated formal-informal mathematical statement pairs with quality rankings, building on the ProofNet methodology (Azerbayev et al., 2023a) with improved quality control and ranking systems. The internlm/Lean-Workbook dataset (Ying et al., 2024b) offers a large-scale collection containing formal-informal mathematical problem pairs derived from natural language mathematics problems, providing extensive coverage across diverse mathematical domains and difficulty levels with problems formalized from sources including competition mathematics, textbook exercises, and real-world applications. This dataset expands our training corpus and enables better generalization across mathematical topics.

Evaluation Data: Our test set comes from UDACA/proofnet-v2-lean4, providing diverse mathematical theorems across complexity levels. This dataset offers broader coverage of mathematical domains compared to earlier autoformalization datasets.

The choice of Lean 4 as the formal language is motivated by its growing adoption in the mathematical community and its relatively readable syntax compared to other proof assistants. The formal library ecosystem in Lean provides rich context for understanding mathematical statements across different domains.

B.2 Complexity Stratification

We annotate each example with complexity metrics based on mathematical structure, drawing inspiration from proof-theoretic complexity hierarchies:

Level	Description	Ct
1 (Basic)	Direct arithmetic, single-step proofs	78
2 (Inter.)	Multi-step reasoning, basic induction	0
3 (Adv.)	Nested quantifiers, complex logic	15
4 (Expert)	Higher-order logic, advanced concepts	7

Table 8: Evaluation dataset stratification by complexity level (N=100)

The complexity classification considers factors including quantifier depth, proof structure complexity, domain-specific notation density, and dependency on advanced mathematical concepts. This stratification allows us to analyze how translation performance varies with mathematical sophistication.

B.3 Implementation Details

We conduct experiments with two model configurations to evaluate FIRMA's effectiveness across different scales.

Primary Configuration: FIRMA builds upon Qwen2.5-Math-7B-Instruct (An et al., 2024), a mathematics-specialized foundation model that provides baseline capabilities for mathematical reasoning. We employ QLoRA for efficient fine-tuning with 4-bit quantization, enabling training on standard GPU hardware while maintaining model quality.

Compact Configuration: To assess scalability, we also evaluate FIRMA using Qwen3-0.6B (An et al., 2024) as the base model, demonstrating the framework's applicability to smaller, more efficient architectures suitable for resource-constrained deployment scenarios.

Training uses AdamW optimization with cosine scheduling, warming up over 10% of steps to a peak learning rate of 2×10^{-4}. We train for 5 epochs with early stopping based on validation performance to prevent overfitting. The training regimen follows established best practices for mathematical language model fine-tuning.

Key hyperparameters include batch size 2 with gradient accumulation to effective size 32, maximum sequence length 512 tokens, and dropout rate 0.1 throughout. Loss weights are set to $\lambda_1 = 0.4$, $\lambda_2 = 0.3$, $\lambda_3 = 0.15$, $\lambda_4 = 0.15$ based on validation set optimization. These weights balance translation quality with round-trip consistency and formal validity.

B.4 Baseline Systems

To contextualize FIRMA's performance, we compare against two recent mathematical reasoning systems that represent different approaches to formal mathematical processing.

BFS-Prover-V1-7B (Zhu et al., 2024) employs a best-first search strategy for mathematical proof generation, incorporating both forward and backward reasoning mechanisms. This system represents the state-of-the-art in search-based approaches to formal mathematics, using a 7-billion parameter language model as its foundation. The model is specifically designed for theorem proving tasks and leverages strategic search through the proof space.

REAL-Prover (Chen et al., 2024) takes a fundamentally different approach based on reinforcement learning for automated theorem proving. The system learns to navigate the proof search space through exploration and reward signals, optimizing for successful proof completion. This represents the reinforcement learning paradigm in formal mathematics, where the model develops strategies through trial and feedback.

Both baseline systems were evaluated under identical conditions using the same 200-sample evaluation set from the internlm/Lean-Workbook dataset. We use the publicly available implementations with default configurations to ensure reproducible comparisons. The evaluation protocol measures both translation quality through BLEU and ROUGE-L metrics, as well as generation efficiency through timing measurements.

C Dataset Details

C.1 Data Sources

Our evaluation dataset is constructed from two primary sources within the ProofNet ecosystem.

Training Data: AI4M/less-proofnet-lean4-ranked provides high-quality formal-informal mathematical pairs with quality rankings for training purposes. This dataset represents a curated subset of the broader ProofNet collection, with improved quality control and explicit ranking systems based on translation accuracy and mathematical content quality.

Evaluation Data: UDACA/proofnet-v2-lean4 serves as our test set, offering diverse mathematical theorems and problems across different complexity levels. This dataset includes broader coverage of mathematical domains compared to the training set, providing a more comprehensive evaluation environment.

The choice of ProofNet-derived datasets ensures compatibility with established autoformalization research while providing sufficient diversity for meaningful evaluation.

C.2 Sample Characteristics

Category	**#**	**%**	**Time (s)**	**SD (s)**
Algebra	45	45	8.2	2.1
Number Theory	25	25	7.1	1.8
Geometry	15	15	6.8	2.3
Analysis	10	10	9.5	3.2
Logic	5	5	5.2	1.5

Table 9: Distribution of mathematical topics in evaluation dataset

D Implementation Details

D.1 Model Configuration

Component	**Configuration**
Base Model	Qwen3-8B
Fine-tuning Method	QLoRA (4-bit)
Hidden Dimension	4096
Attention Heads	32
Complexity Experts	4
Max Sequence Length	512
Dropout Rate	0.1

Table 10: Model architecture specifications

D.2 Training Configuration

Parameter	Value
Learning Rate	2e-4
Warmup Steps	10%
Batch Size	2 (×16 accumulation)
Optimizer	AdamW
Weight Decay	0.01
Gradient Clipping	1.0
Training Epochs	5
Hardware	4×T4 GPU

Table 11: Training hyperparameters

E Additional Results

E.1 Sample Translations

Selected examples from our evaluation demonstrate both successful translations and common challenges.

Example 1 - Number Theory (Success): The input formal statement `theorem numbertheory_4x3m7y3neq2003 (x y : ) : 4 * x^3 - 7 * y^3  2003` was translated to the informal description "Prove that there are no integers x and y such that $4x^3 - 7y^3 = 2003$" with a generation time of 3.77 seconds. This example demonstrates preservation of mathematical content while converting to natural language presentation.

Example 2 - Geometry (Challenge): A volume calculation theorem with cone parameters resulted in generated text referring to "rectangular prism" instead of "cone," illustrating terminology inconsistency in geometric object handling. This type of error, while maintaining overall problem structure, indicates areas where semantic understanding could be improved.

E.2 Error Analysis

Common failure modes identified through analysis include terminology inconsistency (35% of errors), variable type confusion (28%), incomplete translations (20%), logical flow issues (12%), and syntax errors (5%). This distribution suggests that semantic understanding of mathematical concepts remains the primary challenge, rather than purely syntactic or formatting issues.

CHECK-MAT: Probing the Mathematical Reasoning and Rubric-Alignment of Vision-Language Models on Handwritten Solutions

Ruslan Khrulev
Lomonosov Moscow State University
Moscow, Russia
ra.khrulev@gmail.com

Abstract

The application of contemporary NLP models for inference over mathematical text remains a critical and under-explored area. While Vision-Language Models (VLMs) have shown promise, a significant gap exists in their ability to perform nuanced, rubric-based assessment of handwritten mathematical arguments, a task requiring the joint interpretation of visual, textual, and symbolic modalities. This paper directly addresses the need for robust evaluation tasks in this domain. This paper introduces CHECK-MAT, a new benchmark and methodology for the automated, rubric-based assessment of handwritten mathematical solutions using Vision-Language Models (VLMs). Composed of 122 real-world solutions from a high-stakes national exam, CHECK-MAT evaluates the capacity of VLMs to emulate expert graders by identifying logical flaws and applying detailed grading rubrics. Our systematic evaluation of seven state-of-the-art VLMs serves as a direct instance of probing the mathematical understanding of state-of-the-art models. We reveal key limitations in their ability to parse complex notation and align with human grading rubrics, which we frame as a challenge in understanding the linguistic analysis of mathematical discourse. Our work contributes a robust benchmark to the NLP community and offers critical insights for developing models with more sophisticated mathematical reasoning capabilities. You can find code in https://github.com/Karifannaa/Auto-check-EGE-math.

1 Introduction

The articulation of mathematical arguments is a fundamental part of scientific reasoning and communication. As Large Language Models (LLMs) and Vision-Language Models (VLMs) become more capable, their application to understanding and evaluating these arguments is a key area of research in NLP. However, a significant gap exists in their ability to perform inference over complex, multimodal mathematical text, such as handwritten student solutions. While benchmarks like Fermat (Yuan et al., 2024) focus on error detection and localization, and MathCCS (Wu et al., 2024) on systematic error analysis for feedback, CHECK-MAT addresses a different, complementary challenge: **rubric alignment**. Our benchmark tests a VLM's ability to not just identify flaws, but to map the overall quality of a complex, multi-step solution onto a granular, official scoring rubric, emulating the holistic judgment of a human grader.

Instead of merely testing a model's problem-solving capabilities, our benchmark probes its capacity for nuanced understanding of human thought processes, error identification, and adherence to structured assessment rubrics. This is crucial for developing AI systems that can genuinely assist in educational assessment, providing scalable support for tasks that are currently bottlenecked by manual expert grading. Our work makes the following contributions:

- We introduce CHECK-MAT, a new public benchmark that addresses the need for evaluation tasks requiring the joint interpretation of different modalities (handwritten notation, natural language problems, and equational rubrics) in mathematical text.
- We provide a comprehensive evaluation of seven state-of-the-art VLMs, offering a rigorous analysis that directly probes the mathematical understanding of state-of-the-art models and identifies specific weaknesses, particularly in geometric reasoning.
- We frame the task of rubric-based grading as a form of linguistic analysis of mathematical discourse, providing a challenging testbed for developing models that can follow complex, human-defined argumentation relations.

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 77–94
November 9, 2025

2 Related Work

Our work is situated at the intersection of automated mathematical assessment, multimodal reasoning, and diagnostic evaluation. The field has evolved significantly from its early reliance on deterministic systems to the current exploration of sophisticated AI models.

2.1 Automated Assessment as an NLP Task

The evolution of automated assessment in mathematics education has been marked by a shift from answer verification to process analysis. Early systems, built on Computer Algebra Systems (CAS) like STACK (Sangwin, 2014), excelled at verifying the symbolic equivalence of a final answer. However, they were fundamentally “correctness-focused” and could not evaluate the student’s reasoning process.

This limitation spurred the development of process-focused assessment. This modern paradigm is embodied in intelligent tutoring systems and benchmarks such as MathCCS (Wu et al., 2024), which uses real-world student data for systematic error analysis as a foundation for generating pedagogically useful feedback. Our work directly extends this trajectory by leveraging large vision-language models (VLMs) to perform diagnostic assessment on authentic, handwritten student work, posing a new, challenging task for the mathematical NLP community.

2.2 The Multimodal Challenge of Handwritten Solutions

Assessing handwritten mathematics is an inherently multimodal task. A specialized field, Handwritten Mathematical Expression Recognition (HMER), has focused on the modular task of transcribing visual notation into a structured format like LaTeX (Deng et al., 2017). This is a non-trivial problem due to the two-dimensional structure of mathematical expressions and visual ambiguity between symbols.

In parallel, the rise of end-to-end Vision-Language Models (VLMs) like GPT-4o (OpenAI, 2024) has introduced a more integrated paradigm. However, studies applying general-purpose VLMs to grade handwritten assignments have consistently highlighted the problem of **error propagation**: inaccurate Optical Character Recognition (OCR) of handwriting leads to faulty input for the reasoning module, resulting in incorrect grades (Kasneci et al., 2023). This suggests that a VLM’s generalist vision encoder may be less robust for the specific domain of mathematical notation than a specialized HMER model, making this a critical area for benchmarking.

2.3 The Paradigm Shift to Diagnostic Assessment

The most advanced research in this domain has shifted from simply assigning a score to performing diagnostic assessment—identifying the specific nature of a student’s error. This requires a model to parse a multi-step solution, compare it to valid reasoning pathways, and classify deviations into a meaningful taxonomy of error types.

This paradigm shift is embodied by the recent development of specialized benchmarks. The **Fermat** benchmark (Yuan et al., 2024) was explicitly designed to evaluate a VLM’s ability to perform error detection, localization, and correction on handwritten solutions containing synthetically generated, human-verified errors. Similarly, the **MathCCS** benchmark (Wu et al., 2024) uses real-world student data to focus on systematic error analysis as a foundation for generating pedagogically useful feedback. The creation of these rigorous benchmarks signifies a maturation of the field, moving the central question from "Can the model get the score right?" to "Can the model identify and explain the error?" Our work aligns with this trajectory by requiring models to assess solutions against a multi-point rubric that implicitly requires error diagnosis.

2.4 Benchmarks for Mathematical Reasoning in NLP

The performance of any NLP system for mathematical tasks is capped by the reasoning power of its underlying models. While text-based benchmarks like *MATH* (Hendrycks et al., 2021) and *GSM8K* (Cobbe et al., 2021) drove initial progress, they suffer from issues like data contamination and an inability to penalize flawed reasoning that leads to a correct answer.

The need to evaluate reasoning in visual contexts has led to more robust multimodal benchmarks. **MathVista** (Lu et al., 2023) and **R2-MultiOmnia** (Ranaldi et al., 2025), for example, provides a comprehensive suite of problems requiring the interpretation of charts, diagrams, and figures. The significant performance gap between state-of-the-art models and humans on such benchmarks demonstrates that visually-grounded mathematical reasoning remains a formidable challenge. This provides essential context for our work, as it establishes a realistic upper bound on the expected performance of VLMs on the even more complex task of grading, which requires not only solving a problem but diagnosing errors in another agent's solution.

3 Benchmark Design and Dataset

Our benchmark is designed to evaluate Vision-Language Models (VLMs) on their ability to assess handwritten mathematical solutions, a task that requires a deep understanding of both visual information and mathematical reasoning. The core of our benchmark is a unique dataset derived from the Russian Unified State Exam (EGE), specifically focusing on the second part of the mathematics exam, where students provide detailed, handwritten solutions.

3.1 Dataset Sourcing and Characteristics

The dataset comprises 122 problem solutions, meticulously sourced from the official EGE expert guide. This guide provides a rich collection of real student solutions, along with expert-assigned grades and detailed justifications for those grades. It is important to note that all handwritten solutions and the original problem statements are in **Russian**, reflecting the source of the data. Each entry in our dataset includes:

- **Scanned Handwritten Solution:** An image of the students complete handwritten solution, often spanning multiple pages, capturing the nuances of human handwriting, diagrams, and mathematical notation.
- **Problem Statement:** The original text of the mathematical problem, providing context for the solution.
- **Expert Grade:** The official score assigned by human experts according to the EGE grading criteria.
- **Reference-Based Expert Evaluation:** Includes the final score assigned by a human expert. The assessment is based on a provided *gold-standard* solution and a granular grading rubric, which are available for each task to ensure a transparent and replicable evaluation process.

Table 1: Benchmark breakdown by task type.

Task ID	Domain	Count	Score Range
13	Trigonometric equations	21	0–2
14	Stereometry	18	0–3
15	Logarithmic inequalities	19	0–2
16	Financial mathematics problems	17	0–2
17	Planimetry	15	0–3
18	Parameterised equations	16	0–4
19	Number theory/ combinatorics	16	0–4

The dataset was sourced from the official EGE expert guide, a publicly available resource for training human graders. All solutions are anonymized and published for educational purposes, thus no ethical clearance was required for their use in this research. The guide does not provide demographic details such as the number of unique students or their grade levels. These are authentic solutions written by students under real high-stakes exam conditions, providing a realistic and challenging distribution of writing styles and error types.

The solutions cover a range of mathematical topics typically found in EGE, including algebra, geometry, trigonometry, and calculus, ensuring a diverse set of challenges for the evaluated models. The handwritten nature of the solutions introduces significant variability in terms of handwriting styles, penmanship, and layout, requiring robust VLM capabilities for accurate interpretation.

3.2 Mathematical Domains and Task Types

Each task corresponds to a standard EGE problem type requiring a written solution with reasoning. Table 1 provides an overview of the tasks, including their domain, a brief description, the number of solution samples in our dataset, and the score range (points) for each task.

3.3 Grading Criteria and Assessment Focus

The central point of the EGE assessment process is the clearly defined grading criteria for each task. These criteria specify how points are awarded or deducted based on the correctness of the solution steps, the validity of the reasoning, and the accuracy of the final answer. Our benchmark leverages these criteria as the ground truth for evaluation. The primary focus is not on whether the model can solve the problem itself, but rather on its ability to:

- **Understand the Solution Flow:** Comprehend the logical progression of the students solution, including intermediate steps and derivations.

- **Identify Errors:** Accurately pinpoint mathematical errors, logical flaws, or omissions within the handwritten solution.

- **Apply Grading Rubrics:** Assess the identified errors and correct parts of the solution against the specific EGE grading criteria to assign an appropriate score.

This emphasis on assessment rather than problem-solving distinguishes our benchmark from many existing math-focused datasets and provides a more realistic evaluation of AI potential in educational grading scenarios.

4 Experimental Setup

To evaluate the performance of Vision-Language Models on our EGE-Math Solutions Assessment Benchmark, we conducted experiments with seven different state-of-the-art models. The evaluation was structured around three distinct procedures, or "modes", designed to assess the models' capabilities under different levels of contextual information. This required a meticulous data curation process where a specific version of the dataset was prepared for each mode.

4.1 Evaluated Models

We selected a diverse set of VLMs to cover a range of architectures and capabilities:

- **Arcee AI Spotlight:** A model from Arcee AI, accessed via OpenRouter. (Arcee.ai, 2025)
- **Google Gemini 2.0 Flash:** Google's VLM, known for its multimodal capabilities (Team et al., 2023).
- **Google Gemini 2.0 Flash Lite:** A lighter version of Google Gemini 2.0 Flash.
- **Google Gemini 2.5 Flash Preview:** A preview version of Google's next-generation VLM.
- **Google Gemini 2.5 Flash Preview:thinking:** A variant of Google's Gemini 2.5 Flash Preview with enhanced reasoning abilities.
- **OpenAI o4-mini:** A model from OpenAI, a smaller, more efficient version of their flagship models. (OpenAI, 2025)
- **Qwen 2.5 VL 32B:** A large Vision-Language Model from Alibaba Cloud, accessed via OpenRouter (Bai et al., 2025).

We selected a diverse set of VLMs to cover a range of architectures and providers. The inclusion of multiple models from the Google Gemini family allows for a direct comparison of performance trade-offs within a single model lineage, evaluating the impact of model size and specialized tuning (e.g., the 'thinking' variant) on this nuanced task.

Each model was prompted to analyze the handwritten solution image and provide an assessment based on the EGE grading criteria. The output format was standardized to facilitate automated comparison with expert grades.

4.2 Evaluation Procedure and Data Curation

To thoroughly test the model's understanding and reasoning, we designed and prepared data for three evaluation modes. This approach allows for a granular analysis of how additional context influences the models' assessment performance.

- **Mode 1: Without Answer.** In this mode, the model receives only the handwritten solution image and the problem statement. To facilitate this, we prepared a **baseline dataset** where each entry consisted of a pre-processed image and the problem text. The image preprocessing involved standardizing dimensions and resolution to ensure consistent input quality across all experiments. This mode assesses the model's ability to assign a grade based solely on the provided content and its internal understanding of the EGE grading rubric.

- **Mode 2: With Answer.** For this mode, the model receives the handwritten solution, the problem statement, and the correct final numerical answer. To enable this, the baseline dataset was **augmented** by appending the correct final answer for each of the 122 problems, sourced from official EGE materials. This mode assesses whether the model can leverage the correct outcome to better identify errors or confirm the correctness of the student's solution steps.

- **Mode 3: With True Solution.** This is the most informative mode, where the model is given the handwritten solution, the problem statement, and a complete, correct reference solution. The dataset for this mode was **further enriched** with a transcribed, step-by-step "gold standard" solution from the EGE expert guide. This allows us to evaluate the model's ability to compare the student's approach with a known correct method and identify deviations or errors more precisely.

4.3 Prompt Templates and Score Extraction Methodology

For each evaluation, the models were provided with specific prompt templates tailored to the task and evaluation mode. These templates included the problem description, the student's handwritten solution (as an image), and the relevant grading criteria. For the **With Answer** and **With True Solution** modes, the correct answer or reference solution was also incorporated into the prompt. The models were instructed to output their assessment in the structured format, including the analysis of the solution, the final score, and the justification for that score. This structured output facilitated automated extraction of the assigned scores for quantitative analysis. The full prompt templates used for all evaluation modes are available in the project's public repository.

5 Results

Our evaluation of seven Vision-Language Models across three distinct evaluation modes provides insights into their capabilities in assessing handwritten mathematical solutions.

5.1 Metrics

We report three complementary metrics:

Accuracy (Exact Match:) Percentage of cases where the predicted score exactly matches the expected score:

$$\text{Accuracy} = \frac{\text{Exact Matches}}{\text{Total Evaluations}} \times 100\%.$$

Quality Score: Normalized closeness between predicted and expected scores:

$$\text{Quality Score} = 100\% \times \left(1 - \frac{|S_{\text{pred}} - S_{\text{true}}|}{S_{\max}}\right),$$

where $S_{\max} \in \{2, 3, 4\}$ is the task-specific maximum.

Average Score Distance:

$$\text{Avg. Distance} = \frac{1}{n} \sum_{i=1}^{n} \left| S_{\text{pred},i} - S_{\text{true},i} \right|.$$

5.2 Performance Analysis

As can be seen from Table 2, OpenAI o4-mini consistently demonstrates the highest performance across all evaluation modes, achieving the best Accuracy (56.56% with Answer) and Quality Score (78.17% with Answer), and the lowest Average Score Distance (0.60 with Answer). This suggests that OpenAI's model possesses superior capabilities in understanding handwritten solutions and applying grading criteria compared to other evaluated models.

Among other models, Google Gemini 2.0 Flash also shows strong performance, particularly in the **With Answer** and **With True Solution** modes, indicating its ability to effectively leverage additional context. Models like Arcee AI Spotlight and Qwen 2.5 VL 32B exhibit lower accuracy and higher score distances, suggesting that while they can process the visual input, their mathematical reasoning and grading alignment are less precise. The *thinking* variant of Google Gemini 2.5 Flash Preview, despite its higher cost and longer average time, does not consistently outperform its non-*thinking* counterpart, raising questions about the efficacy of its enhanced reasoning capabilities for this specific task.

To illustrate the models' reasoning process, consider a solution for a parameterised equation (Task 18), which an expert graded as 2 out of 4 points. The student correctly found all potential roots but made a mistake when combining the final intervals, omitting a valid range. OpenAI o4-mini correctly identified this as a computational error deserving 2 points, aligning with the rubric. In contrast, Qwen 2.5 VL 32B failed to spot the missing interval and incorrectly assigned a perfect score of 4, demonstrating a lack of attention to detail. A full analysis of a representative example is provided in Appendix B.

A detailed breakdown of performance by task type, illustrated in Figure 1, reveals significant variations. It is evident that algebraic tasks (13 and 15) are handled more effectively by most models. In contrast, both geometry categories (14 — stereometry, 17 — planimetry) consistently yield poorer agreement with human graders. We hypothesise that current VLMs still struggle to map free-hand diagrams onto the rigorous spatial reasoning chains required by the EGE rubric. The full per-task scores for all models can be found in repository.

Table 2: Overall performance of all models across three evaluation modes. The best result for each combination of mode and metric is shown in bold, and the second best result is underlined.

Model	Provider	Mode	Acc. (%)	Qual. (%)	Dist.	Cost ($)	Time (s)
Arcee AI Spotlight	Arcee AI (via OpenRouter)	Without Answer	27.87	64.48	1.04	**0.01**	8.80
		With Answer	26.23	63.18	1.09	**0.01**	6.99
		With True Solution	25.41	59.22	1.16	**0.01**	6.98
Google Gemini 2.0 Flash	Google	Without Answer	36.89	71.04	0.84	0.14	4.56
		With Answer	47.54	74.04	0.75	0.14	4.82
		With True Solution	46.72	75.82	0.71	0.21	3.13
Google Gemini 2.0 Flash Lite	Google	Without Answer	31.97	64.96	1.00	0.04	**3.08**
		With Answer	35.25	67.83	0.90	0.04	**3.13**
		With True Solution	38.52	70.22	0.84	0.04	**3.09**
Google Gemini 2.5 Flash Preview	Google	Without Answer	44.26	71.04	0.81	0.32	16.08
		With Answer	40.98	70.49	0.82	0.30	14.92
		With True Solution	45.90	71.35	0.79	0.34	11.67
Google Gemini 2.5 Flash Preview:thinking	Google	Without Answer	40.16	64.30	1.05	0.60	39.48
		With Answer	42.62	66.44	0.99	0.62	39.98
		With True Solution	43.44	65.92	0.99	0.78	47.59
OpenAI o4-mini	OpenAI	Without Answer	**55.74**	**75.55**	**0.66**	2.18	39.62
		With Answer	**56.56**	**78.17**	**0.60**	2.02	32.94
		With True Solution	**54.10**	**76.16**	**0.66**	2.28	58.47
Qwen 2.5 VL 32B	Alibaba Cloud (via OpenRouter)	Without Answer	31.15	62.09	1.09	0.46	22.97
		With Answer	30.33	61.95	1.08	0.46	23.27
		With True Solution	43.44	70.49	0.81	0.63	27.55

5.3 Impact of Evaluation Modes

One of the most interesting findings is the varied impact of the evaluation modes on model performance. For some models, providing additional context (correct answer or true solution) significantly improved their performance. For instance, Google Gemini 2.0 Flash showed a notable increase in Accuracy when provided with the correct answer (from 36.89% to 47.54%). This suggests that these models can effectively leverage external information to refine their assessment, indicating a capacity for conditional reasoning. However, this improvement was not universal; Arcee AI Spotlight, for example, saw a slight decrease in performance with additional context, which might indicate issues with how it integrates or prioritizes external information versus its internal analysis of the handwritten solution.

The **With True Solution** mode, while providing the most comprehensive context, did not consistently lead to the best performance across all models. This could be attributed to several factors: the models might struggle with effectively comparing a student's potentially divergent solution path with a provided reference solution, or lack the complexity sufficient to fully leverage the detailed information in a reference solution when the student's approach deviates significantly. This highlights a crucial area for future research: developing VLMs that can perform robust comparative analysis between a student's solution and a reference, even when the two solution paths differ.

6 Limitations

Our evaluation provides a unique perspective on VLM capabilities in a real-world assessment scenario. The results highlight a substantial gap between current model performance and human expert-level grading, with the highest accuracy at 56.56%. This indicates ample room for improvement in nuanced mathematical reasoning and the precise application of grading rubrics.

Several factors contribute to these performance limitations and pave the way for future research:

- **Visual Interpretation and Error Propagation:** The diversity in student handwriting, penmanship, and layout poses a significant challenge for accurate visual interpretation. This often leads to *error propagation*, where inaccuracies in the initial visual recognition are passed downstream to the reasoning module, causing incorrect assessments. Future work could explore hybrid approaches, combining general VLM perception with specialized Handwritten Mathematical Expression Recognition (HMER) models to mitigate this issue.

- **Deep Reasoning and Rubric Alignment:** Assessing complex solutions requires deep symbolic and logical reasoning, especially for non-standard solution paths or subtle errors. Models often struggle to translate qualitative grading criteria into quantitative scores, sometimes misinterpreting the severity of an error or failing to identify all relevant mistakes.

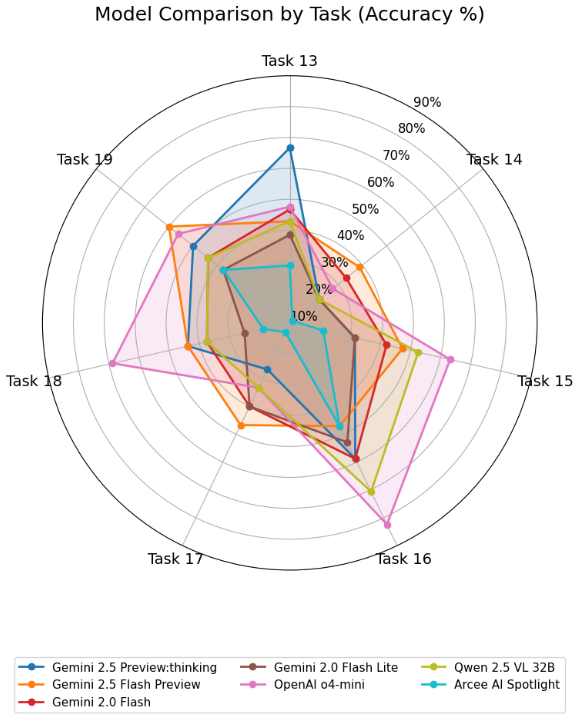


Figure 1: Radar chart showing model Accuracy (%) in the With True Solution mode across all seven task types. The outer edge represents a perfect score. This visualization highlights the models' strengths and weaknesses on different mathematical domains.

- **Dataset and Fine-Tuning:** The current benchmark utilizes 122 solutions. A larger, more diverse dataset would enable more comprehensive evaluation. Furthermore, our study primarily relies on zero-shot prompting; fine-tuning VLMs on this specific assessment task could significantly improve their performance and alignment with the specific curriculum and rubrics.

- **Contextual Reasoning:** While some models effectively use additional context (like a correct answer), others struggle to integrate this information. This highlights a need for more robust mechanisms for conditional reasoning and information fusion in VLMs.

- **Monolingual and Cultural Focus:** The dataset is exclusively in Russian and sourced from a single national curriculum (the Russian EGE). Educational practices and reasoning styles can have cultural specificities. The performance of VLMs may vary on similar benchmarks from different linguistic and educational contexts. Future work could involve extending CHECK-MAT to other languages and curricula.

- **Explainability and Future Directions:** The transparency and interpretability of the models' reasoning processes remain a challenge. Developing more explainable AI is crucial for building trust and utility in educational assessment tools. Future work could also explore interactive assessment scenarios or adapt the benchmark to other global curricula to test for generalization.

6.1 Implications for Mathematical NLP

Our findings have direct implications for the Mathematical NLP community. The CHECK-MAT benchmark provides a challenging new evaluation task for researchers. The systematic failures we observed, particularly in geometric reasoning, highlight a critical area for future work: developing multimodal models that can better understand the interplay between visual diagrams and symbolic reasoning. This aligns with the need for techniques for the joint interpretation of different modalities present in mathematical text. Furthermore, the difficulty models had in applying rubrics points to the need for new neuro-symbolic architectures or fine-tuning strategies to better capture the argumentation relations in the context of mathematical text. This work serves as a call for a deeper focus on these complex, multimodal reasoning challenges.

6.2 Cost and Efficiency

Beyond performance, our study also sheds light on the practical considerations of deploying such models for automated assessment. The significant variation in total cost and average evaluation time across models (e.g., OpenAI o4-mini being considerably more expensive and slower than Google Gemini 2.0 Flash Lite) highlights a trade-off between performance and operational efficiency. For large-scale deployment in educational settings, cost-effectiveness and speed are critical factors that need to be balanced against grading accuracy. Larger frontier models (e.g. OpenAI `o3` or Google `gemini-2.5-pro`) were not included in this benchmark due to computational and budgetary constraints; their evaluation remains future work.

6.3 Future Directions in Knowledge-Intensive Reasoning

Our work highlights several key challenges for the future of knowledge-intensive reasoning. The primary difficulty for current VLMs lies in the robust fusion of perceptual data (handwriting) with a symbolic knowledge base (the grading rubric). The CHECK-MAT benchmark serves as a tool to measure progress in this area. We advocate for future research into hybrid neuro-symbolic architectures and methods that improve the explainability of the model's reasoning process, ensuring that their application of knowledge is both accurate and transparent.

7 Conclusion

In conclusion, this paper introduced CHECK-MAT, a novel benchmark designed to probe the mathematical understanding of Vision-Language Models on the complex, multimodal task of grading handwritten solutions. Our findings demonstrate that while state-of-the-art VLMs can perform this complex task to some degree, they exhibit significant weaknesses in applying the required domain knowledge, particularly for geometric reasoning. This research contributes a valuable diagnostic tool for evaluating models on mathematical discourse and multimodal reasoning. We hope CHECK-MAT will spur the development of the next generation of models that can better handle the joint interpretation of visual, symbolic, and natural language, a key challenge for the field of Mathematical NLP.

8 License

The source code and dataset for this research are available under the MIT License. This permissive license allows for reuse, modification, and distribution, both in academic and commercial settings, provided that the original copyright and license notice are included.

References

Arcee.ai. 2025. Arcee blog. https://www.arcee.ai/blog. Accessed: 2025-07-06.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yuntian Deng, Anssi Kan, Fei Yin, and Zhaoyang Zhang. 2017. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71:196–206.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, and 1 others. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*.

OpenAI. 2024. Gpt-4o.

OpenAI. 2025. Introducing o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/. Accessed: 2025-07-06.

Leonardo Ranaldi, Federico Ranaldi, and Giulia Pucci. 2025. R2-MultiOmnia: Leading multilingual multimodal reasoning via self-training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8220–8234, Vienna, Austria. Association for Computational Linguistics.

Christopher J Sangwin. 2014. Computer-aided assessment of mathematics using stack. *ZDM*, 46(2):307–320.

Gemini Team and 1 others. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Yitong Wu, Yuan Li, Yujun Li, and Wang Zhou. 2024. Mathccs: A new benchmark for mathematical classification and constructive suggestions. *arXiv preprint arXiv:2405.17642*.

Zhen Yuan, Yifan Zhang, Jing Liu, Yuxiang Wang, Jie Zhang, Hanwang Liu, and Tat-Seng Chua. 2024. Fermat: A benchmark for evaluating vlm's ability in factual error correction of handwritten math solutions. *arXiv preprint arXiv:2405.10100*.

A Per-Task Performance Data

This appendix provides the detailed per-task scores for a selection of the evaluated models.

Table 3: Per-task scores — `openai_o4-mini`.

Task	Examples	Accuracy	Average Score	Expected Score	Cost
13	21	47.6%	1.48	0.95	\$0.4259
14	18	27.8%	1.72	1.28	\$0.3465
15	19	**63.2%**	1.68	1.11	\$0.3115
16	17	**82.4%**	1.24	1.29	\$0.2957
17	15	33.3%	1.20	1.20	\$0.2560
18	16	**68.8%**	2.12	2.38	\$0.3543
19	16	**56.2%**	1.75	2.06	\$0.2879

Table 4: Per-task scores — `qwen-2.5-vl-32b`.

Task	Examples	Accuracy	Average Score	Expected Score	Cost
13	21	42.9%	1.62	0.95	\$0.1095
14	18	22.2%	2.17	1.28	\$0.0999
15	19	**52.6%**	1.58	1.11	\$0.0875
16	17	**70.6%**	1.41	1.29	\$0.0783
17	15	33.3%	1.73	1.20	\$0.0753
18	16	37.5%	2.75	2.38	\$0.0970
19	16	43.8%	2.06	2.06	\$0.0868

Table 5: Per-task scores — `arcee-ai_spotlight`.

Task	Examples	Accuracy	Average Score	Expected Score	Cost
13	21	28.6%	0.95	0.95	< \$0.01
14	18	11.1%	2.72	1.28	< \$0.01
15	19	21.1%	0.74	1.11	< \$0.01
16	17	47.1%	1.47	1.29	< \$0.01
17	15	13.3%	2.67	1.20	< \$0.01
18	16	18.8%	2.12	2.38	< \$0.01
19	16	37.5%	2.62	2.06	< \$0.01

Table 6: Per-task scores — `gemini-2.5-flash-preview`.

Task	Examples	Accuracy	Average Score	Expected Score	Cost
13	21	42.9%	1.48	0.95	\$0.0493
14	18	38.9%	1.00	1.28	\$0.0616
15	19	47.4%	1.79	1.11	\$0.0401
16	17	47.1%	1.41	1.29	\$0.0419
17	15	46.7%	0.73	1.20	\$0.0387
18	16	43.8%	1.81	2.38	\$0.0713
19	15	**60.0%**	1.40	2.13	\$0.0414

Table 7: Per-task scores — `gemini-2.5-flash-preview_thinking`.

Task	Examples	Accuracy	Average Score	Expected Score	Cost
13	21	**66.7%**	1.57	0.95	\$0.1096
14	18	22.2%	1.33	1.28	\$0.1043
15	19	31.6%	1.21	1.11	\$0.0998
16	17	**58.8%**	1.53	1.29	\$0.0964
17	15	26.7%	0.73	1.20	\$0.0908
18	16	43.8%	2.50	2.38	\$0.1037
19	16	50.0%	2.44	2.06	\$0.1787

B Representative Example Analysis

This appendix provides a detailed analysis of a representative example from our benchmark to illustrate the evaluation process and the typical performance patterns of the models.

Table 8: Per-task scores — `gemini-2.0-flash-001`.

Task	Examples	Accuracy	Average Score	Expected Score	Cost
13	21	**61.9%**	1.48	0.95	$0.0295
14	18	33.3%	1.61	1.28	$0.0284
15	19	42.1%	1.42	1.11	$0.0276
16	17	**58.8%**	1.47	1.29	$0.0270
17	15	40.0%	0.93	1.20	$0.0254
18	16	37.5%	2.50	2.38	$0.0286
19	16	43.8%	2.31	2.06	$0.0392

Table 9: Per-task scores — `gemini-2.0-flash-lite-001`.

Task	Examples	Accuracy	Average Score	Expected Score	Cost
13	21	**57.1%**	1.38	0.95	$0.0059
14	18	22.2%	1.50	1.28	$0.0056
15	19	31.6%	1.26	1.11	$0.0052
16	17	**52.9%**	1.29	1.29	$0.0051
17	15	40.0%	0.87	1.20	$0.0048
18	16	25.0%	2.62	2.38	$0.0054
19	16	37.5%	2.06	2.06	$0.0049

B.1 Analysis of Solution 18.3.3

B.1.1 Problem Statement

Find all values of parameter a for which the equation has exactly three distinct roots:

$$\sqrt{3x^2 + 2ax + 1} = x^2 + ax + 1$$

B.1.2 Official Grading Criteria (Task 18)

- **4 points:** A well-reasoned, correct solution is provided.
- **3 points:** A set of parameter values is obtained that differs from the correct set only by the inclusion/exclusion of boundary points.
- **2 points:** A correct interval of parameter values is obtained (possibly with incorrect boundary points), OR an incorrect answer is obtained due to a computational error, but all logical steps are correct.
- **1 point:** The roots of the equation are found, and the problem is correctly reduced to investigating these roots under the given condition(s).
- **0 points:** The solution does not meet any of the criteria above.

B.1.3 Visual Materials

Figure 2 shows the student's handwritten solution, and Figure 3 shows the official correct solution provided in the EGE expert guide.

Найдите все значения a, при каждом из которых уравнение $\sqrt{3x^2+2ax+1}=x^2+ax+1$ имеет ровно три различных корня.

Ответ: $-2\le a<-1$; $-1<a<1$; $1<a\le 2$.

$\sqrt{3x^2+2ax+1}=x^2+ax+1$

$\begin{cases} x^2+ax+1\geq 0 \\ 3x^2+2ax+1=x^4+a^2x^2+1+2x^2+2ax^3+2ax \end{cases}$

$\begin{cases} x^2+ax+1\geq 0 \\ x^4+a^2x^2-x^2+2ax^3=0. \end{cases}$

$\begin{cases} x^2+ax+1\geq 0 \\ x^2(x^2+a^2-1+2ax)=0. \end{cases}$

$\begin{cases} x^2+ax+1\geq 0. \\ \left[\begin{array}{l} x=0 \\ x^2+2ax+a^2-1=0; \end{array}\right. \end{cases}$

Уравнение имеет решение, когда $x^2+2ax+a^2-1$ имеет 2 корня и они удовлетворяют неравенству $x^2+ax+1\geq 0$.

$x^2+2ax+a^2-1=0$

$(x+a)^2-1=0$

$(x+a-1)(x+a+1)=0.$

$\left[\begin{array}{l} x=-a+1 \\ x=-a-1 \end{array}\right.$ Подставим x в $x^2+ax+1\geq 0.$

1) $(-a+1)^2+a(-a+1)+1\geq 0.$ 2) $(-a-1)^2+a(-a-1)+1\geq 0$

$a^2-2a+1-a^2+a+1\geq 0.$ $a^2+2a+1-a^2-a+1\geq 0$

$-a+2\geq 0$ $a\leq 2$ $2a+2\geq 0$

$a\in[-1;2].$ $a\geq -1.$

Найдем значение x, когда они совпадают;

3 случая

1) $-a+1=-a-1$ 1) – нет решений

2) $0=-a+1$ 2) $a=1$

3) $0=-a-1$ 3) $a=-1.$ } – выкалываем эти точки

$a\in(-1;1)\cup(1;2].$

Ответ: при $a\in(-1;1)\cup(1;2]$. уравнение имеет 3 разл. корня.

Figure 2: Student's handwritten solution for problem 18.3.3. The expert-assigned score is 2.

Задание 18.3

Найдите все значения a, при каждом из которых уравнение

$$\sqrt{3x^2+2ax+1}=x^2+ax+1$$

имеет ровно три различных корня.

Решение. Исходное уравнение равносильно уравнению $3x^2+2ax+1=\left(x^2+ax+1\right)^2$ при условии $x^2+ax+1\geq 0$.

Решим уравнение $3x^2+2ax+1=\left(x^2+ax+1\right)^2$:

$$3x^2+2ax+1=x^4+2ax^3+\left(a^2+2\right)x^2+2ax+1;\ x^4+2ax^3+\left(a^2-1\right)x^2=0;$$

$$x^2\left(x+a+1\right)\left(x+a-1\right)=0,\ \text{откуда}\ x=0,\ x=1-a\ \text{или}\ x=-1-a.$$

Исходное уравнение имеет три корня, когда эти числа различны и для каждого из них выполнено условие $x^2+ax+1\geq 0$.

Рассмотрим условия совпадения корней. При $a=1$ имеем $1-a=0$. При $a=-1$ имеем $-1-a=0$. При остальных значениях a числа 0, $1-a$, $-1-a$ различны.

При $x=0$ получаем: $x^2+ax+1=1\geq 0$ при всех значениях a.

При $x=1-a$ получаем: $x^2+ax+1=(1-a)^2+a(1-a)+1=2-a$.

Это выражение неотрицательно при $a\leq 2$.

При $x=-1-a$ получаем: $x^2+ax+1=(-1-a)^2+a(-1-a)+1=a+2$.

Это выражение неотрицательно при $a\geq -2$.

Таким образом, исходное уравнение имеет ровно три различных корня при

$$-2\leq a<-1;\ -1<a<1;\ 1<a\leq 2.$$

Ответ: $-2\leq a<-1$; $-1<a<1$; $1<a\leq 2$.

Критерии оценивания выполнения задания	**Баллы**
Обоснованно получен верный ответ	4
С помощью верного рассуждения получено множество значений a, отличающееся от искомого только исключением точек a = –2 и/или $a=2$	3
С помощью верного рассуждения получен промежуток (–2; 2) множества значений a, возможно, с включением граничных точек ИЛИ получен неверный ответ из-за вычислительной ошибки, но при этом верно выполнены все шаги решения	2
Получены корни уравнения $3x^2+2ax+1=\left(x^2+ax+1\right)^2$: $x=0$, $x=1-a$, $x=-1-a$ и задача верно сведена к исследованию полученных корней при условии $x^2+ax+1>0$ ($x^2+ax+1\geq 0$)	1
Решение не соответствует ни одному из критериев, перечисленных выше	0
Максимальный балл	*4*

Figure 3: Official correct solution for problem 18.3.

B.2 Model Assessment Results

The table 10 summarizes the scores assigned by different models in the **With True Solution** mode. The expected score was 2.

Table 10: Model scores for solution 18.3.3.

Model	Assigned Score	Expected	Result
OpenAI o4-mini	2	2	Correct
Qwen 2.5 VL 32B	4	2	Overestimated
Google Gemini 2.0 Flash	2	2	Correct
Google Gemini 2.0 Flash Lite	2	2	Correct
Google Gemini 2.5 Flash Preview	2	2	Correct
Google Gemini 2.5 Flash Preview Thinking	2	2	Correct
Arcee-AI Spotlight	0	2	Underestimated

B.2.1 Key Observations

- **High Accuracy of Most Models:** The majority of the models (5 out of 7) successfully handled the task, assigning the correct score of 2. This group included OpenAI o4-mini and all the tested models from the Google Gemini family.

- **Divergent Errors:** Two models evaluated the solution incorrectly, and their errors were opposites. Qwen 2.5 VL 32B significantly overestimated the score (**4**), while Arcee-AI Spotlight failed to produce a final answer: it became stuck in a loop of writing out equations, which resulted in a score of (**0**).

- **Distinct Failure Modes:** The errors highlight very different failure modes. One model overestimated the score, while the other failed to complete the task entirely. This points to unique flaws in the logic of each model rather than a shared, systematic bias.

B.3 Full Model Responses and Prompt (Translated to English)

B.3.1 Prompt Used for Evaluation (With True Solution)

Analyze the solution of task 18

(an equation, inequality, or system thereof with a parameter) and evaluate it according to the criteria.

Task

{task description}

Assessment criteria for task 18

- **4 points:** A correct answer is obtained with justification.

- **3 points:** A set of parameter values is obtained through correct reasoning, differing from the required set only by the exclusion of boundary points or the inclusion of points not belonging to the answer.

- **2 points:** An interval of the set of parameter values is obtained through correct reasoning, possibly including boundary points, OR an incorrect answer is obtained due to a computational error, but all steps of the solution are correctly performed.

- **1 point:** The roots of the equation are found, and the problem is correctly reduced to the investigation of these roots under the given condition(s).

- **0 points:** The solution does not meet any of the criteria listed above.

IMPORTANT: Assessment Principles

- Evaluate the solution **STRICTLY** according to the criteria.
- Pay attention to mathematical correctness, not the presentation.
- Compare the student's solution with the correct solution provided as a reference.
- Check if the student has correctly performed all key steps of the solution.
- If the student used a different approach, evaluate its correctness and compliance with the criteria.
- Problems with parameters allow for various solution methods: algebraic, geometric, functional.
- Pay **SPECIAL ATTENTION** to the correctness of handling boundary points and the completeness of considering all cases.
- When assessing for 3 points: check that the difference from the correct answer is **ONLY** in the boundary points, not in the main intervals.
- For a geometric approach: check the correctness of the interpretation and the justification of all geometric statements.

IMPORTANT: Instructions for working with the correct solution and the student's solution

You are provided with:

1. The correct solution to the task - use it as a reference for comparison.
2. The student's solution - this is what you must evaluate.

During the analysis:

- Compare each step of the student's solution with the corresponding step of the correct solution.
- Note all deviations and errors.
- Check if the intermediate and final results match.
- Pay special attention to the handling of boundary points and the completeness of considering all cases.

Instructions for checking the solution of a problem with a parameter

1. Check the solution method:
   - Correctness of the chosen approach (algebraic, geometric, functional).
   - Correctness of applying formulas and theorems.
   - Completeness of considering all cases.
2. Check mathematical correctness:
   - Correctness of algebraic transformations.
   - Correctness of working with inequalities.
   - Correctness of finding the domain of permissible values (ODZ).
3. Check the handling of boundary points:
   - Correctness of determining the boundary points.
   - Correctness of including/excluding boundary points in the answer.
4. For a geometric approach, check:
   - Correctness of the geometric interpretation of the conditions.
   - Justification of all geometric statements.
   - Completeness of the analysis of all possible relative positions of geometric objects.

CRITICALLY IMPORTANT: Immediately compare the student's answers with the correct ones!

- **FIRST AND FOREMOST**, check if the student's answer matches the correct answer.
- If the student's answer is **INCORRECT**, this **MUST** be taken into account in the assessment.
- Even if all transformations are performed correctly, but the answer is wrong due to a non-computational error - this must affect the score.
- Do not forget to note all discrepancies between the student's answer and the correct answer.

IMPORTANT: Distinguish between computational and conceptual errors

- **Computational errors:** errors in arithmetic operations, simplifying expressions, calculating values.
- **Conceptual errors:** incorrect application of formulas, wrong solution method, errors in understanding the properties of the parameter.

If a student made only computational errors, but the solution method is correct - this may correspond to the 2-point criterion. If a student made conceptual errors - this usually corresponds to a lower score criterion.

Assessment Examples

Example 1 (score: 4 points) The solution is complete and justified. All values of the parameter for which the system has exactly two solutions are found correctly. All cases are considered, and boundary points are analyzed correctly.

Example 2 (score: 3 points) All stages are present in the solution. Through correct reasoning, a set of parameter values is obtained that differs from the required set only by the exclusion of boundary points or the inclusion of points not belonging to the answer. For example, the answer obtained is $(-2, 2)$ instead of $[-2, 2]$ or $[-2, -1) \cup (-1, 1) \cup (1, 2]$ instead of $[-2, -1) \cup (-1, 1) \cup [1, 2]$.

Example 3 (score: 2 points) The solution is logical, all steps are present, but a computational error was made when solving the inequality, which led to an incorrect answer. Or, through correct reasoning, an interval of the set of parameter values was obtained, possibly including boundary points, but not all possible cases were considered.

Example 4 (score: 1 point) The roots of the equation are found in the solution, and the problem is correctly reduced to the investigation of these roots under the condition $x^2 + ax + 1 \geq 0$, but the further investigation is incorrect or absent.

Example 5 (score: 0 points) The solution contains fundamental errors in understanding the problem or in mathematical reasoning. For example, the roots of the original equation are found incorrectly, or the condition of the problem is misinterpreted.

IMPORTANT: Instructions for working with multiple images

If the solution (correct or student's) is presented in several images:

1. Consider all images as a single solution.
2. Analyze them in sequential order.
3. Take into account information from all images when assessing.

Response Format

Structure your response as follows:

Analysis of the solution [Detailed analysis indicating correct steps and errors]

Comparison with the correct solution [Comparison of the student's approach with the reference solution, indicating key differences]

Final score [Score: X points]

IMPORTANT: The "Final score" section **MUST** be formatted **EXACTLY** in this format: "Final score" - heading. "[Score: X points]" - where X is a number 0, 1, 2, 3, or 4. **DO NOT CHANGE** this format and do not add any other text to it!

Justification of the score [Detailed justification in accordance with the criteria]

Recommendations (if applicable) [Recommendations for correcting errors]

B.3.2 Response from OpenAI o4-mini (Score: 2 - Correct)

Analysis of the solution

1. The student wrote down an equivalent system:
   - a) the equation after squaring;
   - b) the condition that the expression under the root is non-negative.
2. Found the roots of the original equation: $x = 0$, $x = 1$-a, $x = $-$1$-$a$.
3. Analyzed the condition $x^2 + ax + 1 \geq 0$ for these values of x:
   - for $x = 0$, the condition is always met;
   - for $x = 1$-a, from the inequality, they obtained $a \leq 2$;
   - for $x = $-$1$-$a$, they obtained $a \geq $-$2$.
4. Concluded that for three distinct roots, it is necessary that -$2 \leq a \leq 2$ and $a \neq 1$, $a \neq $-$1$.
5. However, in the final answer, the student wrote $a \in ($-$1; 1) \cup (1; 2]$, thus completely omitting the interval $[$-$2;$-$1)$.

Error: when combining the resulting intervals and excluding the boundary points, the student discarded the left part $[$-$2;$-$1)$ for a purely computational/descriptive reason. All key steps (finding roots, conditions on a, accounting for exclusions) were performed correctly, but the answer contains an inaccuracy.

Comparison with the correct solution

Reference answer: $a \in [$-$2;$-$1) \cup ($-$1; 1) \cup (1; 2]$.

The student obtained the same intermediate boundaries -2 and 2, and correctly excluded $a = \pm 1$, but missed the part $[$-$2;$-$1)$ in the final answer. Therefore, the method is correct, but a computational (descriptive) slip in combining the results led to an incomplete answer.

Final score

[Score: 2 points]

Justification of the score

According to the criteria:

- All key steps of the solution (finding roots, domain of permissible values, conditions on the parameter, exclusion of boundary values) were performed correctly.
- The final answer is incorrect due to a computational/descriptive error when combining the intervals.
- The approach is mathematically correct, but the result is inaccurate.

This corresponds to the 2-point criterion: "An interval of the set of parameter values is obtained through correct reasoning, possibly including/excluding boundary points, but the overall answer is incorrect due to a computational error."

Recommendations

- Be more careful when combining intervals and check that no sections have been dropped.
- After obtaining the conditions on the parameter, explicitly write down all three 'pieces' and only then combine them, excluding the single points.

B.3.3 Response from Qwen 2.5 VL 32B (Score: 4 - Overestimated)

Analysis of the solution

Step 1: Forming the system of equations The student correctly understands that for the initial equation $\sqrt{3x^2 + 2ax + 1} = x^2 + ax + 1$, two conditions must be met:

1. $x^2 + ax + 1 \geq 0$ (since the square root is defined only for non-negative values).
2. $3x^2 + 2ax + 1 = (x^2 + ax + 1)^2$ (the main equation).

This approach is correct, and the student correctly transforms the equation:

$$3x^2 + 2ax + 1 = x^4 + 2ax^3 + (a^2 + 2)x^2 + 2ax + 1.$$

Then, they combine like terms and obtain:

$$x^4 + 2ax^3 + (a^2 - 1)x^2 = 0.$$

Next, the student correctly factorizes this equation:

$$x^2(x^2 + 2ax + (a^2 - 1)) = 0.$$

From this, they find the roots:

$$x = 0 \quad \text{and} \quad x^2 + 2ax + (a^2 - 1) = 0.$$

For the quadratic equation $x^2 + 2ax + (a^2 - 1) = 0$, the student finds the roots:

$$x = -a + 1 \quad \text{and} \quad x = -a - 1.$$

Thus, the student correctly finds three roots: $x = 0$, $x = -a + 1$, and $x = -a - 1$.

Step 2: Analysis of the conditions for distinct roots The student recognizes that for the equation to have three distinct roots, these roots must be different. They check the cases where the roots coincide:

1. $x = 0$ and $x = -a + 1$: This is possible if $-a + 1 = 0 \Rightarrow a = 1$.
2. $x = 0$ and $x = -a - 1$: This is possible if $-a - 1 = 0 \Rightarrow a = -1$.
3. $x = -a + 1$ and $x = -a - 1$: This is possible if $-a + 1 = -a - 1$, but this is impossible as it leads to a contradiction.

Step 3: Verifying the condition $x^2 + ax + 1 \geq 0$ The student checks the condition $x^2 + ax + 1 \geq 0$ for each root:

1. For $x = 0$: $x^2 + ax + 1 = 1 \geq 0$ is always met.
2. For $x = -a + 1$: $x^2 + ax + 1 = (-a + 1)^2 + a(-a + 1) + 1 = a^2 - 2a + 1 - a^2 + a + 1 = -a + 2$. This is non-negative if $-a + 2 \geq 0 \Rightarrow a \leq 2$.
3. For $x = -a - 1$: $x^2 + ax + 1 = (-a - 1)^2 + a(-a - 1) + 1 = a^2 + 2a + 1 - a^2 - a + 1 = a + 2$. This is non-negative if $a + 2 \geq 0 \Rightarrow a \geq -2$.

Step 4: Combining the conditions The student correctly combines all the obtained conditions:

- $a \neq 1$ and $a \neq -1$ (for distinct roots).
- $a \leq 2$ and $a \geq -2$ (to satisfy the condition $x^2 + ax + 1 \geq 0$).

Thus, the student obtains:

$$a \in [-2, -1) \cup (-1, 1) \cup (1, 2].$$

Comparison with the correct solution

The correct solution and the student's solution completely coincide in the answer:

$$a \in [-2, -1) \cup (-1, 1) \cup (1, 2].$$

The student correctly performed all the key steps:

1. Transforming the original equation.
2. Finding the roots.
3. Analyzing the conditions for distinct roots.
4. Verifying the condition $x^2 + ax + 1 \geq 0$.
5. Combining all the obtained conditions.

Final score

[Score: 4 points]

Justification of the score

The student justifiably obtained the correct answer by correctly performing all the key steps of the solution. All intermediate results are correct, the boundary points are handled correctly, and all possible cases are considered. The solution method fully complies with the mathematical requirements of the problem, and the answer completely matches the correct solution.

Recommendations

The student's solution contains no errors and requires no corrections. However, the student could improve the presentation by making it more detailed and structured, so that readers can better follow their reasoning. For example, they could explain in more detail why the roots coincide when $a = 1$ or $a = -1$, and why these values are excluded from the answer.

RoMath: A Mathematical Reasoning Benchmark in Romanian

Adrian Cosma[1,2], **Ana-Maria Bucur**[3], **Emilian Radoi**[1]
[1]POLITEHNICA Bucharest National University of Science and Technology
[2]Dalle Molle Institute for Artificial Intelligence Research (IDSIA)
[3]Interdisciplinary School of Doctoral Studies, University of Bucharest
Correspondence: cosma.i.adrian@gmail.com

Abstract

Mathematics has long been conveyed through natural language, primarily for human understanding. With the rise of mechanized mathematics and proof assistants, there is a growing need to understand informal mathematical text, yet most existing benchmarks focus solely on English, overlooking other languages. This paper introduces RoMath, a Romanian mathematical reasoning benchmark suite comprising three subsets: *Baccalaureate*, *Competitions* and *Synthetic*, which cover a range of mathematical domains and difficulty levels, aiming to improve non-English language models and promote multilingual AI development. By focusing on Romanian, a low-resource language with unique linguistic features, RoMath addresses the limitations of Anglo-centric models and emphasizes the need for dedicated resources beyond simple automatic translation. We benchmark several open-weight language models, highlighting the importance of creating resources for underrepresented languages. The code and datasets are available for research purposes.

> "Matematica s-o fi scriind cu cifre dar poezia nu se scrie cu cuvinte."[1]
>
> *Nichita Stănescu, "Matematica poetică"*, Poem dedicated to mathematician Solomon Marcus.

1 Introduction

Mathematics has been a central intellectual preoccupation to humans since the beginning of civilization, the first mathematical writings dating back approximately 4000 years (Friberg, 1981). Historically and in the present, mathematics has been mostly written, spoken and taught in natural language, albeit with its own specialized vocabulary, having strict formalism only sparsely introduced between free-text explanations and reasoning. The primary audience of mathematical reasoning is other humans, not computers. The natural language of mathematics contains a mix of formulas, symbols, neologisms, jargon and words with different meanings than their common meaning (e.g., "real" / "imaginary" numbers). Mathematics implies rigor and precise reasoning, qualitatively different from general NLP. There is a pressing need to automatically process and understand the existing large amount of mathematical text written in natural language to enable efficient knowledge extraction, facilitate automated theorem proving, and enhance accessibility for both researchers and automated systems.

Recently, Large Language Models (LLMs) have shown great promise in handling a multitude of natural language tasks, including tackling mathematical reasoning problems (Ahn et al., 2024; Yue et al., 2023; Azerbayev et al., 2024; Shao et al., 2024). Out of the common benchmark suite for evaluating LLMs, datasets such as GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) remained central in the development of reasoning models (Jaech et al., 2024; DeepSeek-AI et al., 2025), and continue to be challenging even for the larger, proprietary models (Arora et al., 2023).

Current mathematics benchmarks and datasets have focused solely on English, mostly disregarding other low-resourced languages. The tacit requirement for using AI tools is fluency in English (Shi et al., 2022). However, mathematical reasoning ability is independent of the underlying language (Rescorla, 2024) and Anglo-centric models have been shown to exhibit the same biases of the English language, even when prompted in other languages (Wendler et al., 2024; Wang et al., 2023; Liu et al., 2023). The focus on datasets and models in a language other than English allows the democratization of learning for underrepresented languages and cultures.

Recently, Romanian LLM development has

[1]English translation: *"Mathematics may be written with numbers, but poetry is not written with words."*

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 95–111
November 9, 2025 ©2025 Association for Computational Linguistics

started to flourish with initiatives such as OpenLLM-Ro (Masala et al., 2024), having fine-tuned several LLMs on Romanian text. However, for evaluation, the authors used translated versions of popular English datasets and several native Romanian benchmarks, but no evaluation was performed on dedicated reasoning tasks in Romanian. Aside from code generation (Cosma et al., 2024; Dumitran et al., 2024), currently, there is no reasoning benchmark for Romanian.

In this paper, we propose RoMath[2,3], a Romanian mathematical reasoning benchmarking suite comprised of three datasets, *Baccalaureate*, *Competitions* and *Synthetic*, each with its own particularities. RoMath aims to provide a comprehensive benchmark suite, having high-school-level problems across multiple domains (linear and abstract algebra, calculus, limits, geometry, probabilities) and across multiple levels of difficulty, ranging from easy calculations, to baccalaureate-level problems, to more difficult, proof-centric, competition-level problems. The purpose of RoMath is to provide a mathematical benchmark for Romanian and to stimulate the development of enhanced reasoning capabilities of non-English LLMs.

This work makes the following contributions:

1. We construct and release **RoMath**, a novel mathematical reasoning benchmark suite with 76,910 problem statements in Romanian, consisting of three subsets, each with its own particularities and difficulty levels: *Baccalaureate* (5,777 problems), *Competitions* (1,133 problems) and *Synthetic* (63,000 problems). We collect and curate math problems using a semi-automatic workflow using foundational LLMs for providing structured output from unstructured raw OCR input and annotating problems with relevant metadata.

2. We provide a comprehensive benchmark of several English and Romanian open-weight LLMs under several common scenarios - zero-shot, LoRA fine-tuning (Hu et al., 2022) and training with verifiable rewards using GRPO (Shao et al., 2024). Furthermore, we provide an evaluation procedure using an LLM-as-a-judge paradigm (Zheng et al., 2023) for proofs, and analyze its performance to properly estimate solution correctness.

3. We show that simple translation of problem statements is not enough, as sub-par translations of precise mathematical language significantly reduce performance. Consequently, we emphasize the need for more dedicated resources in languages other than English.

2 Related Work

Pretraining datasets for mathematics. There has been an ongoing interest in representation learning for both mathematical expressions and text (Peng et al., 2021; Collard et al., 2022). However, beyond representation learning, with the recent success of LLMs in a wide range of tasks, there has been increased attention to training and evaluating the mathematical reasoning of LLMs. For pretraining, the general approach is to filter Common Crawl web pages and PDFs to obtain high quality math tokens. For instance, datasets such as MathWebPages (Lewkowycz et al., 2022), ProofPile (Azerbayev et al., 2023a), and OpenWebMath (Paster et al., 2023) are used to pretrain high-performing LLMs specialized in math, such as Minerva (Lewkowycz et al., 2022) and LLema (Azerbayev et al., 2023b).

Mathematical reasoning benchmarks. Regarding benchmarks, the most popular dataset is GSM8K (Cobbe et al., 2021), containing middle-school Math Word Problems (MWPs). An improved variant that contains process supervision (i.e., supervision at each intermediary reasoning step) is PRM800K (Lightman et al., 2023). However, these datasets are regarded as too simple to demonstrate the advanced mathematical reasoning of LLMs. Consequently, MATH (Hendrycks et al., 2021) is a comparatively more difficult dataset, containing high-school problems from domains such as calculus, linear algebra, geometry and number theory. MathVISTA (Lu et al., 2024) is another similar benchmark, that contains mathematical reasoning in visual contexts (e.g., plots, natural images, functions).

Aside from simple word problems (Cobbe et al., 2021) and datasets focused on QA-type problems, more difficult competition-level benchmarks have been proposed. For instance, ARB (Sawada et al., 2023) is a dataset comprised of problems from math competitions and problems from specialized books, with special care taken to avoid data contamination. While it contains problems that require proofs, ARB only contains 105 problems. MathOdyssey (Fang et al., 2024) contains difficult

[2]GitHub: github.com/cosmaadrian/romath
[3]Huggingface: hf.co/datasets/cosmaadrian/romath

Name	# Problems	Level	Language	Notes
APE210K (Zhao et al., 2020)	210K	E	Chinese	Requires basic arithmetic and common sense
MATH23K (Ling et al., 2017)	23K	E	Chinese	Contains questions, answers and rationales
CMATH (Wei et al., 2023)	1.7K	E	Chinese	Contains number of reasoning steps
ARMATH (Alghamdi et al., 2022)	6K	E	Arabic	Novel problems and inspired by MATH23K
GSM8K (Cobbe et al., 2021)	8.5k	M	English	Linguistically diverse.
MATH (Hendrycks et al., 2021)	12.5k	H	English	Problems are put into difficulty levels 1-5
PRM800K (Lightman et al., 2023)	12k	H	English	MATH w/ step-wise labels
MathOdyssey (Fang et al., 2024)	387	H C	English	Problems from GAIC Math 2024
ARB (Sawada et al., 2023)	105	C	English	Contest problems and university math proof
AQUA (Ling et al., 2017)	100K	C	English	GRE&GMAT questions
RoMath				
Baccalaureate	5.8k	H	Romanian	Easy-medium, baccalaureate problems.
Competitions	1.1k	M H	Romanian	Difficult, proofs, competition problems.
Synthetic	63k	H	Romanian	Calculation-based, procedurally generated.

Table 1: Comparison with other mathematical reasoning benchmarks. RoMath is the only Romanian mathematics benchmark outside of translated versions of English benchmarks. Table adapted from Ahn et al. (2024).
E = Elementary, M = Middle School, H = High School, C = College.

high-school and university-level problems, but it is similarly small, as it contains only 387 problems.

Non-English benchmarks. Regarding datasets in languages other than English, there have been efforts in Arabic with datasets such as ArMATH (Alghamdi et al., 2022) and Chinese with Ape210k (Zhao et al., 2020), Math23k (Ling et al., 2017), CMath (Wei et al., 2023). Otherwise, outside of (automatically) translated versions of popular sets such as GSM8k (Masala et al., 2024), as far as we know, no datasets currently exist for Romanian or other Latin languages.

Comparison with prior work. Table 1 shows a comparison between similar datasets and RoMath. RoMath comprises middle-school, high-school and competitive high-school problems in Romanian covering multiple subjects and types of problems (proofs, calculations, equations, etc.). Different from prior datasets, RoMath is the first dedicated resource for mathematical reasoning in Romanian, a low-resource language of $\sim$23M speakers, which has its unique linguistic particularities (Dinu and Dinu, 2005; Dinu and Enăchescu, 2007).

3 Method

We describe below the process for collecting *Baccalaureate* and *Competitions*, the two subsets that are collected by crawling publicly available PDFs. The *Synthetic* subset is comprised of programmatically generated problems directly in Romanian.

3.1 Dataset Construction

In order to construct a high-quality set of mathematical problems paired with solutions, we crawl publicly available PDFs from country-wide mathematics competitions and questions from the Romanian baccalaureate exam. Figure 1 showcases our approach. After collecting raw PDFs, usually having separate documents for problem sets and their respective solutions, we utilize an academic document-focused OCR (i.e., MathPix (Mathpix, 2024)) to extract the underlying text and mathematical formulas / statements in LaTeX format. The final output is represented in Markdown format.

To parse the content, instead of relying on brittle handcrafted rules and regex expressions, we utilize a commercial LLM (i.e., Claude 3 Sonnet (Anthropic, 2024)) to parse the raw text and to output structured JSON from unstructured Markdown. The LLM is provided with several examples of how to structure the final JSON (see Appendix A, Table 6 for the system prompt). The JSON output contains the LaTeX-formatted problem statement and its appropriate solution. Finally, we again utilize a commercial LLM to annotate the domain of the problem and to extract final answers for non-proof problems for easier evaluation (similar to Hendrycks et al. (2021), we enclose the final

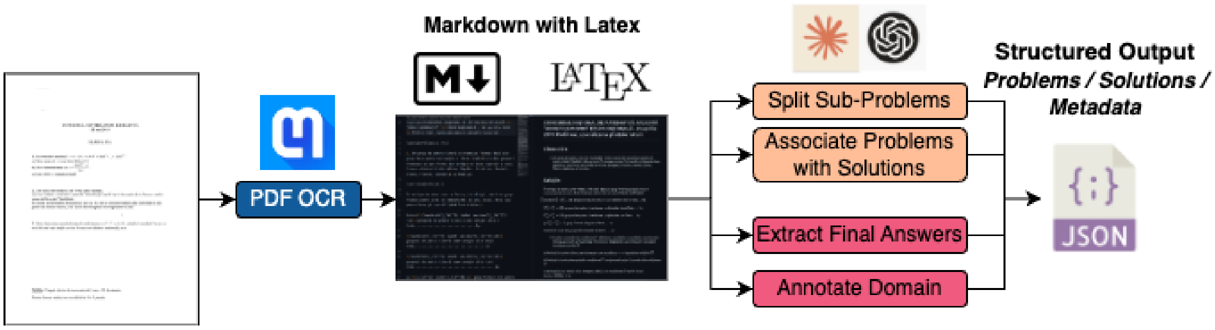


Figure 1: Overall diagram of our approach to curating problems from existing PDFs. We employ MathPix (Mathpix, 2024) to OCR PDFs and obtain markdown with LaTeX formatting for mathematical statements. We further process the markdown using proprietary LLMs to split into sub-problems, associate problems with the appropriate solution and annotate each problem with metadata.

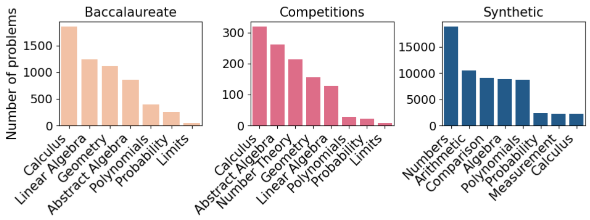


Figure 2: Distribution of the number of problems per domain for *Baccalaureate*, *Competitions* and *Synthetic*.

Synthetic

Problem Statement: Care este rezultatul împărțirii lui -54 la -36495?

Solution: $\frac{6}{4055}$

Problem Statement: Fie u definit ca fiind $(2 - \frac{18}{15}) \cdot 5$. Găsește valoarea lui r din ecuațiile $u \cdot y + 8 = 0, -28 = 3 \cdot r + 4 \cdot y - 5$.

Solution: -5

Baccalaureate

Problem Statement: Se consideră funcția $f : \mathbf{R} \to \mathbf{R}, f(x) = e^x - x$. Să se calculeze $\int_0^1 f(x)dx$.

Solution: $\int_0^1 f(x)dx = e - \frac{3}{2}$. Soluția finală este $\boxed{e - \frac{3}{2}}$

Problem Statement: Se consideră funcțiile $f_n : \mathbf{R} \to \mathbf{R}, f_1(x) = x^3 - 3x^2 + 3x$ şi $f_{n+1}(x) = (f_1 \circ f_n)(x), \forall n \in \mathbf{N}^*, \forall x \in \mathbf{R}$. Să se rezolve în mulțimea numerelor reale ecuația $f_1(x) + f_2(x) + f_3(x) - 3 = 0$.

Solution: Observăm că $x = 1$ este soluție. Dacă $x > 1 \Rightarrow f_1(x) + f_2(x) + f_3(x) > 3$. Analog dacă $x < 1 \Rightarrow f_1(x) + f_2(x) + f_3(x) < 3$. Deci $x = 1$ este soluție unică. Soluția finală este $\boxed{x = 1}$

Competitions

Problem Statement: Se consideră numerele complexe u, v și z astfel încât $|u| = |v| = 1$ și $|u + v| = \sqrt{3}$. Să se demonstreze că: $\mathrm{u} \cdot \overline{\mathrm{v}} + \overline{\mathrm{u}} \cdot \mathrm{v} = 1$.

Solution: $|\mathrm{u} + \mathrm{v}|^2 = 3 \Leftrightarrow (\mathrm{u} + \mathrm{v})(\overline{\mathrm{u}} + \overline{\mathrm{v}}) = 3 \Leftrightarrow \mathrm{u} \cdot \overline{\mathrm{v}} + \overline{\mathrm{u}} \cdot \mathrm{v} = 1$

Problem Statement: Să se rezolve, în $\mathbb{R}$, inecuația: $\left(\frac{2}{5}\right)^{\frac{6-5x}{5x+2}} \leq \frac{25}{4}$

Solution: $\left(\frac{2}{5}\right)^{\frac{6-5x}{5x+2}} \leq \frac{25}{4} = \left(\frac{5}{2}\right)^{-2} \Rightarrow \frac{6-5x}{5x+2} \geq -2$, cu $x \neq -\frac{2}{5}$. $\frac{6-5x}{5x+2} \geq -2 \Rightarrow \frac{x+2}{5x+2} \geq 0 \Rightarrow x \in (-\infty, -2] \cup \left(-\frac{2}{5}, \infty\right)$

Table 2: Qualitative examples from each subset of RoMath.

answer, if it exists, into a \boxed{} tag). If a problem contains multiple sub-problems, we ensure that each sub-problem is self-contained and that the solution does not rely heavily on previous sub-problems' solutions. To split a problem into sub-problems, we used a prompt (presented in Appendix A, Table 6) with specific instructions for parsing the data and output sub-questions that are self-contained. For example, if a problem is structured as follows:

<problem_statement>
<question_1>
<question_2>

The output is formatted as two separate, standalone problems:

<problem_statement> <question_1>
<problem_statement> <question_2>

Additionally, through manual inspection, we further removed any sub-questions that contained references to previous sub-questions (e.g., "Using the result from a) compute [. . .]"). Figure 2 shows the distribution of problems per domain.

3.2 RoMath Suite

RoMath is comprised of three subsets: *Baccalaureate*, *Competitions* and *Synthetic*. By its construction, each subset of RoMath features problems that require both single-step and multi-step reasoning for solving problems correctly. Usually, single-step reasoning problems involve simple calculations, while multi-step reasoning problems require solving intermediate solutions to reach a valid conclusion. Table 2 showcases selected examples from each subset.

Baccalaureate is composed of problems and solutions from the Romanian Baccalaureate exam. The Romanian Baccalaureate is a country-wide exam for graduating high-school students, comprised of three subjects, each with several problems and sub-problems. Students taking the Baccalaureate exam consider the calculus problems, such as solving an integral or computing a limit, to be the most difficult. However, the calculus problems rarely require more than 2 steps of reasoning and some calculation. This subset contains a total of 5777 problems: 4.3k problems for training and 1.48k for testing. Most problems (4617 / $\sim 80\%$) in this subset are verifiable (i.e., have a single final answer), while some (1160 / $\sim 20\%$) require proofs. Furthermore, 4038 / $\sim 69\%$ problems in this category also have intermediate steps provided in the ground-truth solution. In this set, there are multiple domains, with varying difficulty: geometry, combinatorics, abstract algebra, linear algebra, calculus (integrals and derivatives), and limits. In all categories, we discarded any problem that required reasoning over images or plots. For instance, geometry problems do not have an accompanying drawing or figure. If we encountered images in the source PDFs, we removed the problem entirely through manual inspection. The *Baccalaureate* subset includes only standalone geometry problems: an example of such a problem would be the following (here, translated in English for convenience): *"In a Cartesian coordinate system xOy we consider the points $A_n(n, 0)$ and $B_n(0, n)$, with $n \in \{1, 2, 3\}$. Calculate the area of the triangle $A_1A_2B_2$."*

Competitions is the hardest subset of RoMath, containing 1133 problems sourced from mathematics competitions, with problems ranging from local to inter-county and olympiad events, out of which 804 problems are for training and 329 for testing. Different from *Baccalaureate*, this subset also contains middle-school problems. Around half of the problems (594 / $\sim 52\%$) require proofs for a complete solution, while the rest are directly verifiable. Almost all problems in this subset have intermediate explanations. The problems in *Competitions* are considered hard, requiring insight and problem-solving skills outside of simple symbol manipulations (Polya, 1971). The extraction and post-processing steps are identical to those in *Baccalaureate*.

Synthetic is programmatically generated, using the approach of Saxton et al. (2019), in which we manually translate the source key-phrases and formulations in Romanian. Problems in this subset have a single final answer. Problems are mostly algebraic in nature, and are split into arithmetic, calculus, derivatives, integrations, polynomials, composition of problems, comparisons, manipulating expressions (e.g., simplification), numbers, measurements. All problems in this subset are verifiable, having only a single final answer provided, without intermediate steps, making it difficult to directly provide an answer without the use of external tools or chain-of-thought prompting. In contrast to the other sub-sets in RoMath, there is less linguistic variation present in problem statements, but there is complete control over correctness and difficulty. We emphasize that *Synthetic* is *not* a direct translation of the problems contained in DeepMind Mathematics (Saxton et al., 2019), but rather a manual translation of the phrases that are used to generate the problems. As such, one could generate an indefinite number of problems. We make the code for generating *Synthetic* open-source and provide, for convenience, 63k generated problems, out of which 55.9k problems for training and 7.1k for testing.

3.3 Evaluation Procedure

Generally, there are two ways to evaluate solutions: *(i)* for verifiable problems (i.e., containing a single final answer), correctness is estimated by direct string comparison between the model answer and the correct answer after normalization (Hendrycks et al., 2021; Cobbe et al., 2021) and *(ii)* using a proof-checker for problems requiring proofs (Li et al., 2024).

Evaluating the correctness of a solution to mathematical proof problems is still an open problem. Using a proof-checker is not always feasible as it requires the problems and solutions to already be formalized into the language of the proof-checker (Trinh et al., 2024), an unrealistic requirement for most mathematics written in natural language. For proof-type problems, where it is necessary to check for correctness at every reasoning step in natural language, there is no consensus on the evaluation procedure outside of formal proof-checkers.

However, more recent methods (Fang et al., 2024) have adopted a "soft" evaluation of proof solutions by employing an external judge LLM tasked to output a correctness score given the problem statement, the correct solution and a provided solution to be scored.

To evaluate solutions to RoMath, we propose the following procedure: For evaluating verifiable problems, we adopt the procedure from Hendrycks et al. (2021) for string comparison after the solutions are normalized; this requires the model to output solutions in a `\boxed{}` tag. However, if the model does not provide the solution in this format or if the problem requires a proof, we employ a judge LLM to estimate correctness, inspired by several other works (Zheng et al., 2023; Fang et al., 2024). Since the use of proprietary LLMs is prohibitively expensive, there are concerns with reproducibility, and there is no information on the architecture and training dataset, we use existing open-weight models.

4 Baselines and Results

4.1 Judge Evaluation

Very few analyses have been performed to gauge the performance of the judge LLM: for instance, a more recent study (Bavaresco et al., 2024) showed that LLMs exhibit a large variance across datasets in correlation to human judgments. However, there is no study estimating the performance of judge LLMs for mathematical reasoning in a language other than English. Using LLMs as judges is a reasonable proxy for estimating performance, and we show in Section 4.4 that performance is relatively robust across multiple judges.

In this section, we conduct an analysis of the performance of multiple open-weight judge models in evaluating solution correctness in Romanian, using both Romanian and English system prompts (see Appendix A Tables 8 and 9).

We programmatically construct a dataset of 300 problems from the training sets of *Baccalaureate* and *Competitions* containing correct and incorrect solutions (Meadows et al., 2023). Correct solutions are maintained in their original form, and we remove natural language text (keeping only mathematical expressions) of the original ground-truth solution, and incorrect solutions are either original solutions with some operators / number modified (e.g., $+$ sign changed to $-$, or $<$ symbol changed to $\geq$, and others) or a similar solution, but not exactly the same, from another problem based on the Levenstein distance.

In Table 3, we showcase the performance of multiple LLMs-as-judges on our programmatically generated dataset to estimate judge performance. We tested Qwen2 (Yang et al., 2024) family of models, as well as the math-specialized variant Qwen2-Math-7B, deepseek-math (Shao et al., 2024), Phi-3 (Abdin et al., 2024), Llama3-70B (Dubey et al., 2024), Mathstral (Mistral AI, 2024), and Mixtral-8x7b (Jiang et al., 2024). For this synthetic dataset, we obtained that Qwen2-7B-Instruct prompted in English obtained the best overall results of 91% accuracy, judging solution correctness. Surprisingly, the math-specialized models severely underperformed at this task. As such, unless otherwise specified, we used Qwen2-7B-Instruct prompted in English as a judge for the rest of the non-verifiable results.

Judge Model	System Prompt	Acc. ↑	FPR ↓	FNR ↓
deepseek-math-7b-instruct	ro	0.51	0.66	0.31
Meta-Llama-3-70B-Instruct	ro	0.86	0.24	0.03
Mixtral-8x7B-Instruct-v0.1	ro	0.84	0.27	0.03
Qwen2-Math-7B-Instruct	ro	0.87	0.22	0.03
Qwen2-7B-Instruct	ro	**0.90**	0.17	0.02
deepseek-math-7b-instruct	en	0.74	0.31	0.22
Meta-Llama-3-70B-Instruct	en	0.84	0.29	**0.01**
Mixtral-8x7B-Instruct-v0.1	en	0.84	0.27	0.03
Qwen2-Math-7B-Instruct	en	0.89	0.16	0.06
Qwen2-7B-Instruct	en	**0.91**	**0.12**	0.05

Table 3: Judge LLM performance on a programmatically generated dataset of correct and incorrect solutions.

4.2 Model Benchmark

We chose to benchmark several open-weight LLMs, as opposed to proprietary models, to make the benchmark reproducible and to avoid unnecessary inference costs. We evaluated the performance under 0-shot and LoRA fine-tuned models for Qwen2-7B, Phi-3, Meta-Llama-8B and math-specialized variants such as Qwen2-Math-7B, deepseek-math-7b, Mathstral-7b. We evaluated larger models under 0-shot setting: Meta-Llama-70B and Mixtral-8x7B. Furthermore, we also evaluated Romanian-specialized models trained with continual pretraining on Romanian tokens, but with no focus on math tokens: RoLlama3-8B and RoMistral-7b (Masala et al., 2024). The prompt used is presented in Appendix A, Table 7.

For fine-tuning the models, we used LoRA (Hu et al., 2022), using a rank of 8, alpha of 32 and dropout of 0.1, applied on all linear layers. Due to hardware limitations, we used a small batch size of 4 and a learning rate of 0.00002 with a linear decay over the 3 training epochs.

In Table 4, we showcase the performance of the models under zero-shot, and LoRA-fine-tuned scenarios. The best performing model on the *Baccalaureate* subset is deepseek-math-7b, while on *Competitions* and *Synthetic* Mathstral-7b obtains the best results. However, the Romanian models, RoLlama-8b and RoMistral-7b obtain competitive results on all subsets, which can be attributed to

	Model	Scenario	Baccalaureate Accuracy	Baccalaureate F1	Competitions Accuracy	Competitions F1	Synthetic Accuracy	Synthetic F1
Romanian	OpenLLM-Ro/RoLlama3-8b-Instruct	0-shot	0.50	0.67	0.48	0.65	0.18	0.30
		fine-tuned	0.18	0.31	0.50	0.67	–	–
	OpenLLM-Ro/RoMistral-7b-Instruct	0-shot	0.50	0.66	0.44	0.61	0.16	0.27
		fine-tuned	0.18	0.31	0.36	0.53	–	–
General-Purpose	Qwen/Qwen2-7B-Instruct	0-shot	0.40	0.57	0.55	0.71	0.29	0.45
		fine-tuned	0.54	0.70	0.48	0.65	–	–
	microsoft/Phi-3-mini-4k-instruct	0-shot	0.36	0.53	0.33	0.50	0.07	0.14
		fine-tuned	0.25	0.40	0.41	0.58	–	–
	meta-llama/Meta-Llama-3-8B-Instruct	0-shot	0.34	0.51	0.53	0.69	0.25	0.40
		fine-tuned	0.18	0.31	0.33	0.49	–	–
Math-Specialized	Qwen/Qwen2-Math-7B-Instruct	0-shot	0.32	0.48	0.55	0.71	0.27	0.43
		fine-tuned	0.48	0.65	0.57	0.73	–	–
	deepseek-ai/deepseek-math-7b-instruct	0-shot	0.56	0.72	0.59	0.74	0.21	0.35
		fine-tuned	0.29	0.44	0.56	0.72	–	–
	mistralai/Mathstral-7b-v0.1	0-shot	0.30	0.46	0.61	0.75	0.36	0.53
		fine-tuned	0.21	0.34	0.56	0.71	–	–
Large	meta-llama/Meta-Llama-3-70B-Instruct	0-shot	0.25	0.40	0.22	0.36	0.10	0.19
		5-shot	0.08	0.15	0.09	0.16	0.07	0.13
	mistralai/Mixtral-8x7B-Instruct-v0.1	0-shot	0.43	0.60	0.60	0.75	0.32	0.48
		5-shot	0.25	0.40	0.25	0.40	0.24	0.38

Table 4: Results for various open-weight LLMs on *Baccalaureate*, *Competitions* and *Synthetic*, under 0-shot and fine-tuned scenarios.

their better understanding of Romanian text compared to English-focused models, since specialization on mathematical text did not receive a particular emphasis during training. Surprisingly, we obtained that fine-tuning does not always result in improved performance. Fine-tuning improves performance on *Baccalaureate* for Qwen2-7b and Qwen2-Math-7b, while on *Competitions*, RoLlama-7b, Phi-3, Qwen2-Math-7b benefit from further fine-tuning. One possible explanation is that the solutions present in RoMath are qualitatively different (different formatting, explanation style) than solutions present in other math datasets (Cobbe et al., 2021; Hendrycks et al., 2021) and Chain-of-Thought style prompting (Wei et al., 2022). Further investigation on this effect is left as future work. In Figure 3, we show extended results per problem domain for each dataset. Qualitative examples of generated solutions are shown in the Appendix A Tables 10 and 11.

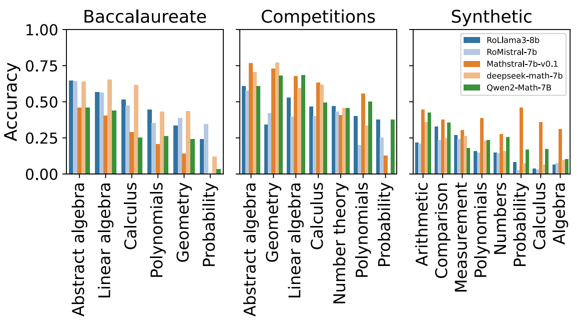


Figure 3: Performance of Romanian models and math-specialized models on each domain from each RoMath subset.

4.3 Training with Verifiable Rewards

Since a significant proportion of problems in RoMath include intermediate steps and are verifiable, we tested whether the problems are of sufficiently high quality to enable training with rewards. We adopt a part of the training procedure from Shao et al. (2024), and fine-tune two variants of the Llama3.2 (Dubey et al., 2024) (1B and 3B parameters) and Qwen2 (Yang et al., 2024) (0.5B and 1.5B) family of models. For supervised fine-tuning (SFT), we train on all problems from *Baccalaureate* and *Competitions* that contain intermediate steps to force the model to conform to the specified output format of *<raționament> [...] </raționament> <răspuns> [...] </răspuns>*.

Further, we train using GRPO (Shao et al., 2024) with 4 completions per prompt on all verifiable problems from *Baccalaureate* and *Competitions*, using only a correctness reward and a format reward. Figure 4 shows the performance on the verifiable problems from the *Baccalaureate* subset for this setting. Training with rewards reliably boosts performance compared to only supervised fine-tuning. As such, RoMath can be a useful resource for training Romanian reasoning models.

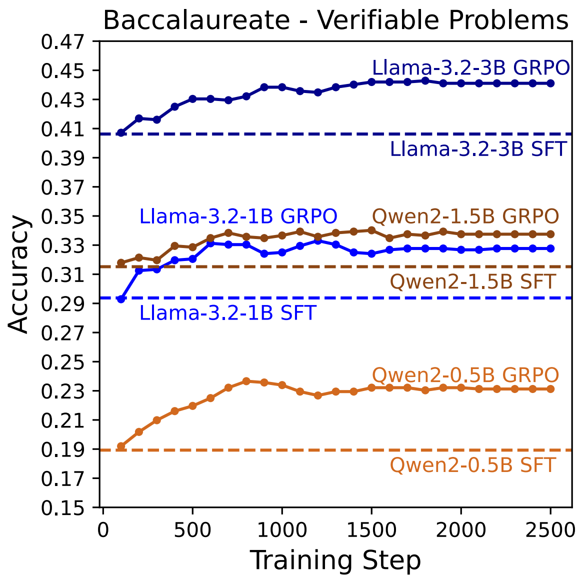


Figure 4: Performance of GRPO-trained LLama-3.2 and Qwen2 on on a subset of *Baccalaureate* that has verifiable answers.

4.4 Impact of the Judge Model

In Figure 5, we compared multiple judge models to gauge their effect on downstream performance. Based on Table 3, we used Qwen2-7B, Llama-70B and Mixtral-8x7b as judges and used them to evaluate the performance of the same Qwen2-7B, Llama-70B and Mixtral-8x7b. We chose the same judges and downstream models to check if judges prefer the output of their own model. From Figure 5 we find that judges do not have "favorites". However, we do find that, for example, in *Competitions*, where there are more proofs than in *Baccalaureate*, the Llama-70B and Mixtral-8x7b judges give higher scores on average, which might explain why results on the *Competitions* subset are higher: judges might artificially inflate results. While the differences between judges are small, there is a clear ascending trend between them.

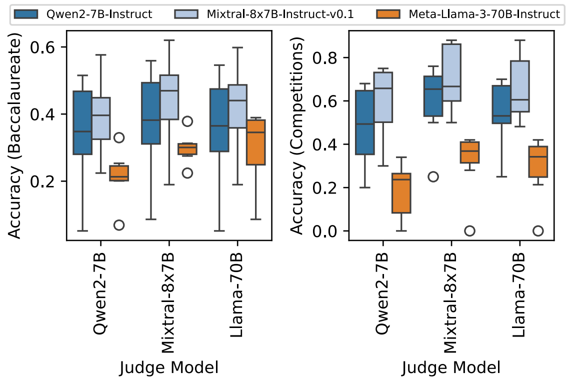


Figure 5: Performance using different judge models.

4.5 Translating Romanian Problems to English

Translating domain-specific technical language is non-trivial. Al-Tarawneh (2024) identified multiple linguistic challenges that make translation difficult. Translating mathematics is challenging due to the need for precise language, as even slight ambiguities can alter meaning. Although mathematical concepts are universal, their interpretation varies across cultures. Additionally, mathematical symbols and notations are not always standardized across languages, and mathematical terms lack direct equivalents in other languages, leading to potential confusion if not properly accounted for.

We used the NLLB (NLLB Team et al., 2022) family of models (600M, 1.3B, and 3.3B) to translate from Romanian to English the test sets for *Baccalaureate* and *Competitions*, as the models have established numerical benchmarks on Romanian to English translation. Directly translating the full problem statement and solution resulted in "gibberish" translations due to the mathematical symbols present in the text. As such, we opted to keep the LaTeX-delimited section intact and only translate the surrounding natural language. While this approach might lose some of the larger context, we found it to be the only satisfactory approach. Still, the resulting translations contain unnatural English formulations and sometimes spurious text. For instance, the problem statement *"Se consideră funcția* $f : \mathbf{R} \to \mathbf{R}$, $f(x) = e^x - x + 1$. *Să se calculeze* $\lim_{x\to 0} \frac{f(x)-f(0)}{x}$*"* is translated as *"It's considered function* $f : \mathbf{R} \to \mathbf{R}$, $f(x) = e^x - x + 1$. *Let's figure it out.* $\lim_{x\to 0} \frac{f(x)-f(0)}{x}$ *[♩ I'm not gonna let you down ♩]"*, in which the part "*[♩ I'm not gonna let you down ♩]*" is introduced spuriously by the translation model.

In Table 5, we showcase the performance of math-specialized LLMs on the English-translated version of *Baccalaureate* and *Competitions* using the different sizes of NLLB. Compared to the original Romanian text, translating severely degrades performance. We found that performance improves with the translation model size, but up to a certain point. The main point of failure is handling the math LaTeX tokens without disrupting the surrounding text. The use of an LLM for translation might be more appropriate only if its reliability and control of its output are properly established, and proper benchmarks for translation in Romanian are in place.

	Model	Translation Model	Romanian Accuracy	English-Translated Accuracy	Diff.
Baccalaureate	Qwen/Qwen2-Math-7B-Instruct	nllb-200-distilled-600M	0.32	0.04	**–0.28**
		nllb-200-1.3B		0.03	**–0.29**
		nllb-200-3.3B		0.03	**–0.29**
	deepseek-ai/deepseek-math-7b-instruct	nllb-200-distilled-600M	0.56	0.09	**–0.47**
		nllb-200-1.3B		0.05	**–0.51**
		nllb-200-3.3B		0.07	**–0.49**
	mistralai/Mathstral-7b-v0.1	nllb-200-distilled-600M	0.30	0.07	**–0.23**
		nllb-200-1.3B		0.07	**–0.23**
		nllb-200-3.3B		0.07	**–0.23**
Competitions	Qwen/Qwen2-Math-7B-Instruct	nllb-200-distilled-600M	0.55	0.17	**–0.38**
		nllb-200-1.3B		0.20	**–0.35**
		nllb-200-3.3B		0.19	**–0.36**
	deepseek-ai/deepseek-math-7b-instruct	nllb-200-distilled-600M	0.59	0.09	**–0.50**
		nllb-200-1.3B		0.12	**–0.47**
		nllb-200-3.3B		0.10	**–0.49**
	mistralai/Mathstral-7b-v0.1	nllb-200-distilled-600M	0.61	0.19	**–0.42**
		nllb-200-1.3B		0.21	**–0.40**
		nllb-200-3.3B		0.20	**–0.41**

Table 5: Results on RoMath-Baccalaureate and RoMath-Competitions for math-specific LLMs in 0-shot setting with English-translated problems. Performance drops significantly due to poor quality translations.

5 Conclusions and Future Directions

In this paper, we proposed RoMath, a benchmarking suite consisting of three datasets with mathematical problems written in Romanian: *Baccalaureate*, *Competitions* and *Synthetic*. We detailed the construction process and composition for each subset and benchmarked several open-weight LLMs under different training and evaluation scenarios. We are the first to provide quantitative results for mathematical reasoning in Romanian.

Surprisingly, we found that mathematics problems written in Romanian can be properly handled by English-centric models, providing proper solutions in Romanian. It is unclear why this occurs, especially since such models are not explicitly trained on Romanian math tokens and most models have strong language filters to train only on English. Our results suggest that such LLMs would potentially receive a passing grade (i.e., more than 50%) on the Romanian baccalaureate exam, scoring an average of $\sim$56% across all problems in *Baccalaureate*.

An important future direction is reliable automatic annotations with Chain-of-Thought (CoT) traces for multilingual reasoning problems. Our results indicate that a significant factor in improving performance in mathematical reasoning is the presence of intermediate reasoning steps in the solutions. Performance is not reliably improved by fine-tuning without CoT, and the presence of more detailed solutions enables scalable training with reinforcement learning algorithms such as GRPO (Shao et al., 2024). Currently, only a subset of RoMath contains intermediate steps for problem solutions, and further structured annotations could significantly increase the data quality.

Limitations

The main limitation of this work is the use of an external LLM as a judge to estimate solution correctness, which might skew the results and artificially inflate performance. For example, some generated solutions for proof-type problems obtain the correct final result, but the intermediate steps are incorrect. In some cases, the judge model deemed these types of solutions as correct, whereas they are not. While this is an inherent limitation in literature for mathematics datasets that contain proofs, this is currently an open problem and there are on-going efforts to formalize proof verification (Gowers et al., 2024). Furthermore, we argued that the proper way to evaluate solutions of generated proofs is by using an external proof verification tool such as Lean (de Moura et al., 2015).

Acknowledgements

This research was supported by the project "Romanian Hub for Artificial Intelligence - HRIA", Smart Growth, Digitization and Financial Instruments Program, MySMIS no. 334906.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

Alalddin Al-Tarawneh. 2024. Bridging languages and numbers: Exploring the intersection of translation studies and mathematics. *Appl. Math*, 18(3):513–519.

Reem Alghamdi, Zhenwen Liang, and Xiangliang Zhang. 2022. ArMATH: a dataset for solving Arabic math word problems. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 351–362, Marseille, France. European Language Resources Association.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

Daman Arora, Himanshu Singh, and Mausam. 2023. Have LLMs advanced enough? a challenging problem solving benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7527–7543, Singapore. Association for Computational Linguistics.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W Ayers, Dragomir Radev, and Jeremy Avigad. 2023a. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023b. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, et al. 2024. Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks. *arXiv preprint arXiv:2406.18403*.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jacob Collard, Valeria De Paiva, Brendan Fong, and Eswaran Subrahmanian. 2022. Extracting mathematical concepts from text. *arXiv preprint arXiv:2208.13830*.

Adrian Cosma, Ioan-Bogdan Iordache, and Paolo Rosso. 2024. RoCode: A dataset for measuring code intelligence from problem definitions in Romanian. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 14173–14185, Torino, Italia. ELRA and ICCL.

Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The lean theorem prover (system description). In *CADE*, volume 9195 of *Lecture Notes in Computer Science*, pages 378–388. Springer.

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. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Anca Dinu and Liviu P Dinu. 2005. On the syllabic similarities of romance languages. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 785–788. Springer.

Liviu P. Dinu and Denis Enăchescu. 2007. *On clustering Romance languages*, pages 521–528.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Adrian Marius Dumitran, Adrian Cătălin Badea, and Ștefan-Gabriel Muscalu. 2024. Evaluating the performance of large language models in competitive programming: A multi-year, multi-grade analysis. *THE 18TH INTERNATIONAL CONFERENCE ON INNOVATIONS IN INTELLIGENT SYSTEMS AND APPLICATIONS - INISTA2024*.

Meng Fang, Xiangpeng Wan, Fei Lu, Fei Xing, and Kai Zou. 2024. Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *arXiv preprint arXiv:2406.18321*.

Jöran Friberg. 1981. Methods and traditions of babylonian mathematics: Plimpton 322, pythagorean triples, and the babylonian triangle parameter equations. *Historia Mathematica*, 8(3):277–318.

Prof Sir Timothy Gowers, AlphaProof, and AlphaGeometry. 2024. Ai achieves silver-medal standard solving international mathematical olympiad problems.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Zhaoyu Li, Jialiang Sun, Logan Murphy, Qidong Su, Zenan Li, Xian Zhang, Kaiyu Yang, and Xujie Si. 2024. A survey on deep learning for theorem proving. *arXiv preprint arXiv:2404.09939*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Chen Cecilia Liu, Fajri Koto, Timothy Baldwin, and Iryna Gurevych. 2023. Are multilingual llms culturally-diverse reasoners? an investigation into multicultural proverbs and sayings. *arXiv preprint arXiv:2309.08591*.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*.

Mihai Masala, Denis Ilie-Ablachim, Alexandru Dima, Dragos Georgian Corlatescu, Miruna-Andreea Zavelca, Ovio Olaru, Simina-Maria Terian, Andrei Terian, Marius Leordeanu, Horia Velicu, Marius Popescu, Mihai Dascalu, and Traian Rebedea. 2024. "Vorbești Românește?" A Recipe to Train Powerful Romanian LLMs with English Instructions. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11632–11647, Miami, Florida, USA. ACL.

Mathpix. 2024. Ai-powered document automation.

Jordan Meadows, Marco Valentino, Damien Teney, and Andre Freitas. 2023. A symbolic framework for systematic evaluation of mathematical reasoning with transformers. *arXiv preprint arXiv:2305.12563*.

Mistral AI. 2024. `https://mistral.ai/news/mathstral/`. Accessed: 2024-09-13.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, et al. 2022. No language left behind: Scaling human-centered machine translation.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2023. Openwebmath: An open dataset of high-quality mathematical web text.

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*.

G. Polya. 1971. *How to Solve It*. Princeton University Press.

Michael Rescorla. 2024. The Language of Thought Hypothesis. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*, Summer 2024 edition. Metaphysics Research Lab, Stanford University.

Tomohiro Sawada, Daniel Paleka, Alexander Havrilla, Pranav Tadepalli, Paula Vidas, Alexander Kranias, John J Nay, Kshitij Gupta, and Aran Komatsuzaki. 2023. Arb: Advanced reasoning benchmark for large language models. *arXiv preprint arXiv:2307.13692*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482.

Wenxuan Wang, Wenxiang Jiao, Jingyuan Huang, Ruyi Dai, Jen-tse Huang, Zhaopeng Tu, and Michael R Lyu. 2023. Not all countries celebrate thanksgiving: On the cultural dominance in large language models. *arXiv preprint arXiv:2310.12481*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. 2023. Cmath: Can your language model pass chinese elementary school math test? *arXiv preprint arXiv:2306.16636*.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. 2024. Do llamas work in english? on the latent language of multilingual transformers. *arXiv preprint arXiv:2402.10588*.

An Yang, Baosong Yang, Binyuan Hui, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. Ape210k: A large-scale and template-rich dataset of math word problems. *arXiv preprint arXiv:2009.11506*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

A Appendix

Given the following mathematics problems in Romanian formatted in MathPix markdown, make a JSON with subject and solution pairs, removing unnecessary boilerplate and extra problem identifiers. The JSON must contain the full problem definition and subject number (e.g. subject 1b). Each sub-question must contain the whole problem definition for completeness. Each subject must be self-contained. Do not output anything else besides the required JSON. Do not modify the latex describing the mathematical formulas.
Example (truncated): "

PROBLEMS:

Se consideră matricea $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ şi şirul $(F_n)_{n\geq 0}$ definit prin relația de recurență $F_{n+1} = F_n + F_{n-1}, n \in \mathbf{N}^*$, cu $F_0 = 0, F_1 = 1$.

a) Să se calculeze determinantul şi rangul matricei A.

b) Să se calculeze F_2 şi F_3.

SOLUTIONS:
a) $\det A = -1 \neq 0 \Rightarrow \operatorname{rang} A = 2$; b) $F_2 = 1, F_3 = 2$.
"

Example Output JSON:
[{
"subject": "1a",
"definition": "Se consideră matricea $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ şi şirul $(F_n)_{n\geq 0}$ definit prin relația de recurență $F_{n+1} = F_n + F_{n-1}, n \in \mathbf{N}^*$, cu $F_0 = 0, F_1 = 1$. Să se calculeze determinantul şi rangul matricei A.",
"solution": "$\det A = -1 \neq 0 \Rightarrow \operatorname{rang} A = 2$"
},
{
"subject": "1b",
"definition": "Se consideră matricea $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ şi şirul $(F_n)_{n\geq 0}$ definit prin relația de recurență $F_{n+1} = F_n + F_{n-1}, n \in \mathbf{N}^*$, cu $F_0 = 0, F_1 = 1$. Să se calculeze F_2 şi F_3.",
"solution": "$F_2 = 1, F_3 = 2$"
}
]
In this example, each sub-problem is self-contained and is paired with the appropriate solution. The sub-problem identifiers (e.g., "a)" and "b)") are stripped. The latex markdown is left intact.

Real Input:

Table 6: Claude 3 Sonnet prompt to format raw Markdown into structured JSON.

```
{
"role": "system",
"content": """Ești un student olimpic la matematică care a participat și câștigat multiple concursuri internaționale de matematică. Rolul tău este să rezolvi probleme de matematică de liceu și să oferi soluții complete și corecte. Problemele care necesită demonstrații trebuie rezolvate complet cu toți pașii intermediari necesari. Problemele care au un singur raspuns final trebuie furnizate într-un format încadrat (‘\boxed‘). Matematica trebuie scrisă în format LaTeX pentru a asigura claritatea și precizia soluțiilor. Textul in format LaTeX trebuie delimitat folosind simbolurile ‘ \(‘ și ‘\)‘. Rezolvările incomplete sau incorecte vor fi evaluate cu scoruri mai mici. Asigură-te că răspunsurile sunt concise, fără prea multe explicații inutile.""",
},
# add few shot examples here: User => Problem statement, Assistant => Solution
{
"role": "user",
"content": """Care este rezolvarea următoarei probleme?

{problem_statement}""",
}
```

Table 7: Romanian prediction prompt.

```
{ "role": "system",
"content": """Asumă-ți rolul unui profesor de matematică responsabil cu evaluarea răspunsurilor studenților pentru o problemă de matematică în raport cu soluțiile corecte furnizate. Soluțiile pot include demonstrații, valori exacte, răspunsuri cu alegere multiplă sau aproximări numerice.

## Criterii de Evaluare:
1. **Echivalență Matematică**: Evaluează răspunsurile pe baza echivalenței matematice, nu doar a preciziei numerice. Verifică dacă diferite expresii algebrice sau simbolice sunt echivalente. Asigură-te că sunt echivalențe precum \( \frac{\sqrt{6}-\sqrt{2}}{2} \) fiind echivalent cu \( \sqrt{2 - \sqrt{3}} \).
2. **Scor**: Atribuie un scor de ’1’ pentru orice răspuns care se potrivește sau este echivalent cu soluția furnizată, fie că este o valoare exactă, o variantă de răspuns (de exemplu, A, B, C) sau o aproximare numerică corect rotunjită. Atribuie un scor de ’0’ pentru răspunsuri incorecte. Nu furniza niciun fel de explicație.
3. **Tratarea Alegerii Multiple**: Dacă soluția furnizată este o variantă de răspuns (de exemplu, A, B, C, D, E, F) și studentul identifică această alegere corect, trateaz-o ca fiind corectă. Dacă soluția este o valoare exactă și studentul furnizează alegerea corespunzătoare care reflectă corect această valoare în conformitate cu contextul problemei, tratează-o de asemenea ca fiind corectă.
4. **Echivalență Numerică**: Tratează răspunsurile numerice ca fiind echivalente dacă sunt corecte cu cel puțin două zecimale sau mai mult, în funcție de precizia furnizată în soluție. De exemplu, atât 0.913, cât și 0.91 ar trebui acceptate dacă soluția este exactă cu două zecimale.
5. **Identități Algebrice și Simbolice**: Recunoaște și acceptă forme algebrice echivalente, cum ar fi \( \sin^2(x) + \cos^2(x) = 1 \) sau \( e^{i\pi} + 1 = 0 \), ca fiind corecte.
6. **Forme Trigonometrice și Logaritmice**: Acceptă expresii trigonometrice și logaritmice echivalente, recunoscând identități și transformări care ar putea modifica forma, dar nu și valoarea.
7. **Demonstrații Matematice**: Evaluează demonstrațiile matematice pe baza corectitudinii și a logicii, nu a stilului sau a formei. Asigură-te că demonstrațiile sunt complete și corecte, chiar dacă sunt prezentate într-un mod diferit de soluția furnizată.

## Formatul Așteptat al Răspunsului: Prezintă răspunsul final cu un scor doar de ’0’ sau ’1’, unde ’0’ semnifică o soluție greșită, iar ’1’ semnifică o soluție corectă. Nu include nicio altă informație sau explicații suplimentare în răspuns.

Problema de matematică este:
{question}.

Soluția corectă din baremul de corectare este:
{true}.

Te rog să evaluezi soluția studentului cu precizie pentru a asigura o evaluare exactă și corectă."""
},
{
"role": "user", "content": "Soluția studentului este {prediction}. Furnizează un doar scor de ’0’ sau ’1’, unde ’0’ semnifică o soluție greșită, iar ’1’ semnifică o soluție corectă. Bazează-ți evaluarea pe criteriile de evaluare furnizate si pe soluția corecta din barem.",
}
```

Table 8: Romanian judge prompt.

```
{
"role": "system",
"content": """Assume the role of a math teacher responsible for evaluating student responses for a math problem against the provided correct solutions. Solutions may include proofs, exact values, multiple-choice answers, or numerical approximations.

## Evaluation Criteria:
1. **Mathematical Equivalence**: Evaluate answers based on mathematical equivalence, not just numerical accuracy. Check if different algebraic or symbolic expressions are equivalent. Ensure that there are equivalences such as \( \frac{\sqrt{6}-\sqrt{2}}{2} \) being equivalent to \( \sqrt{2 - \sqrt{3}} \).
2. **Scoring**: Assign a score of ’1’ for any answer that matches or is equivalent to the provided solution, whether it is an exact value, a choice label (e.g., A, B, C), or a correctly rounded numerical approximation. Assign a score of ’0’ for incorrect answers. Do not provide any explanatory feedback.
3. **Handling Multiple Choices**: If the solution provided is a choice (e.g., A, B, C, D, E, F) and the student identifies this choice correctly, treat it as correct. If the solution is an exact value and the student provides the corresponding choice that reflects this value correctly according to the problem’s context, also treat it as correct.
4. **Numerical Equivalence**: Treat numerical answers as equivalent if they are correct to at least two decimal places or more, depending on the precision provided in the solution. For instance, both 0.913 and 0.91 should be accepted if the solution is accurate within two decimal places.
5. **Symbolic and Algebraic Identities**: Recognize and accept equivalent algebraic forms, such as \( \sin^2(x) + \cos^2(x) = 1 \) or \( e^{i\pi} + 1 = 0 \), as correct.
6. **Trigonometric and Logarithmic Forms**: Accept equivalent trigonometric and logarithmic expressions, acknowledging identities and transformations that might alter the form but not the value.
7. **Mathematical Proofs**: Evaluate mathematical proofs based on correctness and logic, not style or form. Ensure proofs are complete and correct, even if presented differently from the provided solution.
## Expected Output Format: Present your final answer with a score of ’0’ or ’1’ only, where ’0’ signifies an incorrect solution, and ’1’ signifies a correct solution. Do not include any additional information or feedback in your response.

The math problem is:
{question}.

The correct solution from the answer key is:
{true}.

Please evaluate the student’s response with precision to ensure accurate and fair grading."""
},
{
"role": "user",
"content": "The student answer is prediction. Provide a score of ’0’ or ’1’ only, where ’0’ signifies an incorrect solution, and ’1’ signifies a correct solution. Base your evaluation on the provided evaluation criteria and the correct solution from the answer key.",
}
```

Table 9: English judge prompt.

RoMath-Baccalaureate

deepseek-ai/deepseek-math-7b-instruct

***Problem Statement*: Să se calculeze modulul numărului complex $\sqrt{3}+i$.**
***Solution*:** 2
***Generated Solution*:** Pentru a calcula modulul unui număr complex, folosim formula $|z| = \sqrt{a^2+b^2}$, unde $z = a + bi$.
În cazul nostru, $a = \sqrt{3}$ și $b = 1$.
Aplicând formula, obținem:
$|z| = \sqrt{(\sqrt{3})^2 + 1^2} = \sqrt{3+1} = \sqrt{4} = 2.$
Deci, modulul numărului complex $\sqrt{3}+i$ este 2.
Răspuns: $\boxed{2}$.

OpenLLM-Ro/RoLlama3-8b-Instruct

***Problem Statement*:** Să se demonstreze că vectorii $\vec{u} = -2\vec{i} + 3\vec{j}$ și $\vec{v} = 3\vec{i} + 2\vec{j}$ sunt perpendiculari.
***Solution*:** $\text{Din } \vec{a} \cdot \vec{b} = -2 \cdot 3 + 3 \cdot 2 = 0 \Rightarrow \vec{a} \perp \vec{b}$
***Generated Solution*:** Pentru a demonstra că vectorii $\vec{u} = -2\vec{i} + 3\vec{j}$ și $\vec{v} = 3\vec{i} + 2\vec{j}$ sunt perpendiculari, trebuie să arătăm că produsul lor punct este egal cu 0.
Produsul punct al celor doi vectori $\vec{u}$ și $\vec{v}$ este dat de:
$\vec{u} \cdot \vec{v} = (-2\vec{i} + 3\vec{j}) \cdot (3\vec{i} + 2\vec{j})$
Acum, putem calcula produsul punct:
$\vec{u} \cdot \vec{v} = (-2)(3) + (3)(2) = -6 + 6 = 0$
Deoarece produsul punct al celor doi vectori este egal cu 0, vectorii $\vec{u}$ și $\vec{v}$ sunt perpendiculari.

RoMath-Competitions

deepseek-ai/deepseek-math-7b-instruct

***Problem Statement*:** Într-o clasă sunt 22 de elevi, dintre care 12 sunt fete. Să se determine în câte moduri se poate alege un comitet al clasei format din 3 fete și 2 băieți.
***Solution*:** Numărul băieților este $22 - 10 = 12$. Fetele pot fi alese în $\mathrm{C}_{12}^{3} = 220$ moduri. Băieții pot fi aleși în $C_{10}^{2} = 45$ moduri. Comitetul format din 3 fete și 2 băieți dintre cei 22 de elevi ai clasei poate fi format în $\mathrm{C}_{12}^{3} \cdot \mathrm{C}_{10}^{2} = 9900$ moduri.
***Generated Solution*:** Pentru a rezolva această problemă, trebuie să utilizăm formula combinării. Combinarea este o modalitate de a alege un set de elemente dintr-o mulțime mai mare, fără a uita la ordinea elementelor.
În acest caz, trebuie să alegem 3 fete din 12 și 2 băieți din 10 (deoarece 22 - 12 = 10).
Formula combinării este:

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

unde n este numărul total de elemente, k este numărul de elemente ce trebuie alese, iar ""!"" semnifică factorialul unui număr.
Pentru a calcula numărul de moduri în care putem alege 3 fete din 12, aplicăm formula:

$$C(12,3) = \frac{12!}{3!(12-3)!} = \frac{12!}{3!9!} = \frac{12 \times 11 \times 10}{3 \times 2 \times 1} = 220$$

Pentru a calcula numărul de moduri în care putem alege 2 băieți din 10, aplicăm formula:

$$C(10,2) = \frac{10!}{2!(10-2)!} = \frac{10!}{2!8!} = \frac{10 \times 9}{2 \times 1} = 45$$

Pentru a calcula numărul total de moduri în care putem forma comitetul, înmulțim cele două rezultate:

$$220 \times 45 = 9900$$

Deci, există 9900 de moduri în care se poate forma comitetul.
Răspunsul final este $\boxed{9900}$.

OpenLLM-Ro/RoLlama3-8b-Instruct

***Problem Statement*:** Un grădinar a plantat într-una dintre grădinile pe care le îngrijește parcele cu tufe de trandafir din soiuri distincte, astfel încât fiecare parcelă conține trandafiri dintr-un alt soi. Într-o zi are de realizat un aranjament floral din acești trandafiri. Analizează tufele și procedează în felul următor: din prima parcelă taie trei trandafiri, din cea de a doua taie de două ori mai mulți decât din prima ș.a.m.d., tăind dintr-o parcelă de două ori mai mulți trandafiri decât din parcela precedentă. Care este numărul minim de trandafiri pe care trebuie să-l aibă parcela a IX-a pentru a putea fi folosită la realizarea aranjamentului floral?
***Solution*:** Parcela a IX a trebuie să aibă cel puțin: $3 \cdot 2^8 = 3 \cdot 256 = 768$
***Generated Solution*:** Să notăm numărul de trandafiri din fiecare parcelă ca P1, P2, P3,..., P9. Conform problemei, numărul de trandafiri din fiecare parcelă este dat de:
P1 = 3
P2 = 2 * P1 = 2 * 3 = 6
P3 = 2 * P2 = 2 * 6 = 12
P4 = 2 * P3 = 2 * 12 = 24
P5 = 2 * P4 = 2 * 24 = 48
P6 = 2 * P5 = 2 * 48 = 96
P7 = 2 * P6 = 2 * 96 = 192
P8 = 2 * P7 = 2 * 192 = 384
P9 = 2 * P8 = 2 * 384 = 768
Deci, numărul minim de trandafiri din parcela a IX-a este 768.

Table 10: Qualitative examples of correct zero-shot predictions for RoMath-Baccalaureate and RoMath-Competitions.

RoMath-Baccalaureate

deepseek-ai/deepseek-math-7b-instruct

Problem Statement: Să se rezolve în mulțimea numerelor reale ecuația $2^x = 4^x$.
Solution: x=0
Generated Solution: Pentru a rezolva ecuația $2^x = 4^x$, putem scrie 4^x ca $(2^2)^x$, adică 2^{2x}.
Astfel, ecuația devine $2^x = 2^{2x}$.
Prin urmare, putem scrie $x = 2x$.
Împărțind ambii membri ai ecuației cu x, obținem $1 = 2$.
Această ecuație nu are soluții în mulțimea numerelor reale, deoarece 1 nu este egal cu 2.
Deci, ecuația $2^x = 4^x$ nu are soluții în mulțimea numerelor reale.
Răspuns: $\boxed{\text{Nu are soluții}}$

OpenLLM-Ro/RoLlama3-8b-Instruct

Problem Statement: Să se calculeze determinantul $\begin{vmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 2 & 4 & 6 \end{vmatrix}$.
Solution: 0
Generated Solution: Pentru a calcula determinantul matricei date, putem folosi metoda de expansiune prin rânduri. Să calculăm determinantul:

$$\begin{vmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 2 & 4 & 6 \end{vmatrix}$$

În primul rând, putem elimina primul rând și primul element din al doilea rând:

$$\begin{vmatrix} 9 & 11 \\ 2 & 4 \end{vmatrix}$$

Acum, putem calcula determinantul matricei 2x2 rămase:

$$\begin{vmatrix} 9 & 11 \\ 2 & 4 \end{vmatrix} = 9 \cdot 4 - 11 \cdot 2 = 36 - 22 = 14$$

Deci, determinantul matricei date este 14.

RoMath-Competitions

deepseek-ai/deepseek-math-7b-instruct

Problem Statement: Într-un plan, raportat la reperul ortogonal de axe de coordonate $(x\mathrm{xOy})$ se dau punctele: $\mathrm{A}(0; 6); \mathrm{B}(5; 4); \mathrm{C}(-1; 4)$. Pentru $\mathrm{a} = 5$ să se scrie ecuația medianei corespunzătoare laturii BC.
Solution: Determină coordonatele mijlocului lui $[\mathrm{BC}]$ notat cu $\mathrm{M} : \mathrm{x_M} = \frac{5-1}{2} = 2, \mathrm{y_M} = \frac{4+4}{2} = 4 \Leftrightarrow \mathrm{M}(2, 4)$ Ecuația dreptei (AM): $\begin{vmatrix} x & y & 1 \\ 0 & 6 & 1 \\ 2 & 4 & 1 \end{vmatrix} = 0 \Rightarrow AM : x + y - 6 = 0$.
Generated Solution: Pentru a scrie ecuația medianei corespunzătoare laturii BC, trebuie să găsim mijlocul segmentului BC, notat M.
Mijlocul unui segment este dat de media aritmetică a coordonatelor punctelor de capăt ale segmentului. Astfel, coordonatele punctului M sunt:
M(xM; yM) = ((xB + xC) / 2; (yB + yC) / 2)
M(-1 + 5 / 2; 4 + 4 / 2) = M(2; 4)
Mediana AM este dreapta care unește punctul A cu mijlocul M al segmentului BC.
Ecuația unei drepte în plan, cunoscând coordonatele a două puncte A(x1; y1) și B(x2; y2), este: (y - y1) / (y2 - y1) = (x - x1) / (x2 - x1)
Aplicând formula pentru mediana AM, avem: (y - 6) / (4 - 6) = (x - 0) / (2 - 0)
Simplificând, obținem: (y - 6) / (-2) = x / 2
Ecuația medianei AM este: y - 6 = -2x sau 2x + y - 6 = 0
Pentru a = 5, ecuația medianei corespunzătoare laturii BC este: 2x + y - 6 = 0

OpenLLM-Ro/RoLlama3-8b-Instruct

Problem Statement: În raport cu un reper cartezian xOy, considerăm un purice P care sare doar în puncte având ambele coordonate întregi. La o săritură, puricele se deplasează doar pe verticală sau pe orizontală. După o săritură nu este obligatoriu să schimbe direcția de deplasare, dar respectă următoarea regulă: sare 3 unități, apoi 2 unități, 3 unități, 2 unitàti etc. De exemplu, dacă M se află inițial în punctul $(1, 2)$, la prima mutare ar putea fi în punctul $(1, 5)$, iar apoi în $(1, 7)$ dacă păstrează direcția, sau în (3,5), dacă își schimbă direcția. Dacă P se află inițial în origine, demonstrați că poate ajunge în punctul $(2013, 0)$.
Solution: Puricele se poate deplasa doar orizontal, astfel: după două sărituri ajunge în punctul $(5, 0)$, după patru ajunge în $(10, 0)$ și, tot așa, după 804 sărituri ajunge în $(2010, 0)$. La următoarea săritură va ajunge în $(2013, 0)$.
Generated Solution: Pentru a demonstra că puricele poate ajunge în punctul $(2013, 0)$, vom arăta că poate ajunge în orice punct cu coordonate întregi prin repetarea ciclului de deplasare. Să notăm deplasarea verticală ca v și deplasarea orizontală ca h. Ciclul de deplasare este dat de secvența $(v, h, v, h, v, h, \ldots)$, unde $v = 3$ și $h = 2$. Aceasta înseamnă că puricele se deplasează în sus 3 unități, apoi în dreapta 2 unități, apoi în sus 3 unități, apoi în dreapta 2 unități, și așa mai departe.
Acum, să luăm în considerare punctul $(2013, 0)$. Deoarece puricele poate sări doar în puncte cu coordonate întregi, trebuie să găsim o secvență de deplasări care să conducă la acest punct. Putem observa că ciclul de deplasare dat poate fi repetat de 2013 ori pentru a ajunge la punctul $(2013, 0)$.
Pentru a vedea acest lucru, să luăm în considerare deplasările necesare pentru a ajunge la punctul $(2013, 0)$ din origine:
1. Deplasare verticală de 3 unități (de la $(0, 0)$ la $(0, 3)$).
2. Deplasare orizontală de 2 unități (de la $(0, 3)$ la $(2, 3)$).
3. Deplasare verticală de 3 unități (de la $(2, 3)$ la $(2, 6)$).
4. Deplasare orizontală de 2 unități (de la $(2, 6)$ la $(4, 6)$).
5. Deplasare verticală de 3 unități (de la $(4, 6)$ la $(4, 9)$).
6. Deplasare orizontală de 2 unități (de la $(4, 9)$ la $(6, 9)$).
7. Deplasare verticală de 3 unități (de la $(6, 9)$ la $(6, 12)$).
8. Deplasare orizontală de 2 unități (de la $(6, 12)$ la $(8, 12)$).
9. Deplasare verticală de 3 unități (de la $(8, 12)$ la $(8, 15)$).
10. Deplasare orizontală de 2 unități (de la $(8, 15)$ la $(10, 15)$).
Putem continua acest proces până când ajungem la punctul $(2013, 0)$. Deoarece ciclul de deplasare poate fi repetat de 2013 ori, puricele poate ajunge în punctul $(2013, 0)$ după efectuarea ciclului de deplasare de 2013 ori.

Table 11: Qualitative examples of incorrect zero-shot predictions for RoMath-Baccalaureate and RoMath-Competitions.

Into The Limits of Logic: Alignment Methods for Formal Logical Reasoning

Francisco F. López-Ponce[1,2], Gemma Bel-Enguix [2]
[1]Posgrado en Ciencias e Ingeniería de la Computación - UNAM
[2]Grupo de Ingeniería Lingüística - UNAM
francisco.lopez.ponce@ciencias.unam.mx, gbele@iingen.unam.mx

Abstract

We implement Large Language Model Alignment algorithms to formal logic reasoning tasks involving natural-language (NL) to first-order logic (FOL) translation, formal logic inference, and premise retranslation. These methodologies were implemented using task-specific preference datasets created based on the FOLIO datasets and LLM generations. Alignment was based on DPO, this algorithm was implemented and tested on off-the-shelf and pre-aligned models, showing promising results for higher quality NL-FOL parsing, as well as general alignment strategies[1]. In addition, we introduce a new similarity metric ($LogicSim$) between LLM-generated responses and gold standard values, that measures logic-relevant information such as premise count and overlap between answers and expands evaluation of NL-FOL translation pipelines[2]. Our results show that LLMs still struggle with logical inference, however alignment benefits semantic parsing and retranslation of results from formal logic to natural language.

1 Introduction

Reasoning using a formal logic language is one of the basis of mathematical thinking. Being able to abstract a problem in natural language and express the distinct variables and relationships between them in logical terms helps mitigating ambiguity and unclear relationships. Using these formal representations, a step-by-step inference procedure can be carried out in order to obtain a logically valid conclusion of the presented premises. This type of reasoning is crucial for, not only mathematics but, any discipline in need of an explainable decision making process.

State-of-the-art Large Language Models (LLMs) exhibit human-like reasoning capabilities for various tasks such as coding, general academic examination, and reading comprehension (OpenAI, 2023; Anthropic, 2024). However, formal logic and mathematical reasoning has proven to be an area of expertise where LLMs consistently underperform: the Claude 3 series of models barely reach a 42% Accuracy in the AMC 12 (Mathematical Association of America, 2025), and a 61% on the MATH dataset (Hendrycks et al., 2021), even with such scores Claude outperforms models like GPT 4. Given the connection between logic, mathematics, and human cognitive processes (Yang et al., 2024b), improving an LLM's performance in this area is an open research problem. Not only is this interesting as a stand-alone problem, it has very positive implications for the explainability of LLMs. Having a model that can correctly infer and explain step-by-step said process would benefit most current uses of LLMs.

In this article we focus on an end-to-end inference process divided into three main steps. Given a set of premises in natural language (NL) the first step is a translation of the premises from natural language to first-order logic (FOL). The second step is an LLM-based inference procedure based on the FOL representations. The final step corresponds to a retranslation of the conclusion from FOL to natural language. For each step we implement an LLM Alignment methodology focused on the corresponding task in order to improve a model's performance, as well as a corresponding evaluation.

LLM Alignment methods are post-training strategies that modify a model's internal weights in order to generate text that caters with human selected responses for a wide range of downstream tasks. Alignment strategies, based on reinforcement learning with human feedback (Christiano et al., 2017), were originally implemented for auto-

[1]Datasets and models can be found freely on HuggingFace: https://huggingface.co/Kurosawama

[2]Code can be accessed via this paper's GitHub: https://github.com/Kurocaguama/Into-The-Limits-of-Logic

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 112–123
November 9, 2025 ©2025 Association for Computational Linguistics

matic summarization (Stiennon et al., 2020). However, recent research uses these strategies to adjust an LLM's behavior to generate safe and useful responses (Ouyang et al., 2022; Bai et al., 2022; Ji et al., 2025). Interestingly, alignment methods have been shown to improve an LLM's performance in tasks outside safety and harmfulness, making them an integral part of post training for newer releases of models (DeepSeek-AI et al., 2025; OpenAI, 2025).

Our approach differs from similar methodologies that work with the same NL-FOL workflow by omitting the dependency on prompting, the use of an external solver, and self-verification. By integrating translation and inference capabilities into a model via alignment, we obtain a robust model capable of solving our problem on its own without the need of external tools. Additionally, we present a novelty metric that measures logical similarity between two sets of premises based on information concerning predicates and logical connectors. This metric allows us to finely evaluate LLMs during the workflow, expanding our focus from the single truth value that NL-FOL workflows are usually evaluated by.

We tested our methodology around the FOLIO dataset (Han et al., 2024), a human created and annotated dataset focused on natural language to first-order logic translation and inference, that is easily adaptable to our three step pipeline. Evaluation was in accordance to FOLIO and step-specific metrics. Our results show that off-the-shelf LLMs can be easily adjusted to become efficient semantic parsers with a limited amount of information, even being able to follow particular prompt-structures aside from the logical benefit. Pre-aligned models perform decently without any logic-based alignment, yet their performance falters after our alignment strategy is applied, suggesting that our methodology needs to be polished in order to work as an additional layer of post-training.

2 Related Work

2.1 LLMs for Logic-Based Reasoning

Early work in this area, such as ProofWriter (Tafjord et al., 2021), tested how well LLMs could reproduce proof generation based on given facts and rules, opting for a generative strategy over classification systems based on pretrained models such as PRover (Saha et al., 2020).

Recent work has covered a wide variety of generative strategies similar to ProofWriter. Pan et al. (2023)'s Logic-LM and Olausson et al. (2023)'s LINC are an example of tool-augmented systems that test an identical end-to-end problem as ourselves. A key difference is that they incorporate the use of an external solver for the inference and retranslation part of the problem, an addition that enables both systems to outperform classic prompting strategies like Chain-of-Thought (Wei et al., 2023) reasoning. Further research has improved tool-augmented systems (Raza and Milic-Frayling, 2025; Kirtania et al., 2024), even surpassing LogicLM's and LINC's results. However, the addition of the external solver reduces the impact of the LLM within the logic pipeline, limiting the influence to the symbolic formalization of the pipeline. Yao et al. (2023) endows an LLM with a search function over possible answers in order to emulate a tree search. This removes the need of the external solver but still requires certain calculations to be carried out independently of the LLM.

Most LLM-only approaches rely on prompting and in-context learning, often modifying the structure of the prompt to carry out procedures like abstracting, formalizing, and explaining before answering (Ranaldi et al., 2025), formulating the solution as a Python function (Lyu et al., 2023), or giving the prompt a task-specific solution structure (Zhou et al., 2024). An extension of these approaches that is highly similar to our work, is the use of LLM Alignment methods in order to optimize a task-oriented Supervised Fine-Tuning (Ranaldi and Freitas, 2024). A slight difference however, is that this work focuses on general reasoning rather than formal logic.

2.2 Alignment Methodologies

Initial works in LLM Alignment taught models how to summarize texts (Stiennon et al., 2020) and how to behave in terms of safety and usefulness (Ouyang et al., 2022) using Supervised Fine-Tuning (SFT) to obtain a reference model for sampling, and PPO (Schulman et al., 2017) as a training strategy with another LLM. However, PPO is an online algorithm that uses two LLMs during training, meaning that it's a resource heavy option that now serves more as a baseline comparison rather than a widely used strategy. Alternative algorithms have been developed that solve this problem, particularly Direct Preference Optimization (DPO) (Rafailov et al., 2023). This algorithm avoids fine-tuning a reference model as well as the sampling from

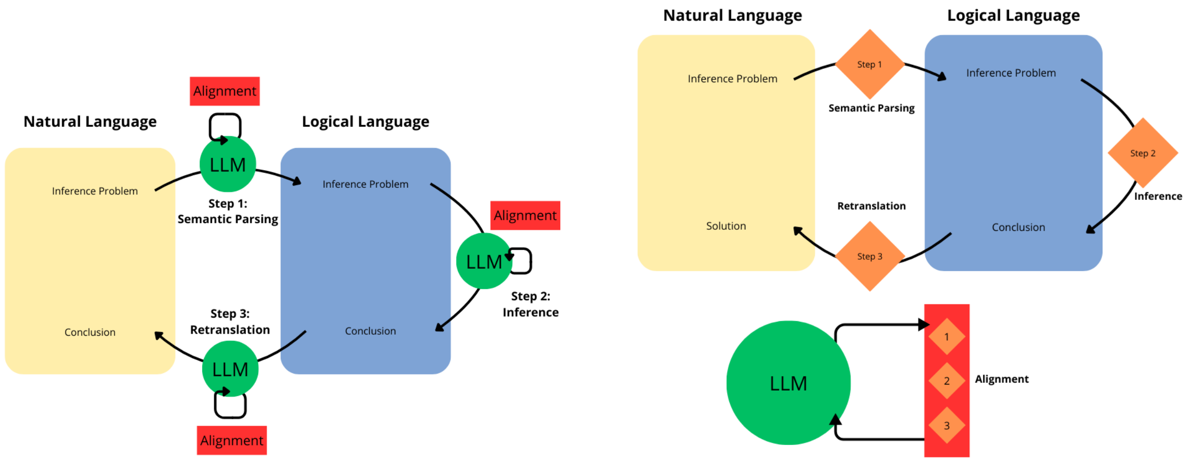


Figure 1: Step-Independent Alignment (left) and Mixture-of-Steps Alignment (right).

said model, it does so by working on a preference dataset that compares pairs of responses to a given prompt. DPO is a widely used algorithm for alignment in state-of-the-art LLMs (Grattafiori et al., 2024).

Further research regarding alignment algorithms has been carried out. The Group Relative Policy Optimization algorithm (GRPO) (Shao et al., 2024) is a memory-optimized version of PPO that avoids the reference model, and instead shows explicitly which responses are preferred by giving a prompt and a "correct" completion of said prompt.

Mathematically speaking, each strategy has its strengths and weaknesses that make it an adequate candidate for alignment (Xu et al., 2024). However we can't just rely solely on mathematical intuition given that we're working with text data as well as a concrete problem that can't be solely encapsulated with loss functions. For said reason we opt for DPO due to the nature of preference datasets used during training. This type of datasets enables a model to compare different answers, analyze differences between them, and consider the preference score in order to adequately adjust the model's response. In logical and mathematical reasoning there isn't always a single correct way to derive a proof, or to semantically parse a set of premises. We believe that by having a dynamic set of scores that measure correctness, a model should be able to efficiently adapt to varying sets of premises, inference steps, and lexical themes.

2.3 Logic and Mathematics Evaluation

LLM logical evaluation has a plethora of datasets to work with. However, not every dataset centered on logical reasoning actually evaluates formal logical language reasoning. Datasets like LogiQA (Liu et al., 2020) and LogicNLI (Tian et al., 2021) are good resources for natural language reasoning, however there's no logical formalization of the premises and answers within those datasets.

Newer benchmarks such as FOLIO (Han et al., 2024), LogicBench (Parmar et al., 2024), or MALLS (Yang et al., 2024a), deal with the formalization of the premises in first-order logic. In particular, FOLIO is an expert-written and human-reviewed dataset (contrary to the other two), that covers each step of an end-to-end pipeline for logic-inference.

3 Experiments

In this section we describe our end-to-end workflow with a working example, the creation of the preference datasets used for alignment, selected LLM checkpoints, and experimental setup.

3.1 Problem Definition

As previously mentioned, our inference procedure is divided into three separate steps, each evaluated independently to determine weak points throughout the end-to-end workflow.

The first step of our three step inference process is the **translation** of a set of premises from natural language to first order logic. This task is also referred to as semantic parsing, and is shared across many logic-based reasoning systems (Pan et al., 2023; Olausson et al., 2023; Raza and Milic-Frayling, 2025). The second step is the **inference** procedure based solely on the premises in their FOL syntax. The LLM should use logically valid inference steps[3] in order to obtain a unique conclusion, expressed in FOL syntax, to the problem formulation. The third and final step corresponds to a **retranslation** of the conclusion to natural language. An example of the whole process can be seen in table 1. The example is extracted from the validation set of the FOLIO dataset.

3.2 Alignment

In order to train an LLM in each task an alignment strategy based on the corresponding step is implemented. Two variations of the alignment procedure are possible: *Step-Independent Alignment* (training with only a single step of the pipeline) and *Mixture-of-Steps Alignment* (training using the full pipeline). Step-Independent Alignment generates three distinctly aligned models, one for each corresponding step. On the other hand, Mixture-of-Steps returns a single model aligned to all of the steps. Figure 1 shows a diagram of the end-to-end inference process and the steps where alignment is performed. Due to the sparse amount of data available for training (1000 data instances at most per step), Step-Independent Alignment is not carried out in our experiments. The implementations talked about in the remainder of the paper describe training and performance using the Mixture-of-Steps methodology.

We implement alignment based on DPO (Rafailov et al., 2023) due to the advantages this algorithm presents over classic SFT + RLHF approaches (Ouyang et al., 2022), particularly in regards with the datasets needed implement the algorithm. DPO uses a preference dataset for training, each entry is comprised of four columns, two of which contain chosen and rejected generations, and two that contain a score that measures a numerical preference over each pair of responses. The chosen and rejected columns share the same input prompt, it's the LLM response that varies.

We created a preference dataset for each step of the end-to-end procedure. Each dataset is model specific meaning that the responses of one model's checkpoint don't affect the behavior of other models. This allows us to evaluate various aspects of LLM behavior such as how susceptible each individual model is to our methodology, how much of an improvement is shown between aligned and vanilla models, parameter size dependency and more. This segmentation between steps generates three datasets per LLM checkpoint used, however combining them into a unique dataset is a straight forward procedure.

[3]The same ones used for the FOLIO dataset.

3.3 Preference Dataset Creation

To create a single entry of the dataset we consider a prompt x, based on which we need to obtain two answers, y_1, y_2 (chosen and rejected), and two scores, s_1, s_2, that serve as preferences for the LLM. The chosen column contains the pair (x, y_1), while the rejected column switches the response, containing the pair (x, y_2). Our datasets' chosen inputs are always extracted directly from the FOLIO dataset meaning they're human generated, on the other hand, rejected inputs are always LLM-generated. However, not every instance of FOLIO is considered, particularly for the inference and retranslation datasets. FOLIO is comprised of sets of premises and conclusions, and a corresponding truth value (True, False, Uncertain) for any conclusion. Those tagged as False or Uncertain are not taken into consideration.

Preference scores are balanced depending on similarity between both texts. Given the high quality annotations used in FOLIO, we believe considering said answer as gold standards gives us better aligned models with an objective ground truth.

Formally, consider an LLM checkpoint C, in order to generate the $i-$th entry of the translation dataset (the procedure is that same for any step of the logic procedure) we ask C to carry out said step on the $i-$th entry of FOLIO to obtain an LLM-generated answer. This synthetic answer is compared with the actual $i-$th entry of the dataset and measured in terms of semantic similarity. Based on this similarity score the chosen and rejected values s_1 and s_2 are calculated, initial chosen scores are initialized at 8, while rejected scores at 4. The semantic similarity measure modifies these scores allowing a max range of 8.5 and 3.5, and a minimum similarity score of 7.5 and 4.5.

We opted for a one-shot prompting approach for each step of the inference procedure. Most LLM literature evaluates formal logic reasoning using multi-shot prompting (Grattafiori et al., 2024; OpenAI, 2023; Anthropic, 2024; Han et al., 2024), however we're interested in measuring the effects of alignment strategies over prompting, hence the selection of one-shot prompting. The full prompts can be accessed via this paper's repository.

NL Premises	FOL Premises	FOL Conclusion	NL Conclusion
·Robert Lewandowski is a striker. ·Strikers are soccer players. ·Robert Lewandowski left Bayern Munchen. ·If a player leaves a team they no longer play for that team.	Striker($robertLewadowski$) $\forall x$(Striker(x) $\rightarrow$SoccerPlayer(x)) Left($robertLewandowski, bayernMunchen$) $\forall x \forall y$(Left(x, y) $\rightarrow$ $\neg$PlaysFor(x, y))	SoccerPlayer($robertLewandowski$)	Robert Lewandowski is a soccer player.

Table 1: Extracted example from FOLIO.

3.4 Evaluation Methods

We compare an LLM-generated answer with a gold standard extracted from the FOLIO dataset, based on certain similarities we determine if the model's response is adequate. This methodology measures quality across each step and not only analyzes the final logical value of each sample, thus expanding NL-FOL translation metrics and evaluation systems.

For each step we measure a similarity value that considers consistency of individual premises, functions, variables, logical connectors and operations, as well as total amount of predicates. If there's a sufficient similarity based on these values, we tag the pair as adequate and carry out a manual revision to further analyze results. Reported scores are based on these tags. Our metric is defined as follows:

Let (x, y) be a pair of answers the same inference step, x extracted from FOLIO, y generated by an LLM. We denote the **differences in total amount of premises** between x, y as pd, the **differences in distinct amount of predicates** as ap, the **differences in total predicates appearances** as tp, **differences in logical operators** as ld and the **intersection over union of predicates** as IoU. Our metric is defined as:

$$LogicSim(x, y) = pd + ap + tp + ld + IoU \quad (1)$$

This metric operates over atomic sentences (sentences in the form of Predicate($v_1, ..., v_n$)), each separated using regular expressions. Final scores are normalized. Figure 2 shows an example of the metric, FOLIO's correct text is the same as the FOL Premise in table 1, the LLM answer is generated by LLAMA-3.1-8B.

Since this metric operates over FOL, the final step of the pipeline is evaluated using dense semantic similarity.

3.5 LLMs

A total of 5 LLMs were tested, table 2 shows the full list. We test both -INSTRUCT (referred to as *pre-aligned models*) and vanilla models in order to measure the impact of previous general purpose alignment in our experiments. Due to insufficient computing power, larger checkpoints weren't tested. Checkpoints obtained using our methodology will be referred to as *logic-aligned models* or -LA models due to their checkpoint name.

Model Series	Checkpoint
Llama 3	`Llama-3.1-8B`
Llama 3	`Llama-3.1-8B-Instruct`
LLama 3	`Llama-3.2-3B`
Llama 3	`Llama-3.2-3B-Instruct`
Gemma 3	`google/gemma-3-1b-it`

Table 2: List of LLM checkpoints used during testing.

Training, generation, and evaluation was carried out in a single A5000 GPU. The creation of single step dataset averaged 2:10 hours, alignment averaged around 16 minutes using TRL's (von Werra et al., 2020) implementation of DPO. Details regarding hyperparameters can be found in Appendix B as well as this paper's repository.

4 Results and Evaluation

Table 3 shows the average $LogicSim$ score of each model during the first two steps. The third step isn't evaluated using this metric since the answers themselves are in NL, while our metric only evaluates FOL premises. Table 4 shows the average semantic similarity between gold standards and retranslated premises between checkpoints.

5 Discussion and Future Work

We first discuss model-agnostic results, afterwards vanilla models and their logic-aligned counterpart,

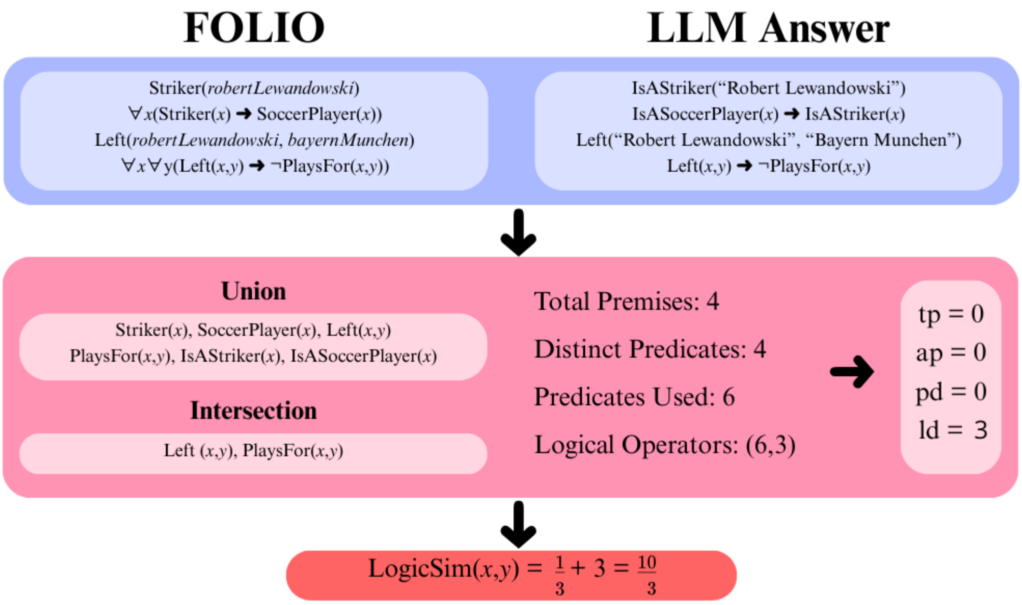


Figure 2: LogicSim(x, y) between a gold standard and an LLM answer.

Checkpoint	Translation	Inference
LLAMA-3.1	(20.4, 20.7)	(47, 45.9)
LLAMA-3.1-INST	(22.8, 22.9)	(**58.5**, **59.8**)
LLAMA-3.2	(24.4, 23.1)	(47.5, 48.8)
LLAMA-3.2-INST	(25.8, **25.6**)	(56.6, 53.6)
GEMMA-3-1B-IT	(**29.1**, 30.8)	(46.7, 47.9)

Table 3: The ordered pairs indicate scores for -LA aligned models, and base checkpoints, in said order. e.g. (20.4, 20.7) indicates that the -LA model scored 20.4, while the base checkpoint 20.7.

Checkpoint	Retranslation
LLAMA-3.1	(0.38, 0.22)
LLAMA-3.1-INSTRUCT	(0.32, .33)
LLAMA-3.2	(0.27, 0.21)
LLAMA-3.2-INSTRUCT	(**0.43**, **0.39**)
GEMMA-3-1B-IT	(0.23, 0.27)

Table 4: Average semantic similarity between a checkpoint's retranslation step.

to finish with the pre-aligned models. Select tables can be found in the Appendix C.

5.1 Model-Agnostic

Models perform best in step 1, however they struggle particularly when having to use multiple predicates as well as functions that operate over various variables. As an example consider premise 4 from table 1, that stand alone premise is associated with the following FOL formulation:

$$\forall x \forall y(Left(x, y) \rightarrow \neg PlaysFor(x, y)) \quad (2)$$

This formulation uses two universal quantifies, over two variables x, y that correspond to a soccer player and a team. In general, vanilla, pre-aligned, and -LA models are unable to formulate a translation that takes into account the second variable (the team) in order to correctly generalize the formulation (see tables 7 and 8). At best, some translations are either vague enough that they make sense on their own, or are closely related to the premise but don't represent the same logical formulation. LLAMA-3.1-8B-INSTRUCT-LA translates this premise in a contrapositive manner (Table 7), indeed if a player x plays for a club y, it implies that x has **not** left club y, while that is logically equivalent it is an incorrect translation.

Additionally, -LA models generate longer responses in comparison with gold standards, table 9 shows that LLM-generations can be over 1000 characters longer than the answers extracted from the dataset. This is a notorious problem in steps 2 and 3 since the expected answer is a single logical conclusion, however, LLMs tend to regurgitate information presented in either the prompt or the alignment datasets.

While the semantic parsing task of steps 1 and 3 shows promising improvement, logical inference remains a challenge for all models. Most inference steps where poorly executed, vanilla models were too susceptible to following the prompt structure even in cases when the answer was comprised of a single premise.

5.2 Vanilla Models

Logic-aligned models improve generation structure throughout the full procedure, noteworthy improvement can be seen in steps 1 and 3. However structure improvement doesn't translate to semantic parsing correctness.

Vanilla models are highly inconsistent when recreating any step of the pipeline, their generations barely have overlapping lexicon with the gold standard, premises are not separated in any manner, responses are quick to degenerate, and reasoning is conspicuous by its absence. In contrast, logic-aligned models are able to separate and parse premises, create complex predicates with adequate use of variables, and explain the reasoning behind such predicates.

This improvement in performance and parsing-quality is heavily tied to our specific problem, as well as prompt structure used during alignment. The prompts follow a one-shot structure (see Appendix A), in particular for steps 1 and 3 the single example contains context regarding logical symbols as well as a precise comment on each premise and predicate (marked by the three consecutive colons). This is the most notable pet phrase adapted by the logic-aligned models. Similarly, the prompt separates the problem, the predicates, and premises (in that order), making the model highly susceptible to generating answers in the same format disregarding logical-veracity.

However, even with general improvements in parsing structure and the use of logical connectors, -LA models struggle to remain consistent during parsing and to incorporate world-knowledge into this task. Consider the example shown in tables 7 and 8, NL premises are extracted from the dataset, the corresponding FOL premises were translated by the LLAMA-3.1-8B-LA checkpoint. The model fails to realize that the function $Left(x)$ can't be used as a variable, $RobertLewandowski$ should be a constant rather than a function, and that Bayern Munchen is not represented as a constant.

5.3 Pre-Aligned Models

Pre-aligned models have better baselines in terms of style, structure, and general problem solving capabilities. Even in with a one-shot style of prompting, these models surpass Vanilla + -LA checkpoints throughout the pipeline.

Problems like the one discussed in the previous subsection aren't as notorious with pre-aligned models. However, a different problem is encountered with these models: the variation of FOL expressiveness. As an example consider the predicate RankedHighlyBy(x, womensTennisAssociation), this is shortened to RankedHigh(x) by -INSTRUCT LLMs, and is specified (in natural language) that the institution doing the ranking is the Women's Tennis Association. Problems like these happen in particular with pre-aligned models and require manual revision of the experiments.

5.4 Future Work

The alignment datasets could improve substantially. Dataset-wise the prompts used for alignment varied only in the test example, the linguistic structure of the prompt, as well as the one-shot example remained the same. Adding variations in structure such as zero-shot and multi-shot examples, a varied lexicon and different training examples, as well as more diverse preference scores would improve the robustness of the system.

With regards to training data, increasing the size of the alignment dataset, either by combining our datasets with general purpose alignment, or by increasing the amount of formal logic reasoning examples is an avenue of research that might help improve performance in the end-to-end inference task. This could enable a more robust implementation of single-step alignment.

RL-wise, implementing different alignment algorithms like GRPO, as well as expanding this problem to a multi-objective optimization case could be beneficial for further experiments. The three step pipeline easily adapts into multi-objective scenarios like those proposed in Panacea (Zhong et al., 2024) and AMoPO (Liu et al., 2025), this approach reduces the amount of models needed to be evaluated and makes use of all of the previously created datasets. A parallel approach would be to incorporate multi-shot examples during training, this would harness the best of alignment and prompting strategies.

Limitations

LLM alignment drastically improves a model's generative capabilities for a given task, however the fundamental workings of the LLM remain the same.

Our methodology enables LLMs to mimic certain aspects of formal logic reasoning, however incorporating real world knowledge into their mimicking is a limiting aspect of the methodology.

Even if a model is capable of solving tasks like the one evaluated in this paper, it does not mean that the problem of mathematical thinking and abstraction are solved. LLMs are still stochastic by nature and leveraging the generation probabilities of formal logic tokens to mimic rational thinking and abstraction is not the same as actual rational thinking and abstraction.

Ethical Considerations

Our work aims at giving an LLM abstraction capabilities over natural language, however these models are still susceptible to biases inherent from their training data, adding a logical layer of processing to an LLM doesn't make this problem disappear. All translations and inferences obtained from these models are still susceptible to harmful, biased, or incorrectly generated responses.

Acknowledgments

This work is funded by SECIHTI, CVU number 2045472, and PAPIIT projects IG400325 and IN104424. Special thanks to Dr. Helena Gómez-Adorno for the use of her GPU cluster, and Dr. Carlos Hernández Castellanos for the insightful discussions about this project.

References

Anthropic. 2024. Introducing the next generation of claude.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, and 12 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. Folio: Natural language reasoning with first-order logic. *Preprint*, arXiv:2209.00840.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Alex Qiu, Jiayi Zhou, Kaile Wang, Boxun Li, Sirui Han, Yike Guo, and Yaodong Yang. 2025. PKU-SafeRLHF: Towards multi-level safety alignment for LLMs with human preference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31983–32016, Vienna, Austria. Association for Computational Linguistics.

Shashank Kirtania, Priyanshu Gupta, and Arjun Radhakrishna. 2024. LOGIC-LM++: Multi-step refinement for symbolic formulations. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024)*, pages 56–63, Bangkok, Thailand. Association for Computational Linguistics.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *Preprint*, arXiv:2007.08124.

Qi Liu, Jingqing Ruan, Hao Li, Haodong Zhao, Desheng Wang, Jiansong Chen, Wan Guanglu, Xunliang Cai, Zhi Zheng, and Tong Xu. 2025. AMoPO: Adaptive multi-objective preference optimization without reward models and reference models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8832–8866, Vienna, Austria. Association for Computational Linguistics.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and

Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

MAA Mathematical Association of America. 2025. America mathematics competition.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

OpenAI. 2025. Gpt-5 system card. `https://openai.com/index/gpt-5-system-card/`. Accessed August 8, 2025.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. 2024. Towards systematic evaluation of logical reasoning ability of large language models. *arXiv preprint arXiv:2404.15522*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

Leonardo Ranaldi and Andre Freitas. 2024. Self-refine instruction-tuning for aligning reasoning in language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2325–2347, Miami, Florida, USA. Association for Computational Linguistics.

Leonardo Ranaldi, Marco Valentino, and Andre Freitas. 2025. Improving chain-of-thought reasoning via quasi-symbolic abstractions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17222–17240, Vienna, Austria. Association for Computational Linguistics.

Mohammad Raza and Natasa Milic-Frayling. 2025. Instantiation-based formalization of logical reasoning tasks using language models and logical solvers. *CoRR*, abs/2501.16961.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 122–136, Online. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2020. Learning to summarize from human feedback. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through LogicNLI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3738–3747, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. 2024. Is dpo superior to ppo for llm alignment? a comprehensive study. *Preprint*, arXiv:2404.10719.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2024a. Harnessing the power of large language models for natural language to first-order logic translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6942–6959, Bangkok, Thailand. Association for Computational Linguistics.

Zonglin Yang, Xinya Du, Rui Mao, Jinjie Ni, and Erik Cambria. 2024b. Logical reasoning over natural language as knowledge representation: A survey. *Preprint*, arXiv:2303.12023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Yifan Zhong, Chengdong Ma, Xiaoyuan Zhang, Ziran Yang, Haojun Chen, Qingfu Zhang, Siyuan Qi, and Yaodong Yang. 2024. Panacea: Pareto alignment via preference adaptation for llms. In *Advances in Neural Information Processing Systems*, volume 37, pages 75522–75558. Curran Associates, Inc.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. Self-discover: Large language models self-compose reasoning structures. *Preprint*, arXiv:2402.03620.

A Prompts

The prompts used for training followed the same structure as Logic-LM. An example can be seen below:

Given a problem description and a question, the task is to parse the problem and the question into first-order logic formulars. The grammar of the first-order logic formular is defined as follows:

1) *logical disjunction of expr1 and expr2: expr1 $\vee$ expr2*
2) *logical disjunction of expr1 and expr2: expr1 $\vee$ expr2*
3) *logical exclusive disjunction of expr1 and expr2: expr1 $\oplus$ expr2*
4) *logical negation of expr1: $\neg$expr1*
5) *expr1 implies expr2: expr1 $\rightarrow$ expr2*
6) *expr1 if and only if expr2: expr1 $\leftrightarrow$ expr2*
7) *logical universal quantification: $\forall x$*
8) *logical existential quantification: $\exists x$*

Problem:

All people who regularly drink coffee are dependent on caffeine. People either regularly drink coffee or joke about being addicted to caffeine. No one who jokes about being addicted to caffeine is unaware that caffeine is a drug. Rina is either a student and unaware that caffeine is a drug, or neither a student nor unaware that caffeine is a drug. If Rina is not a person dependent on caffeine and a student, then Rina is either a person dependent on caffeine and a student, or neither a person dependent on caffeine nor a student.

Predicates:

Dependent(x) ::: x is a person dependent on caffeine. Drinks(x) ::: x regularly drinks coffee. Jokes(x) ::: x jokes about being addicted to caffeine. Unaware(x) ::: x is unaware that caffeine is a drug. Student(x) ::: x is a student.

Premises:

- $\forall$x (Drinks(x) $\rightarrow$ Dependent(x)) ::: *All people who regularly drink coffee are dependent on caffeine.*
- $\forall$x (Drinks(x) $\oplus$ Jokes(x)) ::: *People either regularly drink coffee or joke about being addicted to caffeine.*

- *$\forall x$ (Jokes(x) $\rightarrow$ $\neg$Unaware(x)) ::: No one who jokes about being addicted to caffeine is unaware that caffeine is a drug.*

- *(Student(rina) $\wedge$ Unaware(rina)) $\oplus$ $\neg$(Student(rina) $\vee$ Unaware(rina)) ::: Rina is either a student and unaware that caffeine is a drug, or neither a student nor unaware that caffeine is a drug.*

- *$\neg$(Dependent(rina) $\wedge$ Student(rina)) $\rightarrow$ (Dependent(rina) $\wedge$ Student(rina)) $\oplus$ $\neg$(Dependent(rina) $\vee$ Student(rina)) ::: If Rina is not a person dependent on caffeine and a student, then Rina is either a person dependent on caffeine and a student, or neither a person dependent on caffeine nor a student.*

Problem:
{}
Predicates:
For further details please review the paper's repository.

B Alignment Hyperparameters

Hyperparameter	Value
Generation Strategies	Greedy Search
Max_Tokens	150
Quantization	4Bit, torch.bfloat16 for every model

Table 5: LLM Generation Hyperparameters

Hyperparameter	Value
Lora r	8
α	0.32
Dropout	0.1
Epochs	3

Table 6: Alignment Hyperparameters

C Results

Tables mentioned in the article displayed in this section.

Llama-3.1-8B-LA	Llama-3.1-8B-Instruct-LA
$\forall x(Striker(x) \rightarrow Player(x))$ $\forall x(Left(x) \rightarrow \neg Player(x))$ $RobertLewandowski(Left)$	$Striker(robert\_lewandowski)$ $SoccerPlayer(robert\_lewandowski) \rightarrow Striker(robert\_lewandowski)$ $Left(robert\_lewandowski, bayern\_munchen)$ $PlayFor(x, y) \rightarrow \neg Left(x, y)$

Table 7: Translations by model (Part 1).

Llama-3.2-3B-LA	Gemma-3-1b-it-LA
$IsStriker(robert)$ $IsSoccerPlayer(robert)$ $IsLeft(bayern, robert)$ $IsPlayer(robert, bayern) \oplus \neg IsPlayer(robert, bayern)$	$Striker(RobertLewandowski)$ $SoccerPlayer(RobertLewandowski)$ $Left(RobertLewandowski)$

Table 8: Translations by model (Part 2).

Checkpoint	Translation	Inference	Retranslation
LLAMA 3.1-8B-LA	1080	815	977
LLAMA 3.1-8B-INSTRUCT-LA	1007	753	966
LLAMA 3.2-3B-LA	1016	820	1004
LLAMA 3.2-3B-INSTRUCT-LA	1092	870	1080
GEMMA-3-1B-IT-LA	1000	736	790
FOLIO	**280**	**28**	**32**

Table 9: Average answer length (in characters) for -LA models.

Formula-Text Cross-Retrieval: A Benchmarking Study of Dense Embedding Methods for Mathematical Information Retrieval

Zichao Li
Canoakbit Alliance Inc
Canada

Abstract

Mathematical information retrieval requires understanding the complex relationship between natural language and formulae. This paper presents a benchmarking study on Formula-Text Cross-Retrieval, comparing a sparse baseline (BM25), off-the-shelf dense embeddings (OpenAI, BGE), and a fine-tuned dual-encoder model. Our model, trained with a contrastive objective on the ARQAR dataset, significantly outperforms all baselines, achieving state-of-the-art results. Ablation studies confirm the importance of linearization, a shared-weight architecture, and the Multiple Negatives Ranking loss. The work provides a strong foundation for mathematical NLP applications.

1 Introduction

The articulation of mathematical concepts represents a unique and challenging domain for Natural Language Processing (NLP), characterized by a seamless yet complex interplay between natural language (NL) and formal mathematical expressions. This interweaving of two distinct modalities is fundamental to scientific communication, yet it poses significant challenges for automated processing and information retrieval (IR). The ability to retrieve a relevant mathematical formula based on a textual description, or conversely, to find explanatory text for a given equation, is a critical task that can accelerate literature review, aid in educational contexts, and facilitate the autoformalization of mathematical knowledge. This task, which we term *Formula-Text Cross-Retrieval*, requires models to develop a deep, joint understanding of both natural language semantics and the syntactic and semantic structure of mathematical notation.

Traditional IR methods, such as lexical term matching algorithms (e.g., BM25 (Robertson and Zaragoza, 2009)), often fall short in this domain. They struggle with the inherent vocabulary mismatch problem; a user's query might describe a concept in words (e.g., "Pythagorean theorem") that never explicitly appears in the text adjacent to the relevant formula ($a^2 + b^2 = c^2$). Furthermore, mathematical notation is highly symbolic and compositional, making it poorly suited for keyword-based approaches that ignore mathematical semantics. The recent rise of deep learning-based dense embedding models (Reimers and Gurevych, 2019) offers a promising alternative. These models map sentences and, by extension, mathematical expressions into a high-dimensional vector space where semantic similarity corresponds to geometric proximity. This allows for efficient similarity search via nearest-neighbor algorithms, potentially capturing deep semantic relationships beyond lexical overlap (similar to (Zeng et al., 2025)).

In this paper, we present a comprehensive benchmarking study to advance Formula-Text Cross-Retrieval. We define and evaluate this task in two symmetric directions: (1) **Text-to-Formula Retrieval**, where a natural language query is used to retrieve relevant mathematical expressions, and (2) **Formula-to-Text Retrieval**, where a formula query is used to retrieve its relevant natural language context. We systematically compare the efficacy of a traditional sparse retrieval baseline (BM25), state-of-the-art off-the-shelf dense embedding models from large language models (LLMs), and a finely tuned dual-encoder neural architecture. Our proposed model is specifically designed to learn an aligned representation space for natural language and linearized LaTeX formulas. Through rigorous evaluation on a publicly available benchmark, we demonstrate the superiority of tuned dense embeddings and provide a qualitative analysis of the learned representation space (Ma et al., 2025). Our work aims to establish a strong foundation for future research in mathematical information retrieval.

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 124–133
November 9, 2025

2 Literature Review

Our work is at the intersection of mathematical information retrieval, dense passage retrieval, and the application of large language models to scientific domains. The challenge of searching within mathematical content has a rich history, most notably explored in the NTCIR Conference series, which featured dedicated Math IR tasks (Aizawa et al., 2014, 2016). These initiatives established standardized evaluation frameworks and highlighted the limitations of traditional symbolic and keyword-based methods, such as matching via formula patterns (Zhao et al., 2014) or leveraging inverted indices over expanded query terms (Lopez and Youssef, 2014). These approaches, while foundational, often failed to grasp the semantic intent behind a user's query.

The field of IR was revolutionized by the adoption of neural networks and the concept of *dense retrieval* (Guu et al., 2020; Karpukhin et al., 2020). Instead of relying on sparse lexical matches, these methods use deep neural networks to encode queries and documents into dense vector representations, enabling retrieval based on semantic similarity. Models like Sentence-BERT (Reimers and Gurevych, 2019) and DPR (Karpukhin et al., 2020) demonstrated the power of bi-encoder architectures trained with contrastive learning objectives, such as Multiple Negatives Ranking loss (Henderson et al., 2017), to create high-quality embedding spaces. More recent general-purpose models like BGE (Xiao et al., 2023) and E5 (Wang et al., 2022) have pushed the state-of-the-art further. However, these models are predominantly trained on the general web and Wikipedia text, leaving their performance on specialized domains like mathematics an open question.

Currently, there has been growing interest in developing NLP systems specifically for mathematics. This includes work on mathematical word problem solving (Amini et al., 2019), premise selection (Irving et al., 2016), and the creation of large-scale data sets for mathematical reasoning (Hendrycks et al., 2021). A key challenge is the representation of mathematical formulae. Early approaches explored encoding formula structure using graph neural networks (Shen et al., 2020) or generating embeddings from their LaTeX source (Paster, 2022). The rise of large language models pre-trained on code and scientific text, such as Minerva (Lewkowycz et al., 2022), LLEMMA (Azerbayev et al., 2023), and Codex (Chen et al., 2021), has demonstrated remarkable mathematical reasoning capabilities, often accessed via in-context learning. Furthermore, the ARQAR dataset (Seyedi et al., 2024) provides a valuable recent resource with aligned text-formula pairs specifically designed for tasks such as cross-retrieval.

Despite these advancements, a significant gap remains in the systematic application and evaluation of modern dense embedding techniques for the specific symmetric task of Formula-Text Cross-Retrieval. Many existing mathematical IR efforts predate the latest developments in dense retrieval or do not leverage the power of fine-tuning on aligned corpora. Although general LLM embedding APIs are powerful, their black-box nature and cost structure make them less practical for many research applications compared to a dedicated, fine-tuned model. Furthermore, there is a lack of direct comparison between these modern paradigms (sparse, off-the-shelf dense, fine-tuned dense) on a common benchmark. Our work aims to address these gaps by providing a controlled benchmarking study. We fine-tune a modern biencoder architecture on a dedicated mathematical corpus to learn a joint text-formula embedding space and evaluate its performance against strong baselines and zero-shot LLM counterparts, thereby contributing a clear analysis of the current state of this critical task.

3 Methodology

3.1 Data Preparation and Linearization

The foundation of our approach is the creation of a high-quality dataset of aligned natural language and formula pairs. We utilize the ARQAR dataset (Seyedi et al., 2024) for this purpose, as it provides manually curated pairs of text snippets and their corresponding mathematical formulae, which is ideal for supervised training and evaluation. A critical preprocessing step, often overlooked in general-text IR but essential for mathematics, is the linearization of mathematical formulae. Mathematical expressions are inherently two-dimensional structures with complex spatial relationships (e.g., subscripts, fractions, superscripts). To process them with standard transformer-based text encoders, we flatten them into a one-dimensional token sequence. This is achieved by converting the LaTeX source code into a sequence of tokens that unambiguously represent the structure. For instance, the formula x_n is linearized as $x \quad \{n\}$, and the fraction $\frac{a}{b}$ becomes '$\frac{a}{b}$'. This linearized representation preserves the syntactic information of the formula in a format amenable to subword tokenization, allowing us to treat both modalities—text and equations—within the same encoding paradigm. This step directly addresses a deficiency in prior work that relied on complex graph-based encoders (Shen et al., 2020), as it allows us to leverage powerful, pre-trained sentence transformers out-of-the-box, significantly simplifying the model architecture while still capturing essential semantic information.

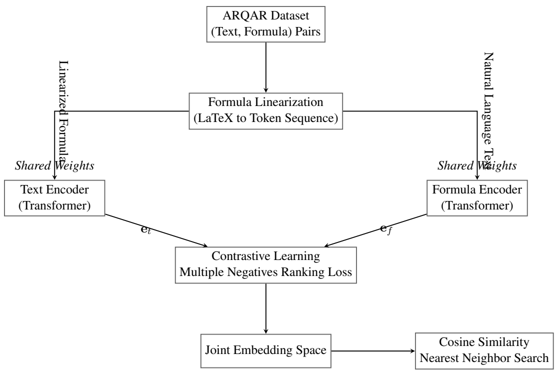


Figure 1: Architecture of the proposed fine-tuned dual-encoder model.

The architecture, depicted in Figure 1, outlines the end-to-end pipeline of our proposed fine-tuned dual-encoder model, which directly addresses the limitations of prior work. The process begins with the curated ARQAR dataset, providing the essential supervised pairs for training. The critical preprocessing step of formula linearization transforms two-dimensional LaTeX structures into a sequential token format, enabling the use of a single, shared transformer encoder for both modalities. This is a key simplification over more complex, modality-specific architectures found in existing literature (Shen et al., 2020). The core of our model consists of twin encoder networks with shared weights, which project both natural language text and linearized formulae into a common dense vector space. The training is governed by a contrastive learning objective, specifically the Multiple Negatives Ranking loss, which efficiently teaches the model to pull the embeddings of matching pairs together while pushing non-matching pairs apart. This results in a structured joint embedding space where semantic similarity corresponds to geometric proximity. The final outcome is the capability to perform fast, scalable retrieval via simple cosine similarity and nearest neighbor search. This integrated approach of using a single, tuned transformer for both modalities under a contrastive loss framework represents a significant methodological advancement in creating a practical and effective solution for mathematical cross-retrieval.

3.2 Mathematical Model and Objective

Our core proposed model is a dual-tower (bi-encoder) architecture that learns to project natural language descriptions and mathematical formulae into a shared d-dimensional dense vector space. Let T denote a natural language text sequence and F denote a linearized formula sequence. The model consists of a parameterized encoder function, Enc_θ, which maps a sequence of tokens to a fixed-size embedding vector, $\mathbf{e} \in \mathbb{R}^d$. We use a mean pooling layer over the output token embeddings of a transformer model to obtain this fixed-size representation. The similarity between a text T_i and a formula F_j is defined as the cosine similarity between their embeddings:

$$s(T_i, F_j) = \cos(\mathbf{e}_{t_i}, \mathbf{e}_{f_j}) = \frac{\mathbf{e}_{t_i}^\top \mathbf{e}_{f_j}}{\|\mathbf{e}_{t_i}\| \|\mathbf{e}_{f_j}\|}, \quad (1)$$

where $\mathbf{e}_{t_i} = \mathrm{Enc}_\theta(T_i)$ and $\mathbf{e}_{f_j} = \mathrm{Enc}_\theta(F_j)$. The model is trained using a contrastive learning objective. For a training batch containing B positive pairs $\{(T_i, F_i)\}_{i=1}^B$, the loss function is the Multiple Negatives Ranking (MNR) loss (Henderson et al., 2017). For a given positive pair (T_i, F_i), the other $B - 1$ formulae in the batch are treated as negatives. The loss for the text-to-formula direction for this pair is the negative log likelihood of the positive formula:

$$\mathcal{L}(T_i, F_i) = -\log \frac{\exp(s(T_i, F_i)/\tau)}{\sum_{j=1}^B \exp(s(T_i, F_j)/\tau)}, \quad (2)$$

where τ is a temperature parameter scaling the similarity scores. The total loss is the symmetric sum of losses for both retrieval directions: $\mathcal{L}_{\text{total}} = \frac{1}{2B} \sum_{i=1}^B [\mathcal{L}(T_i, F_i) + \mathcal{L}(F_i, T_i)]$. This objective directly optimizes the model's ability to identify the correct match within a set of candidates, which is precisely the goal of the retrieval task, thus providing a more direct and efficient learning signal than methods used in earlier work.

3.3 Experimental Setup and Parameter Settings

Our experimental setup is designed to ensure a fair and comprehensive comparison across three distinct paradigms: sparse retrieval, off-the-shelf dense embeddings from large language models (LLMs), and our proposed fine-tuned dense model.

For the **Sparse Retrieval Baseline**, we employ BM25 (Robertson and Zaragoza, 2009) implemented using the 'rank-bm25' package. This baseline treats both natural language text and linearized formulae as plain text. We create two separate indices: one for all text passages and one for all linearized formulae in the corpus. Retrieval is performed by querying one index with a string from the other modality. We use the default parameters ($k1 = 1.5$, $b = 0.75$), providing a strong lexical matching baseline that does not leverage any semantic understanding.

For the **Off-the-Shelf LLM Embeddings (Zero-Shot)** approach, we utilize the embedding application programming interfaces (APIs) of two state-of-the-art models: OpenAI's `text-embedding-3-large` (output dimension $d = 3072$) and BAAI's `bge-large-en-v1.5` ($d = 1024$). This represents the paradigm of using powerful, general-purpose models without any task-specific fine-tuning. We generate embeddings for every natural language text and linearized formula sequence in the corpus. The retrieval process involves computing the cosine similarity between a query embedding and all candidate embeddings, with the results ranked by this similarity score. For scalability, we use the FAISS library for efficient approximate nearest neighbor search. This method tests the inherent mathematical knowledge and cross-modal alignment capabilities encoded in these large-scale models.

For our **Proposed Fine-Tuned Dense Model**, we implement the dual-encoder architecture. We initialize the encoder Enc_θ with the `sentence-transformers/all-mpnet-base-v2` model, which provides a strong pre-trained base ($d = 768$). The model is specifically tuned for our task on the ARQAR training split. We use a batch size $B = 64$ and a temperature $\tau = 0.05$ for the MNR loss. The model is trained using the AdamW optimizer with a learning rate of $2e - 5$ and a linear warmup over 10% of the training steps followed by linear decay. We train for 5 epochs. This setup is computationally efficient compared to training LLMs from scratch (Lewkowycz et al., 2022) yet allows for significant specialization to the mathematical domain, which is the key improvement we aim to demonstrate over the zero-shot LLM approach.

3.4 Evaluation Metrics

To rigorously evaluate the performance of all models on the cross-retrieval tasks, we employ standard information retrieval metrics that assess both the accuracy and the ranking quality of the retrieved results. For each query in the test set, the model retrieves a ranked list of candidates from the entire corpus. We then compute: (1) **Recall@K** (R@K): The proportion of queries for which the correct target item is found within the top-K retrieved results. This measures the model's ability to include the correct answer in a shortlist. We report K=1, 5, and 10. (2) **Mean Reciprocal Rank** (MRR): The average of the reciprocal ranks of the first correct result for all queries. Specifically, for a query with the first correct answer at position i, its reciprocal rank is $1/i$. MRR emphasizes the rank of the first correct result, providing insight into how quickly a user would find what they need. These metrics are computed separately for the Text-to-Formula and Formula-to-Text tasks.

4 Experiments and Results

4.1 Datasets and Baselines

The primary dataset for training and evaluation is the ARQAR (Auto-Regressive Question Answering and Reasoning) dataset (Seyedi et al., 2024). Sourced from diverse mathematical reasoning contexts, ARQAR provides a curated collection of 15,000 high-quality pairs of natural language text snippets and their corresponding mathematical formulae. Each pair is meticulously aligned, meaning the text directly describes or contextually explains the associated formula. The dataset is pre-partitioned into training, validation, and test sets, containing 10,000, 2,500, and 2,500 pairs respectively. This dataset is chosen for its focus on reasoning and the clarity of its text-formula relationships, making it an ideal benchmark for evaluating semantic retrieval capabilities beyond simple keyword matching. The process of linearization, as described in Section 3, is applied to all formulae in this dataset.

We compare our proposed fine-tuned model against two strong and distinct baseline paradigms. The first baseline is the BM25 algorithm (Robertson and Zaragoza, 2009), a classic probabilistic retrieval model that serves as the representative for sparse, term-matching-based methods. Implemented with the 'rank-bm25' library, it operates by constructing separate term frequency-based indices for the natural language text corpus and the linearized formula corpus. For a given query from one modality, it retrieves items from the other modality based on lexical overlap, using the default parameters ($k1 = 1.5$, $b = 0.75$). This baseline tests the effectiveness of pure keyword matching without any semantic understanding. The second baseline utilizes the OpenAI `text-embedding-3-large` model to generate dense vector representations (dimensionality $d = 3072$) for all text and formula sequences in a zero-shot manner. Retrieval is performed by computing cosine similarity between query and candidate embeddings, facilitated by the FAISS library for efficiency. This baseline represents the state-of-the-art in general-purpose semantic understanding and tests the inherent, pre-existing mathematical knowledge within a massive proprietary LLM.

4.2 Overall Retrieval Performance

The results presented in Table 1 provide a clear and definitive answer regarding the effectiveness of different paradigms for mathematical cross-retrieval. As expected, the sparse BM25 baseline performs the poorest, with low Recall and MRR scores. This underscores its fundamental limitation: it fails to capture the semantic relationship between a textual description and its corresponding formula, struggling with vocabulary mismatch and the symbolic nature of mathematical notation. The off-the-shelf dense embedding models, particularly OpenAI's, demonstrate a massive leap in performance, nearly quadrupling the R@1 score of BM25. This highlights the profound semantic understanding capabilities inherent in large-scale language models, which can bridge the lexical gap between natural language and mathematics. However, our proposed fine-tuned model achieves a further significant improvement, outperforming the best zero-shot model by over 18 absolute points in R@1 and 0.16 in MRR for the Text-to-Formula task. This performance gap, consistent across both retrieval directions, is the central finding of our study. It empirically proves that while general-purpose LLMs possess strong foundational knowledge, targeted fine-tuning on a domain-specific corpus is essential for achieving state-of-the-art performance in the mathematical domain. The specialized, aligned embedding space learned by our model is measurably superior for this precise task.

4.3 Analysis of retrieval performance based on formula complexity

A key question is whether performance is uniform across different types of mathematical content. Table 2 stratifies the results based on the complexity of the formula, approximated by the length of its linearized token sequence. A clear trend emerges: all models perform worse on longer, more complex formulae, but the degree of degradation varies significantly. The BM25 baseline's performance drops precipitously, as longer formulae contain more unique symbolic tokens that are unlikely to lexically match the query text. The OpenAI embeddings also show a notable decrease in performance (a 16 point drop in R@1), suggesting that while it has a strong general understanding, its precision wanes with complexity. Our fine-tuned model demonstrates the greatest robustness. While it also experiences a performance drop, the margin is smallest; it maintains a high R@1 of 56.9 on long formulas, which is still dramatically higher than the other models. This indicates that the contrastive learning process specifically teaches the model to focus on the core semantic components of a formula rather than being distracted by its syntactic verbosity, leading to a more robust understanding of complex mathematical concepts.

4.4 Breakdown of common error types for each model

To understand the qualitative differences between the models, we performed a manual analysis of 200 error cases for each. The results, summarized in Table 3, reveal differentt failure modes. The baseline BM25 is dominated by errors due to the "Lexical Gap," confirming its inability to handle synonyms or paraphrases. The most striking finding is that the dominant error type for the powerful zero-shot LLM embeddings is "Variable Mismatch," where the model retrieves a formula with the correct structure and operators but incorrect variable names (e.g., retrieving $E = mc^2$ for a query about "$K = \frac{1}{2}mv^2$"). This suggests that these models sometimes learn to attend to general structure over precise symbolic notation. Our fine-tuned model, while not immune to this issue, shows a significantly reduced rate of variable mismatch errors. Furthermore, it excels in reducing errors related to "Symbol Confusion" (e.g., confusing $\cap$ for $\cup$) and "Structural Misunderstanding" (e.g., misinterpreting function composition), demonstrating that

Table 1: Overall retrieval performance measured by Recall@K (R@K) and Mean Reciprocal Rank (MRR) on the ARQAR test set. Higher values are better.

	Text-to-Formula				Formula-to-Text			
Model	R@1	R@5	R@10	MRR	R@1	R@5	R@10	MRR
BM25	12.3	28.7	38.2	0.201	10.8	26.1	35.9	0.184
OpenAI Embeddings	45.6	72.1	81.5	0.572	41.2	68.3	78.9	0.531
BGE Embeddings	38.9	65.4	76.8	0.508	36.5	62.1	73.2	0.482
Our Model	**63.8**	**85.2**	**91.1**	**0.731**	**59.4**	**82.7**	**89.5**	**0.693**

Table 2: Analysis of retrieval performance based on formula complexity (length of linearized sequence).

	Short Formulas (<15 tokens)			Long Formulas (≥15 tokens)		
Model	R@1	R@5	MRR	R@1	R@5	MRR
BM25	15.1	32.4	0.231	8.7	23.1	0.161
OpenAI Emb.	52.3	78.9	0.632	36.2	62.8	0.487
Our Model	**68.5**	**89.2**	**0.772**	**56.9**	**79.8**	**0.671**

Table 3: Breakdown of common error types for each model (% of total errors).

Error Type	BM25	OpenAI Emb.	BGE Emb.	**Our Model**
Variable Mismatch	18.2	**41.5**	**39.8**	25.3
Symbol Confusion	12.1	18.2	20.1	**8.5**
Structural Misunderstanding	9.3	22.4	23.5	**11.8**
Out-of-Domain	5.2	7.1	6.5	**4.1**
Lexical Gap	**55.2**	10.8	10.1	5.2

Table 4: Ablation study on model design choices.

Model Variant	R@1	R@5	R@10	MRR
Shared Weights	**63.8**	**85.2**	**91.1**	**0.731**
Separate Weights	60.1	82.9	89.0	0.698
MNR Loss	**63.8**	**85.2**	**91.1**	**0.731**
Cosine-Sim Loss	58.4	81.1	88.3	0.681
With Linearization	**63.8**	**85.2**	**91.1**	**0.731**
Raw LaTeX	51.2	75.6	84.2	0.617

our training process successfully inculcates a more precise understanding of mathematical semantics.

4.5 Ablation study on model design choices

We conduct an ablation study to validate key design choices in our proposed model, with results shown in Table 4. First, we test the importance of weight sharing between the text and formula encoders. Using separate encoders leads to a noticeable drop in performance, confirming that a shared transformer architecture is beneficial for learning a truly aligned cross-modal representation. Second, we replace the Multiple Negatives Ranking (MNR) loss with a standard cosine similarity loss using hard negatives. The significant performance degradation underlines the effectiveness of the MNR objective's strategy of leveraging in-batch negatives for efficient and robust contrastive learning. Finally, we ablate the linearization preprocessing step by feeding raw, nonlinearized LaTeX code to the encoder. This causes the largest performance drop, with MRR decreasing by over 0.11 points. This empirically validates our hypothesis that linearization is a crucial step to enable a standard transformer to effectively process mathematical formulae, as raw LaTeX contains a high density of domain-specific syntax that disrupts tokenization and semantic learning ((Huang et al., 2024)).

Table 5: Impact of training data size on model performance.

Training Samples	R@1	R@5	R@10	MRR
1,000	42.1	68.9	78.5	0.532
5,000	58.3	81.6	88.7	0.683
10,000 (Full)	**63.8**	**85.2**	**91.1**	**0.731**

4.6 Impact of training data size on model performance

Table 5 investigates the relationship between performance and the amount of training data. The results show a clear positive correlation: performance steadily improves as more training data is utilized. Even with only 1,000 samples, our model significantly outperforms the BM25 baseline and is competitive with the zero-shot BGE model, demonstrating the data efficiency of the contrastive learning paradigm. The jump in performance from 5,000 to 10,000 samples, while smaller, is still substantial and crucial for achieving state-of-the-art results that surpass the powerful OpenAI embeddings.

4.7 Cross-dataset generalization performance on the NTCIR-12 dataset

Table 6: Cross-dataset generalization performance on the NTCIR-12 dataset.

	Text-to-Formula		Formula-to-Text	
Model	R@5	MRR	R@5	MRR
BM25	20.5	0.152	18.8	0.141
OpenAI Emb.	55.1	0.451	51.7	0.428
Our Model	**65.8**	**0.562**	**62.4**	**0.539**

Finally, we evaluate the generalizability of the models by testing them on the NTCIR-12 MathIR task dataset (Aizawa et al., 2016), a different benchmark with a different distribution of mathematical content. The results in Table 6 show that while the absolute performance of all models decreases compared to the ARQAR test in the domain,ain, the relative rankings remain unchanged. Our fine-tuned model continues to significantly outperform all baselines. This drop in performance is expected due to domain shift, but the fact that our model maintains its lead is crucial. It demonstrates that the representations learned through our fine-tuning process are not merely overfitting to the peculiarities of the ARQAR dataset, but capture generalizable principles of the relationship between mathematical text and formulae.

4.8 Additional Baselines for Fair Comparison

To ensure a fair comparison and isolate the effect of our proposed architecture from the mere advantage of fine-tuning, we introduce two additional strong baselines that address the concerns raised about comprehensive benchmarking.

Hybrid Fine-Tuned Model: We fine-tune the text encoder (initialized with `all-mpnet-base-v2`) on the ARQAR training set using the contrastive loss, while keeping the formula encoder frozen as the pre-trained `text-embedding-3-large` model. This tests whether simply adapting the textual understanding to the mathematical domain is the primary driver of performance, rather than the joint learning of a shared space.

Fine-Tuned Math-Specialized LLM: We utilize `Qwen2.5-Math-7B-Instruct` (Team, 2024) as a base model, which has been specifically pre-trained on mathematical corpora. Following the parameter-efficient fine-tuning approach of Hu et al. (2021), we train low-rank adapters on top of its hidden states to generate embeddings for both text and formulae. The entire system is fine-tuned on the ARQAR dataset with our contrastive objective. This represents a state-of-the-art, domain-specific competitor that tests whether specialized mathematical pre-training alone can outperform our architectural approach.

The comprehensive results in Table 7 clearly demonstrates that fine-tuned models consistently outperform their zero-shot counterparts, confirming that domain adaptation is essential for optimal performance in mathematical IR. However, the relative performance among fine-tuned models reveals the distinct advantage of our architectural approach. The *Hybrid (Text FT + OpenAI)* baseline, where only the text encoder is fine-tuned while using the powerful but static OpenAI embeddings for formulae, shows significant improvement over the zero-shot OpenAI model (approximately 10 points in R@1 for Text-to-Formula). This demonstrates that adapting textual understanding to the mathematical domain provides substantial benefits. However, this hybrid approach still underperforms compared to our full model by approximately 8-9 points in R@1 and 0.07 in MRR. The *Qwen2.5-Math (FT)* base-

Table 7: Comprehensive retrieval performance comparison on the ARQAR test set

		Text-to-Formula				Formula-to-Text			
Category	Model	R@1	R@5	R@10	MRR	R@1	R@5	R@10	MRR
Sparse	BM25	12.3	28.7	38.2	0.201	10.8	26.1	35.9	0.184
2*Zero-Shot	OpenAI Emb.	45.6	72.1	81.5	0.572	41.2	68.3	78.9	0.531
	BGE Emb.	38.9	65.4	76.8	0.508	36.5	62.1	73.2	0.482
3*Fine-Tuned	Hybrid (Text FT + OpenAI)	55.2	79.8	87.3	0.654	51.7	76.9	85.1	0.623
	Qwen2.5-Math (FT)	59.1	82.4	89.2	0.689	55.8	80.1	87.9	0.661
	Our Model	**63.8**	**85.2**	**91.1**	**0.731**	**59.4**	**82.7**	**89.5**	**0.693**

line represents a strong, domain-specialized competitor. Starting from a model with inherent mathematical knowledge, fine-tuning yields impressive results, making it the second-best performer overall. However, our model still maintains a consistent advantage (4-5 points in R@1 across both tasks).

5 Discussion

5.1 Summary of Key Findings

Our study yields three principal conclusions. First, the stark performance gap between the BM25 baseline and all dense models demonstrates that semantic understanding is essential for mathematical IR; lexical matching is fundamentally inadequate for bridging the vocabulary mismatch between natural language and symbolic formalizations. Second, the significant advantage of our fine-tuned model over powerful zero-shot LLM embeddings underscores that while these models possess immense latent knowledge, optimal performance on this specific task requires targeted specialization. Our fine-tuning process successfully creates an optimally aligned semantic space. Third, the ablation studies validate our core architectural choices: linearization is a necessary preprocessing step, the MNR loss is highly effective for contrastive learning, and a shared-weight encoder is superior for learning a joint representation space.

5.2 Theoretical and Practical Implications

Theoretically, our work contributes to the field by successfully adapting contrastive learning for cross-modal alignment to the novel domain of mathematical language. The error analysis, particularly the prevalence of "variable mismatch" errors in zero-shot models, offers a fascinating insight into how these models perceive mathematics: they often prioritize overall formula structure over the specific identities of variables, a tendency our fine-tuning process mitigates. Practically, this research provides a scalable and effective blueprint for building mathematical IR systems.

5.3 Limitations

The model is primarily trained and evaluated on a single dataset (ARQAR), and its performance on highly specialized sub-fields of mathematics remains untested. Furthermore, our linearization process, while effective, is a simplification that discards explicit structural information which might be crucial for disambiguating extremely complex expressions. Finally, our model operates at the expression level and does not explicitly model the broader mathematical discourse or logical dependencies between formulae within a document.

6 Conclusion

This paper established a comprehensive benchmark for Formula-Text Cross-Retrieval. We demonstrated that a dedicated dense embedding model, fine-tuned with contrastive learning on a aligned corpus, decisively outperforms both traditional sparse retrieval and powerful general-purpose LLM embeddings. Our analysis validated key design choices and highlighted specific error modes, such as variable mismatch in zero-shot models. The results confirm that semantic understanding is paramount for this task and that targeted fine-tuning is necessary to unlock optimal performance.

References

Akiko Aizawa, Susumu Fujita, Noriko Kando, Yasushi Motoki, Akiko Takano, and Yoshiaki Watanabe. 2016. Ntcir-12 mathir task overview. *NTCIR*.

Akiko Aizawa, Michael Kohlhase, Masao Ohta, Koji Mineshima, and Yoshinari Morimoto. 2014. Overview of ntcir-11 math-2 task. *NTCIR*, 11:1–5.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Santos, Stephen McAleer, Albert Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics.

Mark Chen, Jerry Tworek, Honghao Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jerry Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Kenton Guu, Kelvin Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, Ray Lukás, and 1 others. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weiming Chen. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Sining Huang, Yukun Song, Yixiao Kang, and Chang Yu. 2024. Ar overlay: Training image pose estimation on curved surface in a synthetic way. *arXiv preprint arXiv:2409.14577*.

Geoffrey Irving, Christian Szegedy, Alexander A Alemi, Niklas Eén, François Chollet, and 1 others. 2016. Deepmath-deep sequence models for premise selection. *Advances in Neural Information Processing Systems*, 29.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Patrice Lopez and Abdou Youssef. 2014. Query expansion for searched mathematical expressions. In *International Conference on Intelligent Computer Mathematics*, pages 383–387. Springer.

Zhichao Ma, Yutong Luo, Zheyu Zhang, Aijia Sun, Yinuo Yang, and Hao Liu. 2025. Reinforcement learning approach for highway lane-changing: Ppo-based strategy design. In *2025 10th International Conference on Electronic Technology and Information Science (ICETIS)*, pages 298–301.

Keiran Paster. 2022. Can number theory help you design your neural network? a case study in embeddings. *URL https://blog. me/paper/embedding. pdf*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Shideh Seyedi, Nidhi Kulkarni, Arian Kordzadeh, Frank Vasile, and 1 others. 2024. Arqar: A dataset for auto-regressive question answering and reasoning. `https://github.com/IBM/arqar`.

Zhihong Shen, Chen Wu, Jialin Su, Yan Wang, and 1 others. 2020. Formula retrieval using tree representation. In *European Conference on Information Retrieval*, pages 389–403. Springer.

Qwen Team. 2024. Qwen2.5-math: Scaling reasoning in mathematical domains. *arXiv preprint arXiv:2409.XXXXX*. Model card and technical report.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Chao Zhao. 2023. C-pack: Packaged resources to advance general chinese embedding. `https://github.com/FlagOpen/FlagEmbedding`.

Shuang Zeng, Dekang Qi, Xinyuan Chang, Feng Xiong, Shichao Xie, Xiaolong Wu, Shiyi Liang, Mu Xu, and Xing Wei. 2025. Janusvln: Decoupling semantics and spatiality with dual implicit memory for vision-language navigation. *arXiv preprint arXiv:2509.22548*.

Le Zhao, Yan Fang, Yue Liu, Liang He, Yuxin Wang, and Yizhi Wang. 2014. A math-aware search engine for math question answering system. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1016–1021.

BanglaMATH : A Bangla benchmark dataset for testing LLM mathematical reasoning at grades 6, 7, and 8

Tabia Tanzin Prama[1,2,3,5], **Christopher M. Danforth**[1,2,3,4], **Peter Sheridan Dodds**[1,2,3,5,6]
[1]Computational Story Lab, [2]Vermont Complex Systems Institute,
[3]Vermont Advanced Computing Center,
[4]Department of Mathematics and Statistics, [5]Department of Computer Science,
University of Vermont, Burlington, VT 05405, USA
[6]Santa Fe Institute, 1399 Hyde Park Rd, Santa Fe, NM 87501, USA

Abstract

Large Language Models (LLMs) have tremendous potential to play a key role in supporting mathematical reasoning, with growing use in education and AI research. However, most existing benchmarks are limited to English, creating a significant gap for low-resource languages. For example, Bangla is spoken by nearly 250 million people who would collectively benefit from LLMs capable of native fluency. To address this, we present BanglaMATH, a dataset of 1.7k Bangla math word problems across topics such as Arithmetic, Algebra, Geometry, and Logical Reasoning, sourced from Bangla elementary school workbooks and annotated with details like grade level and number of reasoning steps. We have designed BanglaMATH to evaluate the mathematical capabilities of both commercial and open-source LLMs in Bangla, and we find that Gemini 2.5 Flash and DeepSeek V3 are the only models to achieve strong performance, with $\geq$ 80% accuracy across three elementary school grades. Furthermore, we assess the robustness and language bias of these top-performing LLMs by augmenting the original problems with distracting information, and translating the problems into English. We show that both LLMs fail to maintain robustness and exhibit significant performance bias in Bangla. Our study underlines current limitations of LLMs in handling arithmetic and mathematical reasoning in low-resource languages, and highlights the need for further research on multilingual and equitable mathematical understanding. Dataset link: https://github.com/BanglaMATH

1 Introduction

Mathematical reasoning is one of the cornerstones of human intelligence and remains a critical focus in the pursuit of artificial intelligence (AI). As AI continues to evolve, empowering machines with a deep and comprehensive understanding of mathematics not only showcases technological advancement but also represents a key milestone toward developing more generalized and capable AI systems. Recently, the emergence of Large Language Models (LLMs) has significantly reshaped the AI landscape, establishing them as powerful tools for automating complex tasks. LLMs have demonstrated remarkable proficiency in mathematical problem-solving (Romera-Paredes et al., 2023; Imani et al., 2023), prompting extensive evaluations of their capabilities across a variety of domains (Liu et al., 2023; Deng et al., 2024; Wu et al., 2023). Notably, LLMs such as ChatGPT (Ouyang et al., 2022) along with (Taylor et al., 2022) have shown impressive performance in generating and interpreting natural language, while the more recent GPT-4 (OpenAI, 2023; Bubeck et al., 2023) has set new standards in both linguistic and logical tasks.

The capacity to understand and solve mathematical problems is an especially desirable trait for LLMs, with wide-ranging applications in education, science, and industry. However, evaluating their mathematical competence is inherently challenging. While numerous benchmark datasets have been developed to assess mathematical reasoning (Cobbe et al., 2021; Amini et al., 2019; Hendrycks et al., 2021), most are limited to the English language, with a few available in Chinese (Zhang et al., 2023a,b; Zhao et al., 2020; Zhou et al., 2023). Mathematical reasoning in Bangla, despite the widespread use of LLMs by Bangla-speaking communities for educational purposes, remains largely unexplored.

To address this gap, we introduce the Bangla Mathmatical Benchmark (BanglaMATH) dataset which consists of 1.7k elementary school level math problems from Bangladesh, the first benchmark dataset designed specifically to evaluate mathematical reasoning in Bangla. The BanglaMATH dataset comprises 1.7k elementary-

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 134–149

level math word problems sourced from authentic Bangla workbooks and examination materials. The dataset includes multiple-choice questions, logical puzzles, and descriptive reasoning problems, each annotated with question type metadata to facilitate fine-grained evaluation. In Figure 1, we display an example problem from the dataset, in Bangla and English, with the correct (human) answer and and an incorrect one from ChatGPT. We posit that the evaluation of LLMs should mirror that of human learners, which would allow us to convey results in a manner that is more intuitive and accessible.

We assess the performance of several widely used Large Language Models (LLMs)—including both commercial APIs and open-source models—on the BanglaMATH dataset. Each problem in BanglaMATH is annotated with grade-level information, allowing us to conduct fine-grained evaluations similar to stating "ChatGPT scored 72 out of 100 in a sixth-grade math exam."

Our results show that Gemini 2.5 Flash and DeepSeek V3 consistently achieve high performance (accuracy $\geq$ 80%) across all three elementary school grade levels. We evaluate model performance across varying levels of arithmetic and reasoning complexity, and observe that accuracy significantly decreases as problem complexity increases. We also examine the robustness of these models by injecting distracting information into the math problems. Our findings reveal that both of the top-performing models (Gemini 2.5 Flash and DeepSeek V3) are easily misled by the presence of such irrelevant information, resulting in incorrect reasoning and answers. To further investigate language bias in LLMs, we translate the BanglaMATH dataset into English and re-evaluate the top-performing models (Gemini 2.5 Flash and DeepSeek V3). Interestingly, both models show an improvement in accuracy by $\geq$ 6.5% when tested on the English-translated version, highlighting a performance disparity for low-resource languages like Bangla.

2 BanglaMATH Dataset

We can frame our work as being driven by the question:

> Do LLMs have the ability to perform mathematical reasoning in Bangla?

To address this, we introduce the BanglaMATH dataset, specifically designed to evaluate LLMs on a diverse set of mathematical problems written in Bangla. Our focus spans across various categories of mathematics, including Arithmetic, Algebra, Geometry, and Logical Reasoning, to provide a comprehensive assessment of LLMs' general-purpose reasoning and arithmetic capabilities.

2.1 Data Collection

The mathematical question dataset is compiled from a wide range of elementary school level sources, such as school exercise books, quizzes, and exam equations. The original materials are primarily in PDF or Microsoft Word formats. These were converted into plain text using automated tools where possible, with manual transcription performed by human annotators when necessary. Since our interest lies solely in text-based word problems, we excluded any questions requiring visual or graphic interpretation. All collected questions undergo a rigorous pre-processing pipeline, including de-duplication and cleaning. This is followed by multiple rounds of human validation by the authors to ensure the quality and accuracy of the dataset.

Data annotation. Each problem in the dataset is annotated with several attributes, including grade level, answer, number of reasoning steps, explanations, and the number of digits involved. We provide a statistical summary of the BanglaMATH benchmark dataset in Table 1.

Grade	Size	Steps	Digits	Length
Eight	516	2.19	2.26	10.17
Seven	679	1.82	2.09	8.99
Six	508	1.69	1.41	7.78

Table 1: Statistics of the BanglaMATH dataset. "Length" denotes the average problem length in terms of the number of words. "Steps" indicates the average number of reasoning steps required to solve a problem. "Digits" refers to the average number of digits involved in each problem's solution.

Grade. Each math problem in the dataset is annotated with its corresponding elementary school grade. This information was collected while gathering subsets of problems specific to each grade, enabling more targeted and accurate grade-level evaluation.

Ground Truth Answer. For each problem, we also annotate the ground-truth answer. Most of the answers are standalone numericals—integers, decimals, fractions, or percentages. A small propor-

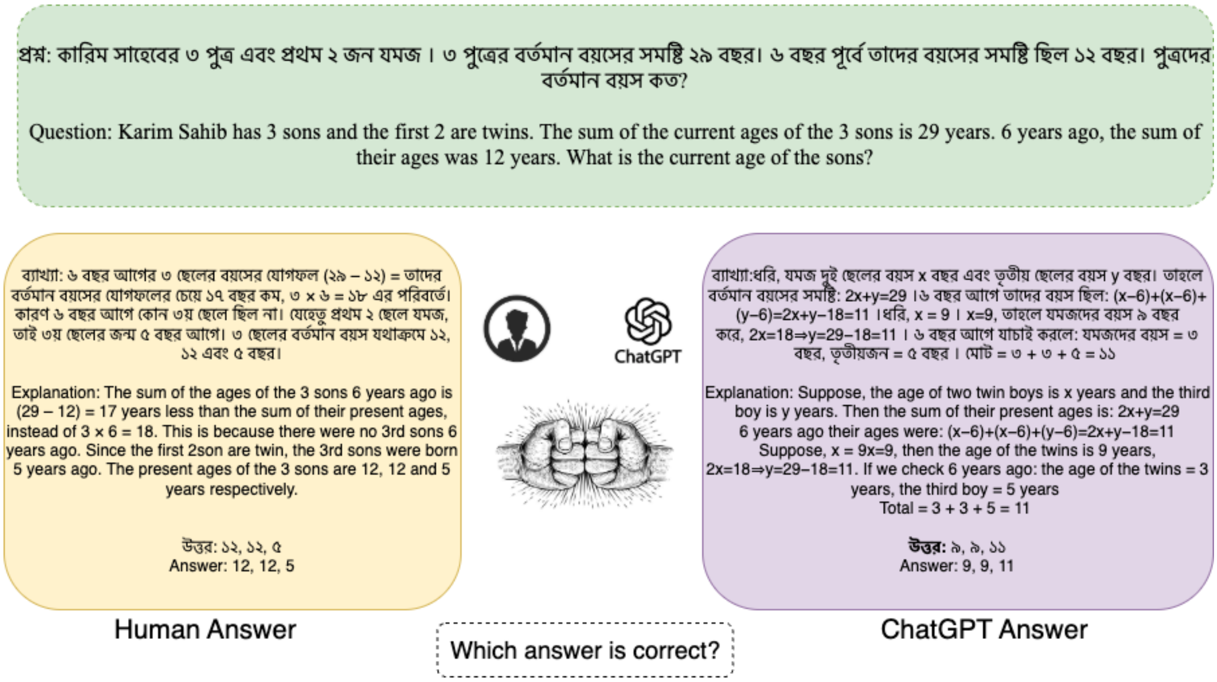


Figure 1: An example from the BanglaMATH dataset illustrating a discrepancy between the correct human-provided explanation and the incorrect response generated by ChatGPT-4o (as of May 18, 2025).

tion of questions, however, have descriptive answers. Among all problems, 9.63% require descriptive answers, while 90.37% can be answered with a single-word numerical response. We do not provide the reasoning process leading to the answer, as our dataset is intended solely for test purposes.

Number of Steps. Additionally, for each problem, we manually annotate the number of reasoning steps required to solve it. For the majority of problems, human annotators can easily reach consensus on the number of steps. In a few cases, the number of steps may vary depending on the specific solution considered, but this ambiguity affects only a small fraction of the dataset and should not pose a significant issue. We use the number of reasoning steps as a proxy for a problem's complexity, which reflects the level of logical analysis and problem-solving strategies required for a language model to arrive at the correct solution. Generally, more reasoning steps correspond to a more intricate thought process and provide more opportunities for the model to make errors or lose track of the problem's structure.

Number of Digits. To determine the number of digits, we identify the maximum base-10 length of any number appearing in the problem using Algorithm 1. Only digits 0–9 are counted; other symbols, such as decimal points, percent signs, and slashes, are ignored.

Algorithm 1 Compute Maximum Digit Count in a Math Word Problem

```
function max_digit_count(problem_statement, answer)
    N ← extract all numbers from problem_statement as strings
    Append str(answer) to N
    digit_counts ← [ len(x) for each x ∈ N ]
    D ← max(digit_counts)
    return D
end function
```

For example, in the problem statement shown in Figure 1, the maximum number is 29, so the digit count is 2. Table 2 shows the annotated samples of BanglaMATH dataset.

3 Experimental Setup

3.1 Models

We evaluate a range of widely used LLMs capable of processing Bangla text and fine-tuned for general-purpose reasoning tasks. These models, developed by various organizations, differ in architecture, size, and access methods—some are available via APIs, others through open-source model weights.

Table 2: Sample problems with their English translations (not part of the dataset) and human annotations. The columns "Answer," "#Steps," and "#Digits" refer to the ground truth answer, number of reasoning steps, and the maximum number of digits in the problem, respectively

Grade	Question (Bangla)	Question (English)	Answer (Bangla)	Answer (English)	#Steps	#Digits
6	ডাক্তার এক রোগিকে সকাল ৬টা থেকে দুপুর ১২টা পর্যন্ত আধ ঘন্টা পরপর ১টি টেবলেট খেতে বললেন। রোগীর মোট কতটি টেবলেট প্রয়োজন?	The doctor asked a patient to take 1 tablet every half hour from 6 AM to 12 PM. How many tablets does the patient need in total?	১৩ টা	13	2	2
7	একটি ৫০০ মিটার লম্বা ট্রেনের গতি ৬০ কিলোমিটার হলে ,অর্ধকিলোমীটার লম্বা একটি সেতু পাড়ি দিতে ট্রেনটির কতক্ষণ সময় লাগবে?	If the speed of a 500 m long train is 60 km/h, how much time will it take to cross a half km long bridge?	১ মিনিট	1 minute	2	3
8	প্রথম সাতটি মৌলিক সংখ্যার গড় নির্ণয় কর। (ক) 5.60 (খ) 8.28 (গ) 7.42 (ঘ) 6.84	Find the average of the first seven prime numbers. (a) 5.60 (b) 8.28 (c) 7.42 (d) 6.84	খ	B	3	1
8	যখন একটি আর্টিকেল 20% লাভে বিক্রি হয়, তখন এটি থেকে যখন এটি 20% হারে বিক্রি হয় তার চেয়ে 60 টাকায় বেশি বিক্রি হয়। আর্টিকেলটির দাম কত? (ক) 150 টাকা (খ) 200 টাকা (গ) 140 টাকা (ঘ) 120 টাকা	When an article sells at 20% profit, it sells for Rs.60 more than when it sells at 20%. How much does the article cost? (a) Rs.150 (b) Rs.200 (c) Rs.140 (d) Rs.120	ক	A	3	3
6	কোনো সংখ্যা অঙ্কের সাহায্যে লেখাকে অঙ্কপাতন বলে?	What is Notation?	কোনো সংখ্যা অঙ্কের সাহায্যে লেখাকে অঙ্কপাতন বলে	system of written symbols used to represent numbers	1	-

- GPT-4 (OpenAI, 2023) (released on March 14, 2023) is a multimodal LLM that demonstrates human-level performance across a range of professional and academic benchmarks. Based on the Transformer architecture, it is pre-trained to predict the next token in a sequence and is capable of analyzing, reading, and generating up to 25,000 words (32,768 tokens) per input. The model is estimated to have 1.76 trillion parameters which is accessible via ChatGPT Plus and the OpenAI API. In this experiment, we utilized the OpenAI API to access and evaluate the GPT-4 model.

- LLaMA 4 (AI, 2025) (released April 5, 2025) is the latest model from Meta AI designed to enable more personalized and natively multimodal experiences. LLaMA 4 is available in two main versions: Llama 4 Scout, a 17-billion active parameter model with 16 experts, and Llama 4 Maverick, a 17-billion active parameter model with 128 experts. Llama 4 Maverick is regarded as the best multimodal model in its class, outperforming GPT-4o and Gemini 2.0 Flash and DeepSeek v3 in reasoning and coding tasks. In this experiment, we evaluate the Llama 4 Maverick model using the Meta.AI [1] website.

- Gemini 2.5 Flash (Gemini Team, 2025) (released June 17, 2025) is the latest model from Google, featuring a maximum input of 1,048,576 tokens and a default maximum output of 65,535 tokens and built on a sparse mixture-of-experts (MoE) (Clark et al., 2022) Transformer architecture. It incorporates advanced reasoning capabilities, allowing users to observe the model's "thinking process" as it generates responses. Gemini 2.5 Flash also introduces agentic AI, supports real-time applications, and is optimized for large context processing with up to 1 million input tokens. The model is accessible through Google AI Studio and Vertex AI. For this experiment, we evaluated Gemini 2.5 Flash using the Google AI Studio [2] platform.

- Grok 3 (released February 17, 2025) is the latest AI model from xAI, combining transformer-based language modeling with symbolic reasoning modules in a 1.2 trillion parameter architecture (Inaba et al., 2003). Grok 3 employs 128 expert networks with dynamic routing, enabling specialized processing for different task types while maintaining 83% parameter activation efficiency (Doshi et al., 2023). Unlike traditional mixture-of-experts (MoE) models, Grok 3 introduces cross-expert attention gates, allowing knowledge sharing between specialized components without catastrophic interference. The training corpus comprises 13.4 trillion tokens which is accessible via official website and X. For this experiment, we use Grok 3 through official website[3].

- DeepSeek-V3 (DeepSeek-AI et al., 2024) (released December 26, 2024) is a Mixture-of-Experts (MoE) language model with 671 billion total parameters, of which 37 billion are activated for each token. DeepSeek-V3 utilizes Multi-head Latent Attention (MLA) and the DeepSeekMoE architecture, building on advances from DeepSeek-V2, to enable efficient inference and cost-effective training. The model is pre-trained on 14.8 trillion tokens, and optimized through supervised fine-tuning and reinforcement learning. It can be accessed directly via chat on the official website and API through the DeepSeek Platform. For this experiment, we evaluated DeepSeek-V3 using the official website.

[1] https://ai.meta.com/
[2] https://deepmind.google/
[3] https://grok.com

Direct Prompting (DP)
তোমাকে একটি গণিত প্রশ্ন দেওয়া হবে বাংলায়। প্রশ্নটি বুঝে শুধুমাত্র সেই প্রশ্নের উত্তর বাংলায় দাও। ব্যাখ্যা, ধাপ লিখবে না — কেবলমাত্র প্রশ্নের উত্তর বাংলায় দাও। প্রশ্ন: {question} উত্তর: {Answer}

Translated Prompt
You will be given a math question in Bengali. Understand the question and provide only the answer in Bengali. Do not write any explanation—only give the answer Question: {question} Answer: {Answer}

Figure 2: Example prompt provided to the LLMs. Note that Bengali is an alternative name for the language Bangla.

3.2 Evaluation Procedure

In our experiments, we employ a zero-shot evaluation approach, refraining from using any auxiliary prompting strategies. Each math problem is presented to the LLMs in its original Bangla text format, without any additional context, examples, or instructions, as shown in Figure 2.

We deliberately choose zero-shot evaluation to reflect a realistic and practical deployment scenario. Since the LLMs considered in this study are fine-tuned for general-purpose use and are designed to function out-of-the-box, we argue that zero-shot evaluation offers a more reliable measure of their baseline capabilities.

Finally, we calculate accuracy scores as part of our evaluation. To assess performance, we extract the numerical answer(s) generated by the model and compare with the annotated ground truth answers. We also conduct manual verification of the LLM-generated outputs and their corresponding ground truth answers to ensure the accuracy of our evaluation, and to account for possible formatting inconsistencies or ambiguities in the responses. An LLM response is marked correct if there is an exact numerical match for single-word numerical responses, or if descriptive answers are judged to be similar in meaning.

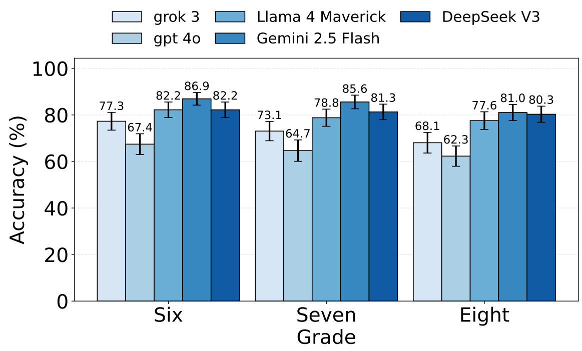


Figure 3: Average test accuracy of LLMs mathematical reasoning on the BanglaMATH dataset for grades 6, 7, and 8

4 Result and Analysis

4.1 Grade level accuracy

The test results [4] are presented in Figure 3, illustrating the accuracy of each model. A notable downward trend in accuracy can be observed, indicating that the performance of all models declines when grade level is increased. Although this outcome is somewhat anticipated, given that higher grade math problems generally present greater difficulty, it is still surprising to observe that half of the models struggle even at grade 6. Among the models, DeepSeek V3 and Gemini 2.5 Flash emerge as the models achieving consistent success—defined as accuracy exceeding 80%—across all question types.

LLaMA 4 demonstrates strong performance on grade 6, but struggles with grades 7 and 8. Grock 3 succeeds in grades six and seven (accuracy exceeding 70%) but fails for grade eight. GPT-4 fails across all grades, having accuracy less than 70%. For certain math problems, all five models fail to produce the correct answer. Appendix A.2 presents examples where the responses from all five LLMs do not match the annotated ground truth.

Overall, the results reveal that although these math word problems are considered relatively simple for an average human adult, they continue to pose significant challenges for general-purpose open-source LLMs, particularly when presented in Bangla.

4.2 Arithmetic Complexity

We examine how arithmetic complexity affects the ability of LLMs to solve elementary math word problems. During annotation, we focus on arithmetic complexity of math problems, approximated by the maximum number of digits in the numbers involved. Intuitively, higher arithmetic or reasoning complexity should make problems more difficult and reduce model accuracy.

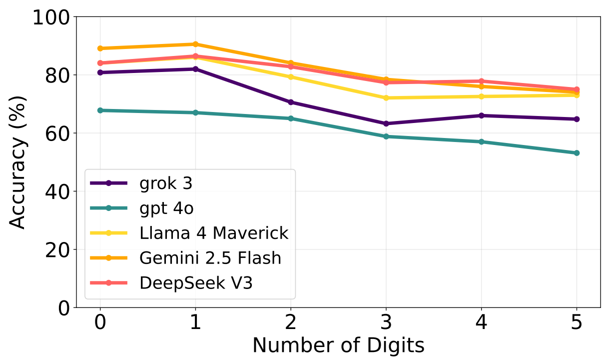


Figure 4: Average test accuracy for each LLM on the BanglaMATH dataset, based on the number of digits in the problem.

In Figure 4, we show the average test accuracy of each model on BanglaMATH, grouped by the number of digits in the problems. We observe that as the number of digits increases, the accuracy for all models drops. For problems involving zero to two digits, DeepSeek-V3, Gemini 2.5 Flash, and LLaMA 4 achieve over 80% accuracy. However, for Grok 3 and GPT-4, accuracy falls below 70% once problems require more than two-digit arithmetic. The lowest accuracies are seen with five-digit problems: among all model DeepSeek-V3 and Gemini 2.5 Flash achieving highest accuracy $\approx$ 75% while GPT-4 accuracy lowest accuracy 58.13%. Overall, the trend clearly shows that LLMs make more mistakes as the arithmetic complexity, measured by the number of digits, in-

[4] Results are obtained early June, 2025.

creases.

4.3 Reasoning Complexity

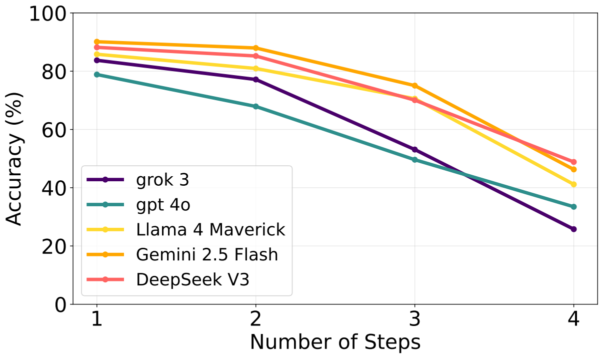


Figure 5: Average test accuracy for each LLM on the BanglaMATH dataset, based on the number of reasoning steps required to solve each problem.

To evaluate how reasoning complexity influences the ability of state-of-the-art LLMs to solve math problems, we determined the number of reasoning steps required to solve each problem during the annotation process. Naturally, problems that require a higher number of steps are considered to have greater reasoning complexity.

As shown in Figure 5, model performance drops significantly as reasoning complexity increases. When a problem requires only one or two reasoning steps, all LLMs perform consistently well, with accuracy $> 70\%$. However, for problems requiring four or more reasoning steps, the accuracy of all models falls below $< 50\%$. This pattern clearly demonstrates that LLMs are much more likely to make mistakes as reasoning complexity increases.

4.4 Robustness

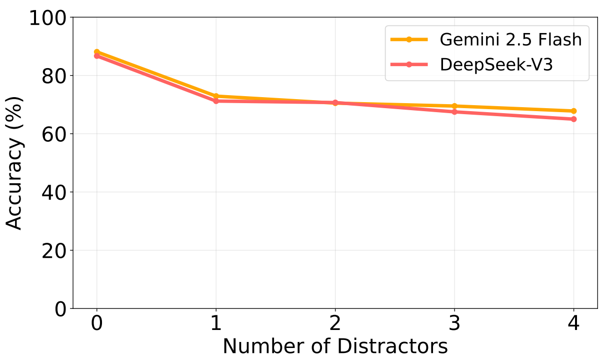


Figure 6: Test accuracy for two LLMs (Gemini 2.5 Flash and Deepseeks v3) against the number of distractors on the subset of BanglaMATH dataset.

We evaluate the robustness of the two top-performing LLMs (Gemini 2.5 Flash and DeepSeek V3) against irrelevant information. Here, irrelevant information refers to details that are related to the context of the problem but are inconsequential to its solution. This type of robustness is especially important, as real-world problems rarely appear in an idealized form where all provided information is necessary. Therefore, it is essential for LLMs to distinguish relevant from irrelevant information and use only the pertinent details to arrive at a correct solution. To test this ability, we manually construct a small distractor dataset containing 60 examples—20 from each grade level. Each example consists of an original problem along with four modified versions that include 1 to 4 pieces of irrelevant information, referred to as distractors. Each distractor includes exactly one number and integrates seamlessly into the original problem context, making it appear contextually appropriate. We test both models on this distractor dataset and observe a significant drop in performance as the number of distractors increases (see Figure 6). Both models experience an accuracy drop of approximately 20% when only two distractors are added. Notably, DeepSeek V3 suffers more degradation than Gemini 2.5 Flash when the number of distractors exceeds two.

In Appendix A.1, Table 3 shows examples of the models' responses to distractor-augmented problems and reveal that the models behave differently in the presence of irrelevant information. Both models adjust their reasoning based solely on the added irrelevant details, often leading to incorrect conclusions. Based on these results, we conclude that neither model demonstrates strong robustness to irrelevant information. This exposes a critical weakness: even advanced LLMs struggle to filter out distractions, highlighting the ongoing need for improved contextual understanding in real-world problem-solving tasks.

4.5 Language Bias

LLMs often exhibit decreased performance when operating in languages other than English (Dey et al., 2024). To investigate potential bias toward low-resource languages like Bangla, we conducted an experiment using a subset of 60 samples as distractor dataset from the BanglaMATH dataset. Each mathematical question was translated from Bangla to English, and we evaluated the performance of Gemini 2.5 Flash and DeepSeek V3

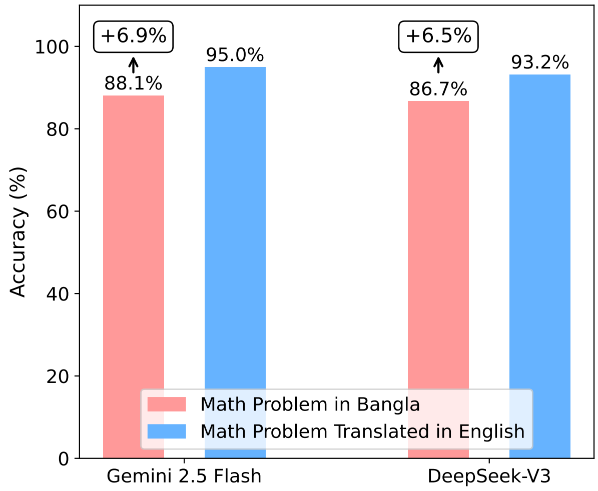


Figure 7: Test accuracy of Gemini 2.5 Flash and DeepSeek V3 on BanglaMATH subset, comparing original Bangla problems and English translations.

on both the original and translated versions. As shown in Figure 7, both models demonstrate a performance improvement of approximately 6.5% after translation. Notably, Gemini 2.5 Flash consistently outperforms DeepSeek V3 in both Bangla and English versions. However, the substantial accuracy gain following translation underscores a significant language bias—revealing that these LLMs reason more effectively in English than in Bangla. This disparity highlights a critical shortcoming: current LLMs trained on low-resource languages like Bangla do not yet match their English-language capabilities. Our findings emphasize the urgent need to enhance the mathematical reasoning abilities of LLMs across diverse linguistic contexts, especially for underrepresented and low-resource languages.

5 Related Works

Math-related datasets are available predominantly in English (Hendrycks et al., 2021; Amini et al., 2019; Cobbe et al., 2021), making them unsuitable for evaluating the reasoning abilities of LLMs in Bangla. Traditional math word problem (MWP) datasets like AddSub (Hosseini et al., 2014) and MultiArith (Roy and Roth, 2016) are integrated into broader MWP repositories. Other similar datasets include SingleEq (Koncel-Kedziorski et al., 2015), AQUA (Ling et al., 2017), and AsDiv (Miao et al., 2020). GSM8K (Clark et al., 2022) and SVAMP (Patel et al., 2021) take advantage of detailed annotations and have prevailed in recent evaluations of LLMs. Similarly, MATH dataset (Hendrycks et al., 2021), which collects problems from American high school mathematics competitions and categorizes them into seven subjects: Pre-algebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Pre-calculus. However, these problems are extremely challenging—even for humans, the accuracy rate is only 40%. Given that many LLMs are still in early stages, using overly difficult problems may have limited utility in evaluating their capabilities.

Other than English, several Chinese and a Hindi (Sharma et al., 2022) math-related datasets exist. AGI-Eval (Zhong et al., 2023) and C-Eval (Huang et al., 2023) target general-purpose, multidisciplinary evaluation for LLMs and contain subsets specifically designed to assess mathematical abilities. The math problems in these datasets range from middle school to college level and are often quite complex. Similarly, Math23K (Wang et al., 2017) and APE210K (Zhao et al., 2020), CMATH (Wei et al., 2023) and K6 (Yang et al., 2023), comprise elementary school-level math word problems. CMATH and K6, are two datasets that are relatively similar to ours and are developed concurrently. Both focus on math word problems from elementary school and organize instances by grade level.

Our work is most direclty inspired by the CMATH dataset, which contains 1.7K problems collected from online workbooks and exams, while K6 comprises 600 problems collected from an educational institution. However, neither of these two datasets has been publicly released, which prevents us from conducting an empirical comparison with them. Additionally, APE210K, contains an enormous 210K Chinese math word problems from elementary school. Its test set alone includes as many as 5,000 problems. However, the test sets do not provide annotations specific to LLM evaluation.

6 Conclusion

We have introduced BanglaMATH, a novel dataset designed to enable fine-grained evaluation of Large Language Models (LLMs) on elementary-level mathematical word problems in Bangla. To the best of our knowledge, BanglaMATH is the first Bangla mathematical benchmark dataset designed to help evaluate the mathematical reasoning abilities of LLMs. Our results show that

BanglaMATH poses a significant challenge even for state-of-the-art LLMs, with performance declining as the grade level increases. Additionally, we observe that as the arithmetic and reasoning complexity of problems increases, the accuracy of all evaluated models decreases. Our robustness analysis reveals that top-performing models such as Gemini 2.5 Flash and DeepSeek V3 struggle when faced with irrelevant or distracting information, highlighting a vulnerability in real-world problem comprehension. Furthermore, we uncover a clear language bias—performance improves when Bangla problems are translated into English. This suggests that current LLMs are less effective at reasoning in low-resource languages like Bangla compared to English. We believe BanglaMATH can serve as a valuable benchmark for advancing non-English mathematical reasoning research, and help evaluate the cross-linguistic mathematical reasoning capabilities of LLMs.

7 Limitations

While we have made a strong effort to evaluate the Bangla mathematical reasoning abilities of various LLMs, due to budget constraints and restricted API access, we are unable to assess all available LLMs. We encourage future work to expand on this study by including a broader range of models, especially as more open-source and multilingual LLMs become accessible. Our focus in this work is on evaluating mathematical understanding and the results underscore that mathematical reasoning—particularly in non-English contexts and in the presence of real-world distractions—remains a significant challenge, even for state-of-the-art models. We hope that BanglaMATH serves as a valuable resource for future research aimed at improving multilingual mathematical reasoning and helps guide the development of LLMs that are robust, linguistically inclusive, and capable of solving real-world math problems effectively.

8 Ethics Statement

All problems in the BanglaMATH dataset are based on publicly available elementary school-level materials. No human subjects or sensitive data were involved during the annotation process; all annotations rely solely on the authors' judgment. We acknowledge that certain labels, such as the number of steps in a solution, may vary based on individual interpretation. We ensure responsible use of LLMs and adhere to ethical guidelines in data handling and model evaluation.

References

Meta AI. 2025. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/?utm_source=twitter&utm_medium=organic_social&utm_content=image&utm_campaign=llama4. Released April 2025. Accessed: July 2025.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*.

A. Clark, D. de las Casas, A. Guy, A. Mensch, M. Paganini, J. Hoffmann, B. Damoc, B. Hechtman, T. Cai, S. Borgeaud, G. van den Driessche, E. Rutherford, T. Hennigan, M. Johnson, K. Millican, A. Cassirer, C. Jones, E. Buchatskaya, D. Budden, and 7 others. 2022. Unified scaling laws for routed language models. *arXiv preprint arXiv:2202.01169*.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bing-Li Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 179 others. 2024. DeepSeek-V3 technical report. *ArXiv*, abs/2412.19437.

Naihao Deng, Zhenjie Sun, Ruiqi He, Aman Sikka, Yulong Chen, Lin Ma, Yue Zhang, and Rada Mihalcea. 2024. Tables as images? Exploring the strengths and limitations of LLMs on multimodal representations of tabular data. *arXiv preprint arXiv:2402.12424*.

Krishno Dey, Prerona Tarannum, Md. Arid Hasan, Imran Razzak, and Usman Naseem. 2024. Better to ask in English: Evaluation of large language models on English, low-resource and cross-lingual settings. *ArXiv*, abs/2410.13153.

Darshil Doshi, Aritra Das, Tianyu He, and Andrey Gromov. 2023. To grok or not to grok: Disentangling generalization and memorization on corrupted algorithmic datasets. *ArXiv*, abs/2310.13061.

Google Gemini Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf. Released June 17, 2025. Accessed: July 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Conference on Empirical Methods in Natural Language Processing*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Fanchao Qi, Yao Fu, Maosong Sun, and Junxian He. 2023. C-Eval: A multi-level multi-discipline Chinese evaluation suite for foundation models. *ArXiv*, abs/2305.08322.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 37--42.

Tsuneo Inaba, Kenji Tsuchida, Tadahiko Sugibayashi, Shuichi Tahara, and Hiroaki Yoda. 2003. Resistance ratio read (R/sup 3/) architecture for a burst operated 1.5V MRAM macro. *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference, 2003.*, pages 399--402.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585--597.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Annual Meeting of the Association for Computational Linguistics*.

Siyang Liu, Naihao Deng, Sahand Sabour, Yilin Jia, Minlie Huang, and Rada Mihalcea. 2023. Task-adaptive tokenization: Enhancing long-form text generation efficacy in mental health and beyond. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 15264--15281, Singapore. Association for Computational Linguistics.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Annual Meeting of the Association for Computational Linguistics*.

OpenAI. 2023. Gpt-4 technical report. Technical report, OpenAI.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Arkil Patel, S. Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *North American Chapter of the Association for Computational Linguistics*.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, and 1 others. 2023. Mathematical discoveries from program search with large language models. *Nature*, pages 1--3.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *ArXiv*, abs/1608.01413.

Harshita Sharma, Pruthwik Mishra, and Dipti Misra Sharma. 2022. HAWP: A dataset for Hindi arithmetic word problem solving. In *International Conference on Language Resources and Evaluation*.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Conference on Empirical Methods in Natural Language Processing*.

Tianwen Wei, Jian Luan, W. Liu, Shuang Dong, and Bin Quan Wang. 2023. CMATH: Can your language model pass chinese elementary school math test? *ArXiv*, abs/2306.16636.

Yufan Wu, Yinghui He, Yilin Jia, Rada Mihalcea, Yulong Chen, and Naihao Deng. 2023. HiToM: A benchmark for evaluating higher-order theory of mind reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10691--10706, Singapore. Association for Computational Linguistics.

Z. Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. 2023. GPT can solve mathematical problems without a calculator. *ArXiv*, abs/2309.03241.

Beichen Zhang, Kun Zhou, Xilin Wei, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2023a. Evaluating and improving tool-augmented computation-intensive math reasoning. *arXiv preprint arXiv:2306.02408*.

Mengxue Zhang, Zichao Wang, Zhichao Yang, Weiqi Feng, and Andrew S. Lan. 2023b. Interpretable math word problem solution generation via step-by-step planning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6858--6877.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. APE210K: A large-scale and template-rich dataset of math word problems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied Sanosi Saied, Weizhu Chen, and Nan Duan. 2023. AGIEval: A human-centric benchmark for evaluating foundation models. In *NAACL-HLT*.

Zihao Zhou, Qiufeng Wang, Mingyu Jin, Jie Yao, Jianan Ye, Wei Liu, Wei Wang, Xiaowei Huang, and Kaizhu Huang. 2023. MathAttack: Attacking large language models towards math solving ability. *CoRR*, abs/2309.01686.

A Appendix

A.1 Model Responses Under Distractor Augmentation

Table 3: Math problem with increasing distractors and model responses from Gemini 2.5 Flash and DeepSeek V3. The “# Distractor” column shows how many irrelevant phrases were added. The first row is the original problem; subsequent rows include 1–4 distractors. The table compares how Gemini Flash 2.5 and DeepSeek V3 reasoning changes with added distractions.

# Distractors	Math Problem	Gemini Flash 2.5 Response	DeepSeek V3 Response
0	খেলনা গাড়ি নিয়ে খেলার সময় মিন্টু দেখলো , সেন্টুর গাড়ির সংখ্যা তার চেয়ে বেশি এবং সেন্টু কে বললো, তুমি আমাকে দুইটা গাড়ি দাও তাহলে আমাদের গাড়ির সংখ্যা সমান হবে ।সেন্টু উল্টো জোড় করে মিন্টুর থেকে ২টি গাড়ি নিয়ে হিসাব করে বলল তার গাড়ির সংখ্যা এখন মিন্টুর দ্বিগুণ । শুরুতে কার কয়টি গাড়ি ছিল ? Mintu saw that Sentu had more toy cars than him. He said, “Give me 2 cars and we’ll have the same number.” Sentu instead took 2 cars from Mintu and said, “Now I have double the cars you have.” What were their original counts?	**Answer:** Mintu had 10 cars, Sentu had 14. **Reasoning:** Because if Sentu gave 2 cars, both would have 12; if Sentu took 2 cars, he would have 16 and Mintu 8 — and 16 = 2 × 8.	**Answer:**Mintu: 10, Sentu: 14 **Reasoning:** If Sentu gives 2 cars, both have 12; if Sentu takes 2, he has 16 and Mintu 8 → 16 = 2×8
1	"খেলনা গাড়ি নিয়ে খেলার সময় মিন্টু দেখলো , সেন্টুর গাড়ির সংখ্যা তার চেয়ে বেশি এবং সেন্টু কে বললো, তুমি আমাকে দুইটা গাড়ি দাও তাহলে আমাদের গাড়ির সংখ্যা সমান হবে ।সেন্টু উল্টো জোড় করে মিন্টুর থেকে ২টি গাড়ি নিয়ে হিসাব করে বলল তার গাড়ির সংখ্যা এখন মিন্টুর দ্বিগুণ ।তাদের মোট খেলনা গাড়ির সংখ্যা ২৫টির কম ছিল। শুরুতে কার কয়টি গাড়ি ছিল ? Mintu saw that Sentu had more toy cars than him. He said, Give me 2 cars and we’ll have the same number. Sentu instead took 2 cars from Mintu and said, Now I have double the cars you have. “The total number of toy cars was less than 25 ". What were their original counts?	**Answer:** Mintu had 6 cars, Sentu had 10 **Reasoning:** Mintu initially had 6 cars and Sentu had 10 — their total is 16, which is less than 25, this is the only pair that satisfies both conditions.	**Answer:**Mintu: 10, Sentu: 14 **Reasoning:** Although the total is 24, this is the only pair that satisfies both conditions, so unchanged

2	খেলনা গাড়ি নিয়ে খেলার সময় মিন্টু দেখলো , সেন্টুর গাড়ির সংখ্যা তার চেয়ে বেশি এবং সেন্টু কে বললো, তুমি আমাকে দুইটা গাড়ি দাও তাহলে আমাদের গাড়ির সংখ্যা সমান হবে ।সেন্টু উল্টো জোড় করে মিন্টুর থেকে ২টি গাড়ি নিয়ে হিসাব করে বলল তার গাড়ির সংখ্যা এখন মিন্টুর দ্বিগুণ । তাদের মোট খেলনা গাড়ির সংখ্যা ২৫টির কম ছিল। প্রতিটি গাড়ির দাম ছিল ৩০ টাকা।শুরুতে কার কয়টি গাড়ি ছিল ? Mintu saw that Sentu had more toy cars than him. He said, Give me 2 cars and we'll have the same number. Sentu instead took 2 cars from Mintu and said, Now I have double the cars you have. " The total number of toy cars was less than 25. Each car cost 30 Taka." What were their original counts?	**Answer:** Mintu had 6 cars, Sentu had 10 **Reasoning:**The price is irrelevant; since the number conditions match, the answer remains unchanged.	**Answer:**Mintu: 10, Sentu: 14 **Reasoning:** Price or total cost does not affect the logic; the previous answer remains valid
3	খেলনা গাড়ি নিয়ে খেলার সময় মিন্টু দেখলো , সেন্টুর গাড়ির সংখ্যা তার চেয়ে বেশি এবং সেন্টু কে বললো, তুমি আমাকে দুইটা গাড়ি দাও তাহলে আমাদের গাড়ির সংখ্যা সমান হবে ।সেন্টু উল্টো জোড় করে মিন্টুর থেকে ২টি গাড়ি নিয়ে হিসাব করে বলল তার গাড়ির সংখ্যা এখন মিন্টুর দ্বিগুণ । তাদের মোট খেলনা গাড়ির সংখ্যা ২৫টির কম ছিল। প্রতিটি গাড়ির দাম ছিল ৩০ টাকা।তারা প্রতিদিন ৩ ঘণ্টা খেলনা গাড়ি দিয়ে খেলে। শুরুতে কার কয়টি গাড়ি ছিল ? Mintu saw that Sentu had more toy cars than him. He said, Give me 2 cars and we'll have the same number. Sentu instead took 2 cars from Mintu and said, Now I have double the cars you have. " The total number of toy cars was more than 25. Each car cost 30 Taka.They play 3 hours a day with the cars ". What were their original counts?	**Answer:** Mintu had 6 cars, Sentu had 10 **Reasoning:**playing time is irrelevant; the answer is based solely on the car count logic. Price or total cost does not affect the logic; the previous answer remains valid	**Answer:**Mintu: 10, Sentu: 14 **Reasoning:** Playing time is irrelevant; the answer is determined by the number logic alone

4	খেলনা গাড়ি নিয়ে খেলার সময় মিন্টু দেখলো , সেন্টুর গাড়ির সংখ্যা তার চেয়ে বেশি এবং সেন্টু কে বললো, তুমি আমাকে দুইটা গাড়ি দাও তাহলে আমাদের গাড়ির সংখ্যা সমান হবে ।সেন্টু উল্টো জোড় করে মিন্টুর থেকে ২টি গাড়ি নিয়ে হিসাব করে বলল তার গাড়ির সংখ্যা এখন মিন্টুর দ্বিগুণ । তাদের মোট খেলনা গাড়ির সংখ্যা ২৫টির কম ছিল। প্রতিটি গাড়ির দাম ছিল ৩০ টাকা।তারা প্রতিদিন ৩ ঘণ্টা খেলনা গাড়ি দিয়ে খেলে।তাদের খেলনা গাড়ির সংগ্রহে ১০টি বিভিন্ন মডেল ছিল। শুরুতে কার কয়টি গাড়ি ছিল ? Mintu saw that Sentu had more toy cars than him. He said, Give me 2 cars and we'll have the same number. Sentu instead took 2 cars from Mintu and said, Now I have double the cars you have. "The total number of toy cars was more than 25. Each car cost 30 Taka.They play 3 hours a day with the cars.Their collection included 10 different models ". What were their original counts?	**Answer:**Insufficient Information **Reasoning:** The given information in this problem makes it impossible to confirm solutions.	**Answer:**Mintu: 6, Sentu: 14 **Reasoning:** new condition (number of models) suggests a different solution

A.2 LLM Response

Table 4: Example math problems from the BanglaMATH dataset where all LLMs provide incorrect answers.

Questions (Bangla)	**Answer (Bangla)**	**Math Problem (English)**	**Answer (English)**	**GPT-4**	**LLaMA**	**Gemini 2.5 Flash**	**Deep Seek-V3**	**grok 3**
১০ বছর পূর্বে মিলি ও পুষ্পার বয়সের অনুপাত ছিল ৭:১। বর্তমানে তাদের বয়সের অনুপাত ছিল ৩:১। ১৬ বছর পূর্বে মিলি ও পুষ্পার বয়সের অনুপাত কত?	১৬ বছর পূর্বে পুষ্পার জন্ম হয়নি	10 years ago, the ratio of Mili and Pushpa's ages was 7:1. At present, their age ratio is 3:1. What was their age ratio 16 years ago?	Pushpa was not born 16 years ago	16:1	13:1	13:1	5:1	5:1

পনির ও রবি-নের আয়ের অনুপাত ৪:৩, রবিন ও তপ-নের অনুপাত ৫:৪। পনিরের আয় ১২০০০ টাকা হলে, তিনজনের মোট আয় কত?	২৮২০০ টা-কা	The income ratio of Ponir and Robin is 4:3, and the ratio of Robin and Topon is 5:4. If Ponir’s income is 12,000 Taka, what is the total income of the three?	28,200 Taka	39,600 Tk.	31,200 Tk.	26,100 Tk.	31,200 Tk	22,500 Tk
খেলনা গাড়ি নিয়ে খেলার সময় মিন্টু দে-খলো, সেন্টুর গাড়ির সংখ্যা তার চেয়ে বেশি। সেন্টুকে বললো, তুমি আমাকে দুইটা গাড়ি দাও তা-হলে আমাদের গাড়ির সংখ্যা সমান হবে। সেন্টু উল্টো জোড় করে মিন্টুর থেকে ২টি গাড়ি নিয়ে বলল তার গাড়ির সংখ্যা এখন মিন্টুর দ্বিগুণ। শুরুতে কার কয়টি গাড়ি ছিল?	সেন্টুর ছিল ১০টি, মিন্টুর ছিল ১৪টি খে-লনা গাড়ি	While play-ing with toy cars, Mintu noticed that Sentu had more cars than him. Mintu said, “If you give me two cars, we’ll have the same number.” Sentu instead forcibly took two cars from Mintu and said, “Now I have double the cars you have.” How many toy cars did each have originally?	Sentu had 10, Mintu had 14 toy cars	Mintu had 6, Shentu had 10 cars	Mintu had 14, Shentu had 18 cars	Mintu had 6, Shentu had 10 cars	Mintu had 6, Shentu had 10 cars	Mintu had 6, Shentu had 10 cars .

কোন একটি বিয়ের অনুষ্ঠানে রান্না করতে বাবুর্চি ও তার সহকর্মী মোট ৪০০টি পেঁয়াজ কাটেন। বাবুর্চি প্রতি মিনিটে অন্তত ৩টি পেঁ-য়াজ এবং তার সহকর্মী প্রতি মিনিটে অন্তত ২টি পেঁয়াজ কাটতে পারে। যদি বাবুর্চি তার সহকর্মীর চেয়ে ২৫ মিনিট আগে পেঁয়াজ কাটা বন্ধ করে, তবে কে কতটি পেঁয়াজ কেটে-ছিল আর কার কতক্ষণ সময় লেগেছিল?	বাবুর্চী ৭০ মিনিটে ২১০টি, সহকারী ৯৫ মিনিটে ১৯০টি	At a wedding ceremony, a chef and his assistant together cut 400 onions. The chef can cut at least 3 onions per minute, and the assistant at least 2 onions per minute. If the chef stopped cutting 25 minutes before the assistant did, how many onions did each cut and how long did they work?	Chef: 210 onion (70 min), Assistant: 190 onion (95 min)	Chef: 240 onion As-sis-tant: 160 onion	Chef: 240 onion Assis-tant: 160 onion	Chef: 175 onion Assis-tant: 240 onion	Chef: 240 onion As-sis-tant: 160 onion	Chef: 240 onion As-sis-tant: 160 onion
কারিম সাহে-বের ৩ পুত্র এবং প্রথম ২ জন যমজ। ৩ পুত্রের বর্তমান বয়সের সমষ্টি ২৯ বছর। ৬ বছর পূর্বে তাদের বয়সের সমষ্টি ছিল ১২ বছর। পুত্রদের বর্তমান বয়স কত?	১২, ১২, ৩ বছর	Mr. Karim has three sons, the first two are twins. The sum of their current ages is 29 years. Six years ago, the sum of their ages was 12 years. What are their current ages?	12, 12, and 3 years old	10,10, 9	5,5,10	9,9 ,11	10,10, 9	12,12, 5

Logically Constrained Decoding

Franklin Ma and Alan J. Hu
Department of Computer Science
University of British Columbia
franklin.ma@ubc.ca, ajh@cs.ubc.ca

Abstract

Constrained decoding is a state-of-the-art technique for restricting the output of a Large Language Model (LLM) to obey syntactic rules, e.g., a regular expression or context-free grammar. In this paper, we propose a method for extending constrained decoding beyond syntactic constraints, to enforcing formal, logical constraints that reflect some world model being reasoned about. We demonstrate proof-of-concept implementations for the game of chess, and for propositional resolution proofs: we constrain the LLM's decoding such that the LLM is free to output whatever tokens it wants, as long as it does not make illegal moves (chess) or unsound proof steps (resolution). We believe this technique holds promise for improving LLMs' generation of precise, formal reasoning, as is particularly necessary for mathematics.

1 Introduction

Proof is the quintessential distinguishing feature of mathematical discourse. Like other forms of argumentation, the statements in a proof must be syntactically correct and semantically meaningful, and the overall text should lead to the desired conclusion. What makes proofs distinctive is that each statement must be *sound*, i.e., it must obey formal logical rules with respect to the preceding statements. This is akin to the moves in a game or puzzle: each step must be a legal move. Many applications require such precise, correct, logical reasoning, underscoring the importance of research on NLP for mathematics.

Large Language Models (LLMs) have made extraordinary progress on mathematical reasoning, e.g., both OpenAI and Google DeepMind recently announced gold-medal-level performance on International Math Olympiad problems (Wei, 2025; Luong and Lockhart, 2025). However, even leading-edge frontier models frequently make mistakes. In this paper, we do not concern ourselves with

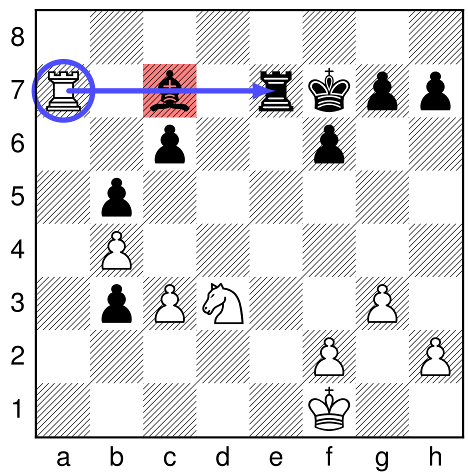


Figure 1: ChatGPT-5 (White) is playing against Stockfish 17.1 (Black). On move 30, White attempts Rxe7+ (shown in blue), i.e., taking Black's rook at e7 and attacking Black's king with White's rook at a7. This is illegal as Black's bishop on c7 is in the way.

wild hallucinations, but rather focus on formally invalid reasoning: statements that, given the precise logic of the world model underlying the reasoning, are illegal or incorrect. For example, we pitted ChatGPT-5 against the well-known chess engine Stockfish (Romstad et al., 2008–present) in a casual game.[1] ChatGPT played a solid opening, but stumbled a bit in the mid-game, reaching the position shown in Fig. 1 with ChatGPT (White) to play its 30th move. At this point, ChatGPT attempted a flagrantly illegal move, presumably

[1]The details of the game are unimportant, but for the curious: We used Stockfish version 17.1, with default settings except a depth-limit of 15. ChatGPT-5 played White, and Stockfish played Black. The moves played were: 1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O Nxe4 6. d4 b5 7. Bb3 d5 8. dxe5 Be6 9. c3 Be7 10. Re1 O-O 11. Nbd2 Bc5 12. Nxe4 dxe4 13. Qxd8 Rfxd8 14. Rxe4 Bxb3 15. axb3 Rd1+ 16. Re1 Rxe1+ 17. Nxe1 Nxe5 18. Bf4 Bd6 19. Bxe5 Bxe5 20. Nd3 Bd6 21. Re1 a5 22. Ra1 f6 23. Kf1 Kf7 24. Ke2 c6 25. g3 Bc7 26. b4 Re8+ 27. Kf1 a4 28. b3 axb3 29. Ra7 Re7, whereupon ChatGPT attempted an illegal move.

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 150–167

indicating that it had lost track of the underlying world model (the board state). Or, for a more mathematical example, we asked ChatGPT-5 to produce a resolution-based refutation proof for a pigeonhole problem with 3 pigeonholes. (See §2.2 and §4.2 for more explanation.) ChatGPT produced 74 logically sound (but sometimes useless or repetitive) resolution steps, before declaring on the 75th step, confidently but completely unfoundedly, that it had reached a contradiction and completed the "proof". We witnessed similar failures with Claude Sonnet 4.

Considerable research has boosted performance of language models on complex reasoning tasks, with techniques like chain-of-thought (Wei et al., 2023) and reinforcement learning with verified rewards (Wang et al., 2025). However, such techniques do not *guarantee* that the LLM will not produce logically illegal outputs. Furthermore, it seems inefficient to try to train language models to do fully precise, logical reasoning, when existing symbolic techniques can handle that well. For example, Pan et al. 2025 show that it is theoretically possible for a custom-programmed transformer to decide propositional satisfiability (albeit inefficiently), but that an empirically trained transformer for 3SAT generalizes and scales poorly; in contrast, existing SAT solvers routinely solve practical problem instances with millions of variables. We believe a neuro-symbolic approach — i.e., augmenting the language model with logical, symbolic reasoning — holds great promise to marry the best attributes of both approaches.[2]

Specifically, we build our work on constrained decoding, a state-of-the-art technique for restricting the output of an LLM to obey syntactic rules, e.g., a regular expression (Beurer-Kellner et al., 2023; Willard and Louf, 2023) or context-free grammar (Willard and Louf, 2023; Ugare et al., 2024).[3] We propose to lift the concept of constrained decoding beyond syntactic constraints, to enforcing formal, logical constraints that reflect some underlying world model. We demonstrate proof-of-concept implementations for the game of chess, and for propositional resolution proofs. We show that our method is easily implemented with several different open-source language models, ensuring generation of guaranteed-correct outputs, while not otherwise perturbing the language models.

2 Background

2.1 Chess

For 60 years, the game of chess has been proclaimed "the Drosophila of AI".[4] "Drosophila" refers to *Drosophila melanogaster*, a species of fruit fly that has been a favorite subject of biological research as a model organism: they are cheap and fast to raise, relatively simple for experiments and analysis, yet they can illuminate the same concepts important in larger and more relevant organisms. AI research has used chess for exactly analogous reasons, and we follow this tradition by using chess for our initial implementation and experiments.

Chess is a two-player, deterministic (i.e., there is no luck involved), perfect-information (i.e., there is no hidden information, like face-down playing cards), turn-based (i.e., the players take turns making moves) game. Each player starts with a standard set of playing pieces, arranged on the playing board in a standard configuration. One player (dubbed "White") plays the light-colored pieces, and moves first; the other player ("Black") plays the dark-colored pieces. There are a variety of types of pieces, with specific formal rules governing how each piece is allowed to move on the board, and to "capture" (remove from the board) pieces from the opposing player. For example, in the board position shown in Fig. 1, White's piece on square a7 is called a "rook" and is allowed in a single turn to move any distance vertically or horizontally, but only through empty squares. It could also move to square c7, resulting in the removal from the board of Black's "bishop" currently on that square. But it is not allowed to move past c7, because Black's bishop occupies that square and blocks further movement. Each player has one distinguished piece, called the "king", and to win the game, a player tries to reach a game state in which one is attacking the opponent's king such that they cannot prevent their king being captured (called "checkmate"). It is also possible for a game

[2] There is even intriguing neuroscience evidence in support of such an approach. In a brain imaging study on highly educated subjects, professional mathematicians used completely different neural pathways to solve math problems, whereas the non-mathematically trained subjects relied solely on their language pathways, with lower accuracy (Amalric and Dehaene, 2016).

[3] In recent work, Mündler et al. 2025 have also extended constrained decoding beyond syntactic constraints, to generate type-safe code. Our work is philosophically very much aligned with theirs.

[4] According to Ensmenger 2012, this metaphor originated with Russian mathematician Alexander Kronrod in 1965 and first appeared in print in (Simon and Chase, 1973).

to end as a draw, in which neither player wins. For example, if a player has no legal moves, but his king is not under attack, then the game ends in a draw.

Chess has a rich literature, spanning centuries. We have presented just enough concepts so that a reader unfamiliar with chess can follow the key points of this paper. We reiterate that our goal is not to produce a superior chess engine, but to use chess as an example of an underlying world model with formal rules, which we can use to constrain an LLM playing chess, such that it never makes an illegal move.

2.2 Propositional Resolution Proofs

In propositional logic, all variables are Boolean (true/false), and there are no function or predicate symbols. Many logical operators are standard, e.g., AND, OR, NOT, etc., but it is standard to assume that formulas are in conjunctive normal form (CNF): a *literal* is either a variable x, or its negation $\overline{x}$; a *clause* is the disjunction of a set of literals, e.g., $(x_1 + x_2 + \overline{x_3})$; and a formula is the conjunction of a set of clauses, e.g., $(x_1 + x_2 + \overline{x_3})(x_3 + \overline{x_4} + x_5)$. (We use $+$ to denote OR, and juxtaposition to denote AND.) Propositional logic is the foundational layer of logical reasoning, making it an ideal testbed for the reasoning capabilities of any AI system. As such, we propose that propositional logic be the drosophila of reasoning.[5]

Formally, a mathematical proof is simply a sequence of statements, leading from a set of assumptions to a desired conclusion, such that (1) each statement is logically implied by the assumptions and preceding statements in the proof, and (2) this implication can be efficiently checked, usually syntactically according to the rules of a given proof system. Specifically, in this paper, we focus on proof by *resolution*: given two clauses $(A_1 + \cdots + A_n + x)$ and $(B_1 + \cdots + B_m + \overline{x})$, where the A_i and B_j are literals, and x is some variable, then the conjunction of the two clauses implies the clause (called the "resolvent") $(A_1 + \cdots + A_n + B_1 + \cdots + B_m)$. In a proof by resolution, each statement in the proof must be the resolvent of clauses from the assumptions or previously generated proof statements. Resolution is known to be a sound (i.e., no false statement can be proven) and complete (i.e., any true statement can be proven) proof system. Without loss of generality, we further restrict ourselves in this paper to proofs by *refutation*, meaning that the desired conclusion is to imply *false*, which proves that the original assumptions are a contradiction.

For convenience in interacting with the text-based LLMs, we adopt the common convention of denoting variables simply by their number, and denoting negation using the minus sign. So the earlier example of clauses $(x_1 + x_2 + \overline{x_3})(x_3 + \overline{x_4} + x_5)$, would be denoted $(1\ +\ 2\ +\ -3)\ (3\ +\ -4\ +\ 5)$.

2.3 Constrained Decoding

At a high level, a typical LLM works as follows:

initialize $buf \leftarrow$ initial prompt
repeat
    $dist \leftarrow \mathrm{Softmax}(\mathrm{Nnet}(buf))$
    sample $next\_token$ from $dist$
    append $next\_token$ to buf
until $next\_token = EOS$

where buf is the context buffer; Nnet is the neural network in the LLM that produces weights for each possible next token; $dist$ is a probability distribution over the possible next tokens, generated via some version of softmax; and EOS is the end-of-sequence token.

The goal is to constrain the LLM to generate only output that obeys some syntax rules. But given the large investment in training the network Nnet, we do not want to modify it. And given that evaluating $\mathrm{Nnet}(buf)$ is slow, we wish to avoid any backtracking or speculative evaluation.

Constrained decoding takes advantage of this basic LLM architecture to modify only the decoding step, to mask out illegal token choices: (Changes are highlighted in green.)

initialize $buf \leftarrow$ initial prompt
initialize parser $P.\textsc{Init}(buf)$
repeat
    $dist \leftarrow \mathrm{Softmax}(\mathrm{Nnet}(buf))$
    $mask \leftarrow P.\textsc{LegalNextTokens}()$
    disallow in $dist$ any token not in $mask$
    sample $next\_token$ from $dist$
    append $next\_token$ to buf
    update $P.\textsc{UpdateState}(next\_token)$
until $next\_token = EOS$

Here, P is some sort of parsing engine for the syntactic constraints being enforced. For example, if

[5] Pan et al. 2025 express a similar sentiment: "Boolean SAT solving captures the essence of deductive logical reasoning because: 1) Boolean logic lies as the foundation of all logical reasoning, and 2) many modern SAT solvers are inherently formal deductive systems that implement the resolution proof system."

we wish to force the LLM output to obey a regular expression, then P could maintain a finite-state automaton that tracks all states from which there is a path to an accepting state. By restricting the next token to always be in $P.\textsc{LegalNextTokens}()$, we guarantee that the generated output cannot violate the regular expression.

Constrained decoding has the desired properties: Nnet is not modified, and not evaluated more than necessary, and the output is *guaranteed* to obey the syntactic restrictions. An additional desirable property is that it is "minimally invasive" (Beurer-Kellner et al., 2024), meaning all legal behavior of the LLM is still allowed, with the same relative probabilities. Clever implementation can make constrained decoding very efficient. For example, "token misalignment" occurs if the LLM and parser tokenize the text stream differently; this can be solved efficiently by pre-computing what is essentially small automaton that performs a limited look-ahead at what tokens the parse is prepared to accept. (Beurer-Kellner et al., 2024; Hamilton and Mimno, 2025) It is also useful to be able to switch between constrained and unconstrained decoding, because restricting the LLM to only constrained output can limit its reasoning ability. (Banerjee et al., 2025) This can be accomplished easily by having the parser recognize specific token sequences to start and stop enforcing constraints.

3 Logically Constrained Decoding

But what if we wish to enforce richer constraints than mere syntax? For example, we might wish to generate a program that obeys specified functional properties, or a mathematical proof that is logically sound. For such an application, syntactical constraints are insufficient, because there is an underlying world model, upon which the correctness or incorrectness of an output depends. For program correctness, this world model might include the values and types of program variables, assumptions on program paths, and the semantics of various operators. For mathematical proof, the underlying model might include constraints (assumed or derived) on the domains and interpretations of all formal symbols, and the status of assumptions and proof goals.

In this paper, we propose the concept of *logically constrained decoding*. We retain the framework of constrained decoding, with its desirable attributes, but we seek to enforce formal, logical constraints that reflect some world model underlying the reasoning.

The basic idea is actually very simple. The key insight is that the parser P in (normal, syntactic) constrained decoding is *already* doing logically constrained decoding, but just for a very limited, underlying world model. For regular expressions, the world model is just a finite automaton; for CFGs, a pushdown automaton. Why not substitute a richer world model?

So, in logically constrained decoding, we generalize the parser into a symbolic constraint engine. Just as in normal constrained decoding, it watches the generated tokens, and updates the state of its internal world model. And just as in normal constrained decoding, it reasons about this world model to mask out illegal next tokens, guaranteeing that the generated output is correct with respect to this underlying world model. The difference is that this world model can be more complex, involving symbolic reasoning.

The challenge, of course, is the LegalNextTokens operation. For purely syntactic constraints, with a finite or pushdown automaton as the underlying world model, the legal next tokens can be calculated via standard automata-theoretic techniques. But for more general constraints, it's not obvious that one can compute the set of legal next tokens, i.e., tokens that are the next token in the prefix of an overall correct output. Are there any nontrivial, non-syntactic world models for which we can implement logically constrained decoding? We answer this question affirmatively with two simple, but non-trivial examples: chess and propositional resolution proofs.

Chess, as a "Drosophila" experiment, turns out to be easy, but illustrates constraining LLM output according to formal, logical rules completely unlike the syntax-focused prior work on constrained decoding. The underlying world model is simply the state of the chess board.[6] As the LLM generates chess moves, the symbolic constraint engine updates the board state. For the LegalNextTokens operations, the constraint engine solves for the set of legal next moves in the current game state, looks at the partially generated move from

[6]Chess afficianados will note that aside from the obvious positions of pieces on the board, state includes some additional information, like *en passant* pawns, castling options, etc. But this is all finite-state and well-documented, e.g., in Forsyth-Edwards notation.

the LLM (if any), and allows any token that can lead to generating one of the complete legal moves.

Propositional resolution proofs are our second, more complex example. Propositional proofs are the essence of mathematical proof, and as mentioned earlier, resolution lies at the core of practical, modern SAT solvers. In this case, the state of the underlying world model is the set of clauses that were either given as assumptions, or have been proven already. A legal "move" is a resolvent of existing clauses. (We can also enforce that the move does not generate a duplicate of an existing clause, or a useless tautology like $(x+\overline{x})$.) When the LLM is trying to generate its next move, the symbolic constraint engine enforces that each additional token maintains that the generated output be a prefix of a legal resolvent. For example, given assumptions (1) $(-1\ +\ 2)$ (-2), the legal resolvents are (2) (generated by resolving (1) and $(-1\ +\ 2)$), and (-1) (generated by resolving $(-1\ +\ 2)$ and (-2)). So, the LLM is constrained[7] to generating a (next, and then either a 2 or a $-$, and then if it had generated $(2$, it must generate a) next, and if it had generated $(-$, it must generate a 1 next, etc.

So, it is possible in theory to do logically constrained decoding for two small, but non-trivial problems. But is it efficient enough to improve the accuracy of real LLMs? That is a question that must be answered empirically.

4 Empirical Results

We now evaluate how well logically constrained decoding works on our two example world models. Specifically, we explore (1) Is it easily implementable in practice on real LLMs? (2) Does it improve the quality of LLM outputs on these problems in practice? and (3) What is the impact on the LLMs' token throughput?

To answer the first question, we implemented logically constrained decoding for these two problems on several different LLM families: Qwen2.5 7B / 14B / 32B (Qwen et al., 2025), Llama 3.1 8B (Grattafiori et al., 2024), Gemma3 7B / 14B / 27B (Team et al., 2025), Phi4-mini (Microsoft et al., 2025), and Ministral-8B (Jiang et al., 2024). We selected these models because they are open-source, and fit on our computing infrastructure. (Our experiments were run on a shared cluster, using Dell EMC C4140 GPU compute nodes, with 8GB RAM per core, and NVIDIA Tesla V100 GPUs with 16GB or 32GB.) All models are instruction fine-tuned. To account for numerical instability, Gemma3 models were run with FP32, while others in FP16. 8-bit quantization was used for all Gemma3 models and Qwen2.5 32B. Code was in Python, using the HuggingFace Transformers Library (Wolf et al., 2020). Overall, we encountered no particular difficulties in implementing logically constrained decoding with these LLMs. Our code is available on GitHub[8].

We evaluate the effect on LLM output quality in §4.1 for chess, and §4.2 for resolution proofs. In §4.3, we report on the effect of logically constrained decoding on LLM performance (token throughput).

4.1 Chess Results

We first investigate how much improvement logically constrained decoding provides to LLMs to avoid illegal moves. Anecdotally, LLMs play openings well, but gradually perform worse as the game progresses. As we saw in Figure 1, even frontier models eventually attempt moves that violate the rules of chess. Thus, as our figure of merit, we look at the number of legal moves an LLM can make before it makes an illegal move. At the syntax level, we ask models to output moves in Standard Algebraic Notation (SAN). While several other notation systems exist (i.e. Long Algebraic Notation or Portable Game Notation), we consider a move valid only if it is written in SAN. More importantly, though, we check whether each move is a valid move according to the rules of chess, for the current board configuration.

To create a consistent opponent, we play LLMs against the Stockfish chess engine. We downloaded the latest 17.1 version and performed experiments across 5 difficulty settings, with a depth of 15 plies. At lower difficulties, Stockfish chooses moves more randomly.

Because of the randomness in both Stockfish and the LLMs, we play 20 games (10 as White, 10 as Black) for each model, with and without logically constrained decoding. When it is Stockfish's turn, it plays (with some randomness, depending on the difficulty) what it believes to be the best move. When it is the LLM's turn, we allow it to generate a maximum of 10 tokens, which is longer

[7] As noted earlier, the symbolic constraint engine can be designed to allow the LLM some unconstrained thinking tokens, and only constrain specific parts of the output.

[8] `https://github.com/terwo/logically-constrained-decoding`

Reason	Games by Type Unconstrained	Constrained	Total
Illegal Move	897	0	897
Checkmated	3	898	901
Draw	0	2	2
Total	**900**	**900**	**1800**

Table 1: Chess Game Outcomes Across All Experiments. For each of the Unconstrained condition (the original LLM) and the Constrained condition (the LLM with logically constrained decoding), we played 9 LLMs against 5 different levels of Stockfish for 20 games each, for a total of 900 games. Without the benefit of logically constrained decoding, the LLMs attempted illegal moves in 897 out of 900 games; in the other 3, the LLM lost before it could make an illegal move. With logically constrained decoding, the LLMs never made illegal moves in any game. Draws were due to the five-fold repetition rule.

than any possible SAN move in any position. For the Constrained condition, we allow the start of a legal move to have leading whitespace, and the end to have trailing whitespace, but do not allow whitespace between partial continuations of them. For example, "␣Nf4" is allowed but "N␣f4" is not. Each completed set of moves (one from both White and Black) is then formatted into the prompt for the LLM's next move. Full prompts are detailed in Appendix A.

Detailed experimental results are shown in Figure 2. On average, the unconstrained models generated roughly 5 moves before making an illegal one, whereas the logically constrained models played legal moves for much longer, until the natural end of the game. Table 1 summarizes the reasons that each game ended, over all experiments. Clearly, logically constrained decoding makes a dramatic difference in LLM correctness for chess.

4.2 Resolution Proof Results

These experiments are to assess the improvement that logically constrained decoding provides to LLMs, to prevent the generation of incorrect proof steps. Unlike in chess, there is no value in generating a longer proof — a proof is either correct or not. Therefore, we count the number of correctly generated proofs, over repeated trials, as our figure of merit.

Similar to the variety of chess notations, there are various syntaxes to represent a clause in propositional logic. We chose the notation common in Boolean SAT solving and in electrical engineering, where the plus sign $+$ symbolizes a logical OR.

As the challenge problem for the proofs, we chose the well-known Pigeonhole Problem: proving that it's impossible to place $n + 1$ pigeons in n pigeonholes if no two pigeons can share a pigeonhole. These are hard proofs: the propositional encoding for this problem has $\Theta(n^2)$ variables and $\Theta(n^3)$ clauses, and the resolution proof has an exponential lower-bound in size. We generate problem instances that encode the pigeonhole problem for 1, 2, and 3 holes. Our specific SAT encoding is shown in Appendix B.

For 1 and 2 holes, we run each model 50 times for each constraint condition (with and without logically constrained decoding). For 3 holes, due to limited time and computing resources, we could not complete the full number of trials — more details below. Similar to our implementation for playing chess, LLMs were permitted to output tokens that correspond to brackets, literals, or plus signs with leading or trailing whitespace. The LLMs are also allowed to output reasoning steps in separate lines that start with a double backslash `//`, and output the clauses for the proof on new lines. The prompts used are in Appendix A.

Discussing the results in increasing order of pigeonhole size:

1 Pigeonhole, 2 Pigeons: Many unconstrained models were able to successfully generate resolution proofs for the pigeonhole problem with only 1 hole and 2 pigeons. With the benefit of logically constrained decoding, the success rate goes to 100%. A correct resolution proof for this encoding is very short (only 2 resolvents before the empty clause), so we limited the output generation to 100 tokens. Figure 3 shows the accuracy for both constraint conditions across all models for 50 iterations.

2 Pigeonholes, 3 Pigeons: With 2 pigeonholes, the proof becomes much harder. Across 50 iterations, no Unconstrained models were able to successfully complete the resolution proof. In contrast, with logically constrained decoding, every model successfully generated a correct proof 100% of the time. Figure 4 depicts these results graphically. We limited the output generation to 1000 tokens. We also attempted this proof informally on some commercial frontier models. (We do not have the resources to do extensive experiments on these models.) ChatGPT-5 completed this proof

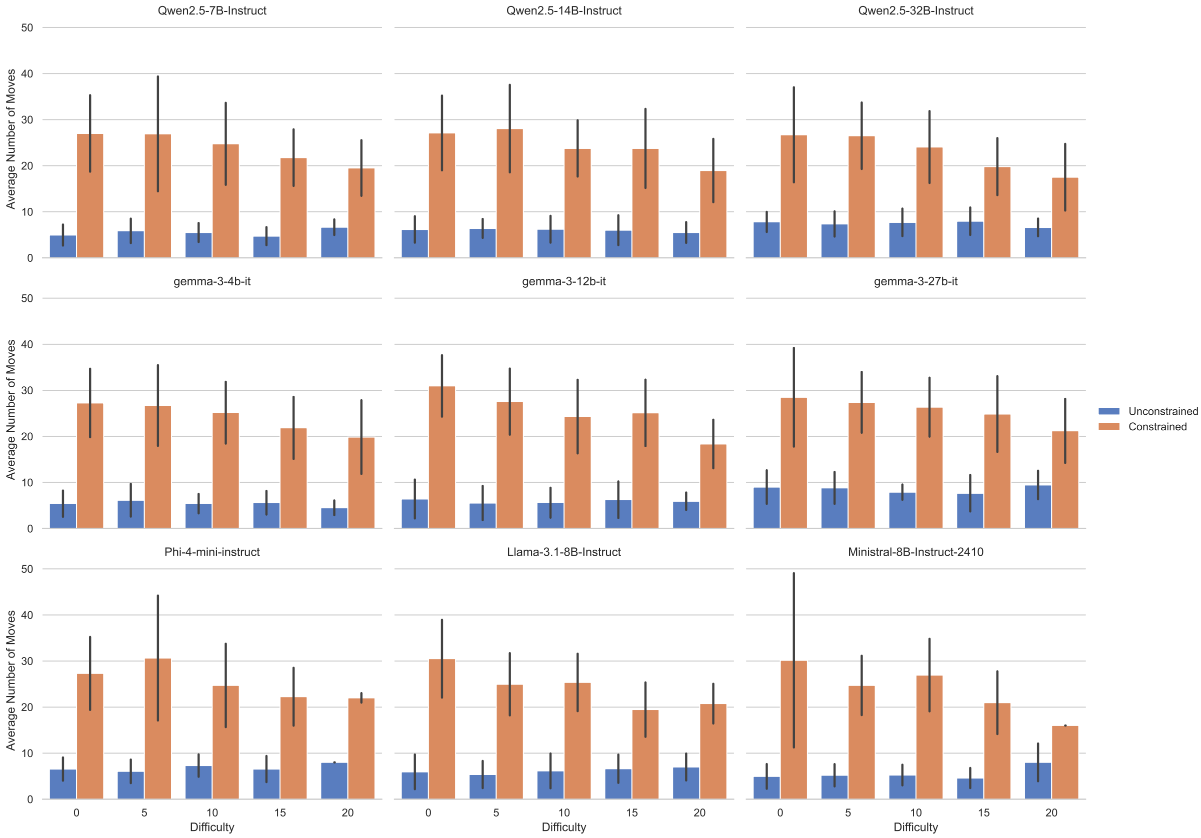


Figure 2: Detailed Experimental Outcomes for Chess Games. There are nine subgraphs here, one for each LLM, labeled above the subgraph. On each subgraph, the x-axis is the Stockfish difficulty level that was the opponent of the LLM. The y-axis is the average number of moves played, over 20 games. (The whiskers show 1 standard deviation.) For each experimental condition, the blue bar on the left is for the original, unconstrained LLM, and the orange bar on the right is with logically constrained decoding. In the unconstrained condition, the LLMs make illegal moves very soon after starting to play. With logically constrained decoding, the LLMs never make illegal moves, so last long enough to eventually lose, losing faster to the stronger Stockfish difficulty settings.

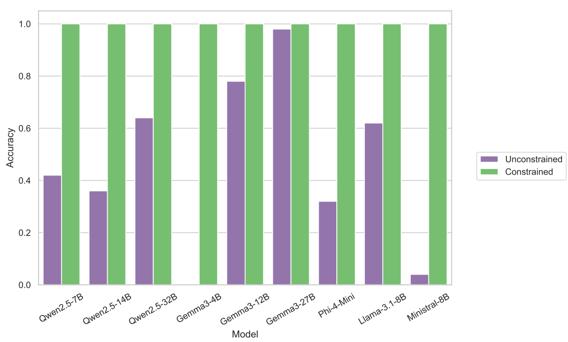


Figure 3: Results for Pigeonhole Proofs of Size 1. The y-axis is the fraction of proof attempts that were correct (out of 50 attempts). The x-axis has a pair of bars for each LLM. In each pair, the blue bar on the left is the success rate for the original, unconstrained LLM; the orange bar on the right is the success rate with logically constrained decoding. A missing bar indicates 0% correct proofs. These small LLMs are able to generate correct resolution proofs in many cases, but this improves to 100% with logically constrained decoding.

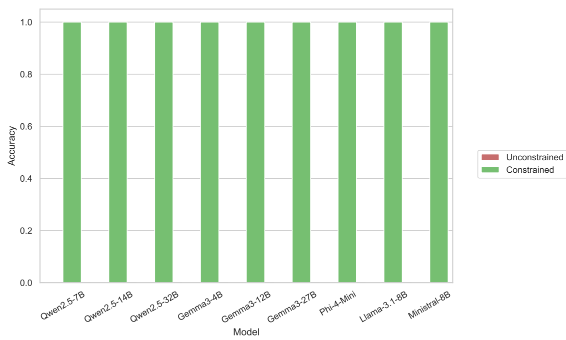


Figure 4: Results for Pigeonhole Proofs of Size 2. This graph has the same interpretation as Figure 3. However, all the blue bars are missing, because no unconstrained LLM was able to complete this proof correctly. The constrained LLM successfully completes the resolution proof on all attempts.

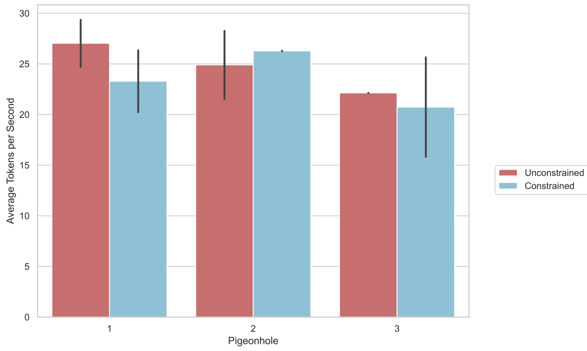


Figure 5: The average tokens per second generated by Qwen2.5-7B across both Unconstrained and Constrained conditions. The logically constrained decoding for pigeonhole instances of size 3 is optimized by enforcing the LLM to only choose possible resolvents of minimal length.

successfully, but Claude Sonnet 4 did not.

3 Pigeonholes, 4 Pigeons: Due to limited time and computing resources, we reduced the number of trials from 50 to 20 per experimental condition. None of the unconstrained models were able to successfully complete this resolution proof. We limited the output generation to 3000 tokens.

For the logically constrained LLMs, we were not able to complete the experiments. This is a long and hard proof, and as the number of clauses in the proof grew, the number of resolvents became unmanageably large. However, if we relax the goal of minimal invasiveness, we can exploit the logical structure of clauses to improve efficiency: specifically, if a partially generated clause is already a legal resolvent, it is pointless to allow the clause to grow any longer, as that only makes the clause weaker. Accordingly, we can modify the constraint engine to force the LLM not to generate a needlessly long resolvent. With this optimization, even the small Qwen2.5 7B model manages to complete the proof correctly. In contrast, as described in the introduction, ChatGPT-5 tries to make an unsound deduction in its proof attempt.

4.3 Effect on LLM Performance

As a measure of the effect of logically constrained decoding on LLM throughput, we measured the average number of tokens per second across all resolution proofs generated by Qwen2.5-7B in Figure 5. The optimization mentioned above is applied to the pigeonhole instances of size 3. The latency overhead is generally minimal in our experiments.

On the downside, our symbolic constraint engine for the resolution proofs doesn't scale well as the proof length grows, because it is trying to generate all possible resolvents. For example, the shortest proof generated by Qwen2.5-7B on the 3-pigeonhole proofs was 126 clauses long, and on this shorter proof, the throughput was 24.7 tokens per second. In contrast, the longest proof took 627 clauses, which slowed the throughput down to 5.5 tokens per second. This motivates our future work, to explore more efficient proof systems that avoid this blow-up in the number of resolvents.

5 Conclusion and Future Work

We have introduced *logically constrained decoding*, which lifts the concept of constrained decoding beyond syntactic constraints to enforcing logically correct output with respect to an underlying, formal world model. We have demonstrated proof-of-concept implementations for chess and for propositional resolution proofs, on nine different LLMs. Our technique guarantees that the small LLMs do not generate illegal outputs for the problems being solved, and enables them to generate correct outputs on problems that even state-of-the-art, proprietary frontier models solve incorrectly.

The main line for future work is to expand the applicability of our technique to additional domains, such as code generation or richer proof systems (e.g., Lean). The principle challenge is how to turn the logical constraints into something that can be enforced efficiently at the token level. For code generation, we are excited by the work of Mündler et al. (Mündler et al., 2025) on constrained generation of type-safe programs. For mathematical proof, we are currently developing a more efficient technique for a more powerful proof system than resolution.

Limitations

In empirical research on LLMs, there is always the risk of unexpected behavior due to minor variations in prompts. For example, there are many notations for Boolean OR in common use that might have appeared in training data, so an LLM might behave differently if we had prompted it to use $\vee$ or even `\lor`, instead of $+$. We have not explored varying the prompts, but we do not expect that our results would change materially. Our prompts are disclosed in Appendix A.

We can modify and perform experiments only on open-source language models, so it is unclear to what degree our results can be applied to proprietary, frontier models. Similarly, even with open-source models, we were limited by our available computing resources to using smaller models. We believe these are sufficient to demonstrate the promise of our approach, but more extensive experiments would be valuable.

In our prompts, we suggest limits to how the LLM can "think aloud" in its answers. This is purely to simplify our implementation, so that our symbolic reasoning engine can easily ignore the unstructured portions of the LLM output. Banerjee et al. 2025 show that strictly constraining LLM outputs can reduce the LLM's reasoning ability, but this can be restored by allowing the LLM to generate unconstrained output with only clearly delimited parts subject to the constraints. Our chess experiments did not allow arbitrary unconstrained "thinking" outputs, so the LLMs likely did not play as well as they might have. Nevertheless, we were evaluating LLMs only on whether moves were legal or not. Our resolution proof experiments did allow the LLMs to generate unconstrained outputs within comments, along the lines suggested by Banerjee et al.

Our current implementation for resolution does not scale to larger proofs. Even so, we are able to generate correct proofs with smaller LLMs for problems that befuddle large, frontier LLMs. As noted above, we are working on a much more efficient proof system, which should scale better.

Acknowledgements

This research was funded through an Undergraduate Summer Research Award and a Discovery Grant, both from the Natural Sciences and Engineering Research Council of Canada (NSERC). This research was also supported through the computational resources and services provided by Advanced Research Computing at the University of British Columbia.

References

Marie Amalric and Stanislas Dehaene. 2016. Origins of the brain networks for advanced mathematics in expert mathematicians. *Proceedings of the National Academy of Sciences*, 113(18):4909–4917.

Debangshu Banerjee, Tarun Suresh, Shubham Ugare, Sasa Misailovic, and Gagandeep Singh. 2025. CRANE: Reasoning with constrained LLM generation. In *42nd International Conference on Machine Learning*.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2023. Prompting is programming: A query language for large language models. *Proc. ACM Program. Lang.*, 7(PLDI).

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2024. Guiding LLMs the right way: Fast, non-invasive constrained generation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Nathan Ensmenger. 2012. Is chess the drosophila of artificial intelligence? a social history of an algorithm. *Social Studies of Science*, 42(1):5–30.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, Bruno Régaldo-Saint Blancard, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. 2024. xval: A continuous numerical tokenization for scientific language models. *Preprint*, arXiv:2310.02989.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas

Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng

Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Sil Hamilton and David Mimno. 2025. Lost in space: Finding the right tokens for structured output. *Preprint*, arXiv:2502.14969.

Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. 2018. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437.

Alexey Ignatiev, Zi Li Tan, and Christos Karamanos. 2024. Towards universally accessible SAT technology. In *SAT*, pages 4:1–4:11.

Albert Jiang, Alexandre Abou Chahine, Alexandre Sablayrolles, Alexis Tacnet, Alodie Boissonnet, Alok Kothari, Amélie Héliou, Andy Lo, Anna Peronnin, Antoine Meunier, Antoine Roux, Antonin Faure, Aritra Paul, Arthur Darcet, Arthur Mensch, Audrey Herblin-Stoop, Augustin Garreau, Austin Birky, Avinash Sooriyarachchi, Baptiste Rozière, Barry Conklin, Bastien Bouillon, Blanche Savary de Beauregard, Carole Rambaud, Caroline Feldman, Charles de Freminville, Charline Mauro, Chih-Kuan Yeh, Chris Bamford, Clement Auguy, Corentin Heintz, Cyriaque Dubois, Devendra Singh Chaplot, Diego Las Casas, Diogo Costa, Eléonore Arcelin, Emma Bou Hanna, Etienne Metzger, Fanny Olivier Autran, Francois Lesage, Garance Gourdel, Gaspard Blanchet, Gaspard Donada Vidal, Gianna Maria Lengyel, Guillaume Bour, Guillaume Lample, Gustave Denis, Harizo Rajaona, Himanshu Jaju, Ian Mack, Ian Mathew, Jean-Malo Delignon, Jeremy Facchetti, Jessica Chudnovsky, Joachim Studnia, Justus Murke, Kartik Khandelwal, Kenneth Chiu, Kevin Riera, Leonard Blier, Leonard Suslian, Leonardo Deschaseaux, Louis Martin, Louis Ternon, Lucile Saulnier, Lélio Renard Lavaud, Sophia Yang, Margaret Jennings, Marie Pellat, Marie Torelli, Marjorie Janiewicz, Mathis Felardos, Maxime Darrin, Michael Hoff, Mickaël Seznec, Misha Jessel Kenyon, Nayef Derwiche, Nicolas Carmont Zaragoza, Nicolas Faurie, Nicolas Moreau, Nicolas Schuhl, Nikhil Raghuraman, Niklas Muhs, Olivier de Garrigues, Patricia Rozé, Patricia Wang, Patrick von Platen, Paul Jacob, Pauline Buche, Pavankumar Reddy Muddireddy, Perry Savas, Pierre Stock, Pravesh Agrawal, Renaud de Peretti, Romain Sauvestre, Romain Sinthe, Roman Soletskyi, Sagar Vaze, Sandeep Subramanian, Saurabh Garg, Soham Ghosh, Sylvain Regnier, Szymon Antoniak, Teven Le Scao, Theophile Gervet, Thibault Schueller, Thibaut Lavril, Thomas Wang, Timothée Lacroix, Valeriia Nemychnikova, Wendy Shang, William El Sayed, and William Marshall. 2024. Ministral 8b.

Thang Luong and Edward Lockhart. 2025. Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the International Mathematical Olympiad. `https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/`. [Online; accessed 2025-Aug-25].

Microsoft, :, Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dong Chen, Dongdong Chen, Junkun Chen, Weizhu Chen, Yen-Chun Chen, Yi ling Chen, Qi Dai, Xiyang Dai, Ruchao Fan, Mei Gao, Min Gao, Amit Garg, Abhishek Goswami, Junheng Hao, Amr Hendy, Yuxuan Hu, Xin Jin, Mahmoud Khademi, Dongwoo Kim, Young Jin Kim, Gina Lee, Jinyu Li, Yunsheng Li, Chen Liang, Xihui Lin, Zeqi Lin, Mengchen Liu, Yang Liu, Gilsinia Lopez, Chong Luo, Piyush Madan, Vadim Mazalov, Arindam Mitra, Ali Mousavi, Anh Nguyen, Jing Pan, Daniel Perez-Becker, Jacob Platin, Thomas Portet, Kai Qiu, Bo Ren, Liliang Ren, Sambuddha Roy, Ning Shang, Yelong Shen, Saksham Singhal, Subhojit Som, Xia Song, Tetyana Sych, Praneetha Vaddamanu, Shuohang Wang, Yiming Wang, Zhenghao Wang, Haibin Wu, Haoran Xu, Weijian Xu, Yifan Yang, Ziyi Yang, Donghan Yu, Ishmam Zabir, Jianwen Zhang, Li Lyna Zhang, Yunan Zhang, and Xiren Zhou. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *Preprint*, arXiv:2503.01743.

Niels Mündler, Jingxuan He, Hao Wang, Koushik Sen, Dawn Song, and Martin Vechev. 2025. Type-constrained code generation with language models. *Proceedings of the ACM on Programming Languages*, 9(PLDI):601–626.

Leyan Pan, Vijay Ganesh, Jacob Abernethy, Chris Esposo, and Wenke Lee. 2025. Can transformers reason logically? a study in SAT solving. In *42nd International Conference on Machine Learning (ICML)*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Tord Romstad, Marco Costalba, Joona Kiiski, and Stockfish Community. 2008–present. Stockfish: Strong open-source chess engine. `https://stockfishchess.org/`. [Online; accessed 2025-Aug-25].

Herbert A. Simon and William G. Chase. 1973. Skill in chess. *American Scientist*, 61:394–403.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Shubham Ugare, Tarun Suresh, Hangoo Kang, Sasa Misailovic, and Gagandeep Singh. 2024. Syncode: Llm generation with grammar augmentation. *Preprint*, arXiv:2403.01632.

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. 2025. Reinforcement learning for reasoning in large language models with one training example. *Preprint*, arXiv:2504.20571.

Alexander Wei. 2025. I'm excited to share that our latest @OpenAI experimental reasoning LLM has achieved a longstanding grand challenge in AI: gold medal-level performance on the world's most prestigious math competition—the International Math Olympiad (IMO). `https://x.com/alexwei_/status/1946477742855532918`. [Online; accessed 2025-Aug-25].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Brandon T. Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *Preprint*, arXiv:2307.09702.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

A Full Prompts

The prompts used for experiments are listed below. There is no whitespace after the colon in all prompts.

Chess Prompts

White's Perspective

You are a chess grandmaster.
You are playing chess as White in the starting position. You are playing against a strong opponent.
Output your next, competitive move in Standard Algebraic Notation (SAN). Do not include move numbers.

White:

Black's Perspective

You are a chess grandmaster.
You are playing chess as Black after White has made their first move. You are playing against a strong opponent.
Output your next, competitive move in Standard Algebraic Notation (SAN). Do not include move numbers.

White:
`{stockfish_chosen_move}`
Black:

Figure 6: Prompts for Playing Chess Against Stockfish. If playing as Black, the first prompt will include the first move chosen by Stockfish.

Unconstrained Resolution Prompt

Generate an unsatisfiability proof for the given clause database using only resolution steps.
Rules for Clauses:
1. Each clause must start with '(' and end with ')'.
- Integers must be separated by '+' with optional spaces around it.
- Negated literals have a leading '-'
- Example: (1 + -3 + 4) 2.
Each derived clause must be valid with respect to the original database and all previously generated clauses.
- A clause C is valid if it is the resolvent of two existing clauses in the current set.
- The two parent clauses must share exactly one pair of complementary literals.
- The resolvent is formed by taking all literals from both parents except the complementary pair, with duplicate literals removed.
3. Do not repeat any clauses already in the database or previously generated.

Output Format:
- Each line is either:
a) A comment line starting with '//' followed by reasoning, OR
b) A clause line in parentheses only, with no extra text.
- No introductions, no summaries, no prose outside of comment lines.
- First non-comment line must be a clause.
- The proof must end exactly with the empty clause ().
Example:
Clause database: (1 + 2) (1 + -2) (-1 + 2) (-1 + -2) (1 + 2 + 3)
Proof:
// Resolving (1 + 2) and (-1 + 2) on literals 1 and -1 gives (2)
(2)
// Resolving (1 + -2) and (-1 + -2) on literals 1 and -1 gives (-2)
(-2)
// Resolving (2) and (-2) gives the empty clause ()
()

Now, generate an unsatisfiability proof for the following:
Clause database: `{clause_database}`
Proof:

Figure 7: The Prompt to Generate Resolution Proofs with Unconstrained Decoding

Constrained Resolution Prompt

Generate an unsatisfiability proof for the given clause database using only resolution steps.
Rules for Clauses:
1. Each clause must start with '(' and end with ')'.
- Integers must be separated by '+' with optional spaces around it.
- Negated literals have a leading '-'
- Example: (1 + -3 + 4) 2.
Each derived clause must be valid with respect to the original database and all previously generated clauses.
- A clause C is valid if it is the resolvent of two existing clauses in the current set.
- The two parent clauses must share exactly one pair of complementary literals.
- The resolvent is formed by taking all literals from both parents except the complementary pair, with duplicate literals removed.
3. Do not repeat any clauses already in the database or previously generated.

Output Format:
- Each line is either:
a) A comment line starting with '//' followed by reasoning, OR
b) A clause line in parentheses only, with no extra text.
- No introductions, no summaries, no prose outside of comment lines.
- First non-comment line must be a clause.
- The proof must end exactly with the empty clause ().
Example:
Clause database: (1 + 2) (1 + -2) (-1 + 2) (-1 + -2) (1 + 2 + 3)
Proof: (2) (-2) ()

Now, generate an unsatisfiability proof for the following:
Clause database: `{clause_database}`
Proof:

Figure 8: The Prompt to Generate Resolution Proofs with Logically Constrained Decoding

B Pigeonhole Encoding

We encode our pigeonhole principle instances with n holes and $k = n + 1$ pigeons: for each pigeon $i \in \{1, ..., k\}$ and hole $j \in \{1, ..., n\}$, we introduce a new variable variable x_{ij}, which is True if pigeon i is in hole j.

The CNF formula has two types of clauses:

1. Pigeon clauses
   For each pigeon i, it must be in some hole
   $$(x_{i1} + x_{i2} + ... + x_{in}) \text{ for each } i \in \{1, ..., k\}$$
2. Hole clauses
   For each hole j, for every pair of distinct pigeons $i \neq i'$, both must not be in the same hole
   $$(-x_{ij} + -x_{i'j}) \text{ for each } j \in \{1, ..., n\}, 1 \leq i < i' \leq k$$

Since there are $n + 1$ pigeons yet only n holes, the formula is unsatisfiable, which can be proven with resolution.

We store these formulas in the DIMACS format. We load these formulas with the PySAT package (Ignatiev et al., 2018, 2024). An example of our encoding for a pigeonhole instance of size 2 is as follows:

```
p cnf 6 9
1 2  0
3 4  0
5 6  0
-1 -3 0
-1 -5 0
-3 -5 0
-2 -4 0
-2 -6 0
-4 -6 0
```

C Tokenization

We only allow one consistent method of representing answers to each given problem (e.g., SAN for chess, using plus signs OR in resolution proofs). In our implementation, we track the generation state, and condition the valid next tokens on the current state. For example, in the resolution proof example, the logically constrained model can only generate literals after outputting a left paranthesis or plus sign.

Therefore, we ensure that no tokenizer would split legal strings across states. After investigating how each model family would tokenize text that represent legal moves / valid clauses under our specified syntax, we did not find cases where strings that are composed of multiple states would be represented as one token (e.g., "+␣3" could be tokenized separately as "+" and "␣3" but not fully as one token). Many tokenizers also represent larger numbers by splitting them up digit-by-digit (Golkar et al., 2024), and we account for this in the state transitions for SAT solving by allowing the generation state to enter a "partial literal" state.

D Sampling Parameters

We use all defaults provided by the HuggingFace Transformers library. We explicitly set temperature = 1.0, but otherwise defer to the model's default configuration (e.g., top-k, top-k, etc.).

Pigeonhole Size (n)	**Max Token Limit**
1	100
2	1000
3	3000

Table 2: Maximum Token Limits Allocated Depending on the Pigeonhole Size

Modeling Tactics as Operators: Effect-Grounded Representations for Lean Theorem Proving

Elisaveta Samoylov **Soroush Vosoughi**
Department of Computer Science, Dartmouth College
{elisaveta.v.samoylov.26@dartmouth.edu, Soroush.Vosoughi}@dartmouth.edu

Abstract

Interactive theorem provers (ITPs) such as Lean expose proof construction as a sequence of tactics applied to proof states. Existing machine learning approaches typically treat tactics either as surface tokens or as labels conditioned on the current state, eliding their operator-like semantics. This paper introduces a representation learning framework in which tactics are characterized by the changes they induce on proof states. Using a stepwise Lean proof corpus, we construct *delta contexts*—token-level additions/removals and typed structural edits—and train simple distributional models (Δ-SGNS and CBOW-Δ) to learn tactic embeddings grounded in these state transitions. Experiments on tactic retrieval and operator-style analogy tests show that Δ-supervision yields more interpretable and generalizable embeddings than surface-only baselines. Our findings suggest that capturing the semantics of tactics requires modeling their state-transformational effects, rather than relying on distributional co-occurrence alone.

1 Introduction & Related Work

Interactive theorem provers (ITPs) such as Lean expose proof construction as a sequence of *tactics* that transform a proof *state*, the local context and current goal(s), until all goals are discharged. Learning to select or synthesize tactics has therefore become a central problem in neural theorem proving, with notable progress in HOL4, HOL Light, and Coq through systems such as TacticToe, HOList, and CoqGym (Gauthier et al., 2021; Bansal et al., 2019; Yang and Deng, 2019). In parallel, LeanDojo has catalyzed research around Lean by providing an open, reproducible environment, datasets, and retrieval-augmented provers (Yang et al., 2023).

Yet, most existing approaches model tactics either as labels to be predicted from the *current* state or as tokens in tactic sequences, rather than as *operators that cause state change*. This elides the essential semantics of tactics, *"what they do to the proof state?,"* and can yield brittle representations tied to surface forms, local names, or co-occurrence statistics.

Problem formulation. We propose a representation-learning view in which a tactic, t, is characterized by its *effect on the state*. Given triples $(s_{\text{before}}, t, s_{\text{after}})$ from stepwise Lean proofs, we learn embeddings that make $E(s_{\text{after}})$ predictable from $E(s_{\text{before}})$ and an operator-like embedding e_t. Concretely, we use simple CBOW/skip-gram objectives in which the *delta* between `state_before` and `state_after`, i.e., token-level additions/removals and typed edits such as goal/hypothesis count changes or relation normalizations, serves as the context of t. This Δ-supervision biases the model to encode a tactic by the *modification it induces* (e.g., the tactic `have` introduces new hypotheses, the tactic `nlinarith` often discharges inequality goals, etc) rather than by memorizing surface patterns.

Proof of concept. This paper is a proof-of-concept showing that surface-level distributional patterns that often suffice in natural-language modeling are inadequate for theorem proving. Capturing tactic semantics demands tailored, effect-grounded representations that model the proof-state transition induced by each tactic.

We instantiate this idea on the **Lean Workbook** corpus (Ying et al., 2024; InternLM, 2024), which contains tens of thousands of contest-level math problems formalized in Lean 4 with natural-language statements, formal statements, and stepwise proof states. Our models are deliberately minimal: (i) Δ-*SGNS* (center = tactic; contexts = positive Δ-tokens and typed edits) and (ii) *CBOW-*Δ (contexts $\rightarrow$ tactic), trained in the spirit of word2vec (Mikolov et al., 2013a,b). We evaluate with (a) $\Delta \rightarrow$ *tactic retrieval* (does a bag of edits

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 168–175
November 9, 2025 ©2025 Association for Computational Linguistics

identify the tactic?) and (b) an *operator analogy* test adapted from translation embeddings (Bordes et al., 2013): $\cos\big(E(s_{\text{before}}) + e_t,\ E(s_{\text{after}})\big)$.

Despite their simplicity, these models learn meaningful, interpretable tactic embeddings that reflect operator-like behavior.

Motivation and expected benefits. Two gaps motivate Δ-supervision: (i) Generalization. State-only classifiers and sequence models must implicitly infer a tactic's effect as their representations often entangle names and pretty-print idiosyncrasies. By centering supervision on structural edits (e.g., added hypotheses, reduced goal lists, relation flips), Δ-aware representations should transfer across theorems and libraries, complementing retrieval-based systems such as LeanDojo (Yang et al., 2023). (ii) Interpretability and modularity. Operator-style embeddings provide concrete semantics. In other words, they predict how a state should move in embedding space under t, enabling tactic-family clustering, edit-level probing, and straightforward integration into search controllers that prefer tactics moving states *toward* a solved region.

Novelty relative to prior work. Prior systems typically (a) learn tactic selection from the current state, (b) predict the next tactic from tactic sequences, or (c) fine-tune LLMs end-to-end to emit Lean scripts (Gauthier et al., 2021; Bansal et al., 2019; Yang and Deng, 2019; Yang et al., 2023). In contrast, we elevate the *state transition* to the supervisory signal: the label for a tactic is its *observed effect*. This yields compact, interpretable embeddings that are immediately useful for tactic ranking and are natural starting points for stronger operator models.

By reframing tactics as operators on proof states and demonstrating that even bag-of-edits supervision suffices to obtain useful embeddings, this work offers a lightweight, interpretable bridge between ITP state representations and neural models, complementary to environment-heavy frameworks and retrieval-augmented provers (Yang et al., 2023).

2 Methodology

2.1 A brief primer on Lean and proof states

Lean is an ITP based on dependent type theory. A proof proceeds by repeatedly applying *tactics* that transform a *proof state*, which consists of (i) a *local context* of named hypotheses and (ii) a (multi)set of *goals* to be proved. In Lean's pretty printer, the turnstile symbol $\vdash$ separates the context from the current goal.

Examples of Lean code can be found in Appendix B.

2.2 Dataset and supervision triples

We use a stepwise Lean proof corpus (Lean Workbook), which provides, for each proof step, a triple $(s_{\text{before}},\ t,\ s_{\text{after}})$, where t is the head tactic applied to transform the pretty-printed proof state s_{before} into s_{after}. Each record also includes a `proof_id` (the theorem/proof identifier). The lines are sorted such that their order corresponds to their position within the proof. We operate on a 25.2k-step slice for the experiments reported in this paper.

Our central idea is to treat the tactic t not merely as a label but as an *operator* whose semantics are expressed by the change from s_{before} to s_{after}. To that end, we introduce Δ*-contexts* which track the various changes in the state of the proof.

In the following sections, we outline the preprocessing steps, the construction of the Δ-contexts, and the construction of the two distributional models (Δ-SGNS and CBOW-Δ).

2.3 Preprocessing and canonicalization

Lean states are textual pretty-prints. To make edits comparable across problems, we process the Lean code in following steps in order to produce the delta contexts :

- **Tokenization.** We split on Lean tokens and mathematical operators (including unicode) such as $\vdash$, $\leq$, $\geq$, ^, *, /, +, -, :, =, (,).
- **Alpha-renaming.** In order to prevent name leakage, We anonymize local variable and hypothesis names (e.g., `a, b, c, h, ha, hb, ...`) to placeholders ($\_x1, \_x2, \_h1, \_h2, \ldots$).
- **Formatting normalization.** We normalize whitespace and numeric formats.
- **Goal/context split.** We detect the first $\vdash$ and treat the tokens to its left as context and to its right as goal; if no goal remains, we append a marker `NO_GOALS`.
- **Multi-goal states.** If a state contains multiple goals, we use the union of their tokens. Set-level pooling keeps the representation permutation-invariant.

2.4 Delta construction: tokens and typed edits

The first Δ-construction is token-level and considers the change in token-count in the context before and after the tactic. For each pair $(s_{\text{before}}, s_{\text{after}})$, we compute *token*-level deltas for each token w:

$$\Delta(w) = \text{count}_{s_{\text{after}}}(w) - \text{count}_{s_{\text{before}}}(w). \quad (1)$$

We keep only the *positive* token deltas and emit a context symbol TOK_w for each net added token w, repeated $|\Delta(w)|$ times. To inject weak structure, we add *typed* edit indicators extracted by simple regex rules:

- **Hypothesis count change:** Δ_ADD_HYP / Δ_REM_HYP based on the changes in counts of `name : type` hypothesis pairs in the context.
- **Goal change:** Δ_GOAL_SOLVED if $\vdash$ disappears; otherwise Δ_GOALS_+k/-k for detected changes.
- **Relation flips:** The Δ_REL_GE_TO_LE, Δ_REL_LE_TO_GE when the goal changes direction.
- **Operator counts:** Δ_ADD/REM_SYM_σ for $\sigma \in \{$^, *, /, +, -, =, $\leq$, $\geq\}$.

The final *delta context* for a step is the multiset

$$\mathcal{C} \;=\; \{\texttt{TOK\_}w : \Delta(w) > 0\} \;\cup\; \{\text{typed edit indicators}\}$$

Appendix A walks through a worked example illustrating the procedures in Sections 2.3 and 2.4.

2.5 Models

We learn three lightweight distributional models using only the delta contexts $\mathcal{C}$ and tactic heads t. Let $e_x \in \mathbb{R}^d$ denote the embedding of token/tactic x.

Δ-SGNS (tactic → delta). We train a skip-gram with negative sampling where the *center* word is the tactic head TACTIC t, and each $c \in \mathcal{C}$ is a *context* word. The per-pair loss is

$$\ell_{\text{SGNS}}(t, c) \;=\; -\log \sigma(e_t^\top e_c) - \sum_{c^- \sim P_n} \log \sigma(-e_t^\top e_{c^-}) \quad (2)$$

with negatives c^- drawn from a unigram distribution raised to the $3/4$ power. This objective encourages the tactic vector e_t to co-occur with the *added* tokens and typed edits it tends to induce.

CBOW-Δ (delta → tactic). We train a CBOW model that averages the context embeddings and predicts the tactic:

$$\bar{e}_{\mathcal{C}} \;=\; \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} e_c \quad (3)$$

$$\ell_{\text{CBOW}}(\mathcal{C}, t) = -\log \sigma(\bar{e}_{\mathcal{C}}^\top e_t) - \sum_{t^- \sim P_n} \log \sigma(-\bar{e}_{\mathcal{C}}^\top e_{t^-}). \quad (4)$$

with negatives t^- sampled from the tactic frequency distribution.

Sequence-only baseline (no state). For comparison, we train a skip-gram model over *tactic sequences* within each proof (window size w), ignoring states entirely (the standard tactic co-occurrence baseline).

2.6 State embeddings and the operator-analogy test

To probe operator-like behavior without a parametric state encoder, we define a simple bag-of-tokens state embedding using the learned table:

$$v(s) \;=\; \frac{1}{|\text{tok}(s)|} \sum_{w \in \text{tok}(s)} e_{\texttt{TOK\_}w}. \quad (5)$$

We then test the *analogy* by checking whether $v(s_{\text{before}}) + e_t$ aligns with $v(s_{\text{after}})$ in the learned space. This mirrors translation tests in distributional semantics and knowledge-graph embeddings, but here the translation vector is the tactic embedding.

2.7 Training protocol and hyperparameters

We split data by `proof_id` into train/validation/test with ratios $80/10/10$ to avoid leakage across steps of the same proof. To mitigate tactic imbalance (e.g., the tactics `simp` and `have` are very frequent), we cap training steps per tactic at a maximum K (we use $K = 5000$). The validation/test are left untouched. Unless otherwise stated, we use the following settings (chosen for stability and speed):

- **Dimensions:** $d = 256$ for Δ-SGNS and CBOW-Δ; $d = 128$ for the sequence baseline.
- **Negative sampling:** 15 negatives per positive for Δ-SGNS and CBOW-Δ; 10 for the sequence baseline; negatives drawn from unigram$^{3/4}$.

- **Optimization:** learning rate 0.03; gradient clipping at norm 5.0; 15 epochs (Δ-SGNS/CBOW-Δ and sequence baseline).
- **Windows:** sequence baseline window $w = 4$.

3 Experiments and Results

Our goal is to verify the central hypothesis that *a tactic is defined by its effect on the proof state*. To attribute performance to specific sources of signal and to rule out alternative explanations (e.g., sequence co-occurrence or name leakage), we run three ablations that toggle which Δ-features are visible to the model: (1) **Tokens-only** — contexts contain *only* added surface tokens ($\texttt{TOK\_}w$). This tests whether surface additions alone suffice to identify tactics. (2) **Typed-only** — contexts contain *only* typed structural edits (ΔADD_HYP, ΔGOAL_SOLVED, relation flips, operator-count changes). This tests whether structural signals alone are sufficient. (3) **Full** — *both* tokens and typed edits. This tests complementarity and the necessity of structural cues for best performance.

We evaluate two Δ-aware models (Δ-SGNS and CBOW-Δ) and a sequence-only SGNS baseline (no states) in order to separate *state-change* supervision from *tactic co-occurrence*.

3.1 Controls against leakage and distributional artifacts

All ablations share the same proof-level split and test cardinalities, so differences reflect the availability of Δ-features rather than label-set changes or sample-size effects. Local-name anonymization and pretty-print canonicalization further reduce the chance of learning from spurious surface cues (e.g., hypothesis names). Together, these controls support the conclusion that the observed gains derive from modeling *state change*.

3.2 Δ → tactic retrieval

We report test-set Mean Reciprocal Rank (MRR) and Recall@k for both Δ-SGNS (center=tactic, contexts=Δ-tokens) and CBOW-Δ (contexts→tactic). Table 1 shows the results. CBOW-Δ shows clear complementarity: *Full* outperforms *Tokens-only* and *Typed-only* (MRR 0.085 vs. 0.068/0.037), a ∼25% relative gain over Tokens-only, indicating that typed edits add non-redundant information beyond surface additions. *Typed-only* retains non-trivial accuracy (MRR 0.037), implying structural cues are necessary but not sufficient. *Tokens-only* captures surface regularities (rewrites, symbol traces) but misses generalization afforded by typed edits. On Δ-SGNS, *Tokens-only* > *Full* suggests skip-gram with a shared table is sensitive to heterogeneous context types. Simple gating or reweighting of typed cues likely will help when using SGNS.

	Δ-SGNS				CBOW-Δ			
Ablation	MRR	R@1	R@5	R@10	MRR	R@1	R@5	R@10
Full	0.032	0.017	0.047	0.054	**0.085**	**0.058**	**0.116**	**0.144**
Tokens	**0.039**	**0.021**	**0.056**	**0.069**	0.068	0.051	0.085	0.095
Typed	0.009	0.004	0.009	0.012	0.037	0.012	0.053	0.086

Table 1: Test Δ → tactic retrieval. Higher is better.

3.3 Operator behavior

We test whether $v(s_{\text{before}}) + e_t$ aligns with $v(s_{\text{after}})$ in the learned space (cosine and rank among after-states). Table 2 shows these results. Typed edits *improve* alignment when combined with tokens: *Full* surpasses *Tokens-only* (cosine 0.67 vs. 0.62; rank 422.3 vs. 429.2, lower is better). Under *Typed-only*, the analogy metric is not meaningful because state embeddings $v(\cdot)$ aggregate $\texttt{TOK\_}w$ vectors that are *untrained* in that ablation (cosine ≈ 0, rank ≈ 1 by construction).

Ablation	Avg Cosine	Avg Rank (lower↓)	N
Full	0.67	422.3	2590
Tokens-only	0.62	429.2	2590
Typed-only	0.00	1.0 †	2590

Table 2: Operator-analogy for Δ-SGNS: $\text{cosine}\big(v(s_{\text{before}}) + e_t, v(s_{\text{after}})\big)$ and rank of the true after-state among test after-states.

3.4 Sequence-only baseline (control)

A tactic-sequence SGNS achieves next-tactic MRR = 0.081 on N=525 bigrams. While this is a *different* prediction target, it serves as a control: co-occurrence alone does not explain Δ → tactic performance, and the best CBOW-Δ MRR is achieved on a strictly harder task that conditions on state edits rather than neighboring tactics.

3.5 Structure of tactic embeddings

Figure 1 visualizes 2D UMAP projections of the 256-dimensional tactic embeddings learned under the *Full* setting for CBOW-Δ (Fig 1a), Δ-SGNS (Fig 1b), and the sequence-only baseline (Fig 1c).

These UMAP projections show that the CBOW-Δ and Δ-SGNS spaces exhibit visually separable clusters, consistent with effect-grounded organization of tactics. By contrast, the sequence-only baseline forms a diffuse, nearly isotropic cloud with no obvious cluster structure under the same projection. These observations support our claim that co-occurrence–only objectives, which can work well in natural-language settings, do not induce discriminative tactic representations in Lean to the same extent as Δ-aware training.

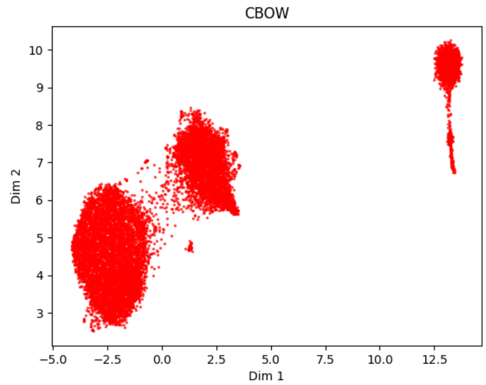
4 Discussion and Conclusion

This paper introduced a representation-learning perspective in which tactics are modeled as operators defined by their effects on proof states. By leveraging Δ-supervision—token-level additions/removals and typed structural edits—we trained lightweight CBOW and skip-gram models that learn embeddings grounded in state transitions. Our experiments demonstrated that these embeddings capture tactic semantics more robustly than surface-only or sequence-based baselines. Typed edits, in particular, provided complementary information that improved generalization and interpretability, enabling operator-style behaviors such as tactic-family clustering and state-translation analogies.

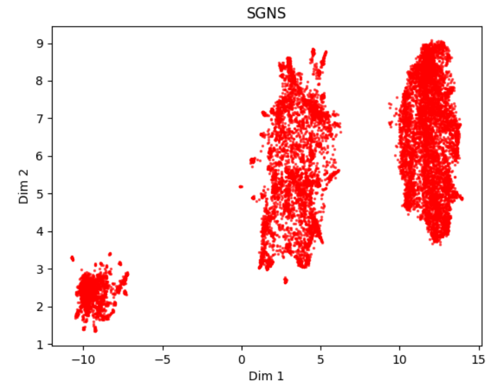
Beyond empirical gains, the central contribution lies in showing that surface-level distributional regularities, often sufficient in natural language modeling, are inadequate for theorem proving. Capturing the semantics of tactics requires effect-grounded representations that model the transformations they induce. While our models are deliberately minimal, they establish a lightweight, interpretable bridge between symbolic proof states and neural methods, complementary to large-scale LLM fine-tuning and retrieval-augmented provers. These findings motivate future work on scaling Δ-supervision to larger datasets, enriching structural edits, and integrating operator embeddings into end-to-end proof search systems.

Limitations

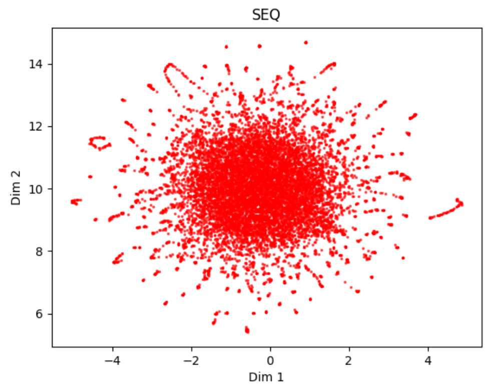
Our study is a proof-of-concept with several limitations. First, we restrict ourselves to a relatively small subset of the Lean Workbook corpus, leaving open whether results scale to larger, more diverse mathematical domains. Second, our models are intentionally minimal and ignore contextual subtleties such as multi-tactic interactions, proof search heuristics, or long-range dependencies. Third, the Δ-contexts rely on simple tokenization and regex-based typed edits, which may omit more nuanced structural information (e.g., type-class resolution or meta-variable instantiation). Finally, we evaluate on proxy tasks such as retrieval and anal-



(a) CBOW



(b) SGNS



(c) SEQ baseline

Figure 1: UMAP visualizations of the 256-dimensional embeddings for CBOW, SGNS, and SEQ (sequence-only baseline) under the Full setting.

ogy, which, while informative, are not substitutes for downstream proof success. Addressing these limitations will require richer state encodings, integration with stronger search strategies, and broader evaluation benchmarks.

Ethical considerations

This work focuses on mathematical theorem proving in Lean and does not involve human subjects, sensitive data, or societal risks.

Acknowledgment

This work was partially supported by student research awards from the Neukom Institute for Computational Science, the Lovelace Research Program, and the Leslie Center at Dartmouth College.

References

Kshitij Bansal, Sarah Loos, Markus Rabe, Christian Szegedy, and Stewart Wilcox. 2019. HOList: An environment for machine learning of higher order logic theorem proving. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 454–463. PMLR.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26.

Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, and Michael Norrish. 2021. Tactictoe: Learning to prove with tactics. *Journal of Automated Reasoning*, 65(2):257–286.

InternLM. 2024. Lean workbook (and lean workbook plus) dataset. `https://huggingface.co/datasets/internlm/Lean-Workbook`. Hugging Face dataset card; reports ∼57k Lean Workbook and ∼83k Lean Workbook Plus problems.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26.

Kaiyu Yang and Jia Deng. 2019. Learning to prove theorems via interacting with proof assistants. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6984–6994. PMLR.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models. In *NeurIPS 2023 Track on Datasets and Benchmarks*. Paper available at NeurIPS Datasets and Benchmarks.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Lean workbook: A large-scale lean problem set formalized from natural language math problems. *arXiv preprint arXiv:2406.03847*.

A Worked Example: Building a Δ-Context

We illustrate Sections 2.3–2.4 by reconstructing the Δ-context for the proof step:

State-Before

a b c : $\mathbb{R}$

ha : $0 < a$ hb : $0 < b$ hc : $0 < c$

habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$

$\vdash \dfrac{a^3}{(1 - a^8)} + \dfrac{b^3}{(1 - b^8)} + \dfrac{c^3}{(1 - c^8)} \geq \dfrac{9}{8}$

Tactic

```
have h1 := sq_nonneg (a^2 - 1)
```

State-After

a b c : $\mathbb{R}$

ha : $0 < a$ hb : $0 < b$ hc : $0 < c$

habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$

h1 : $0 \leq (a^2 - 1)^2$

$\vdash \dfrac{a^3}{(1 - a^8)} + \dfrac{b^3}{(1 - b^8)} + \dfrac{c^3}{(1 - c^8)} \geq \dfrac{9}{8}$

Step 1: Tokenization (Sec. 2.3). We split on Lean/Unicode symbols and mathematical operators and on identifiers. The subphrase

$$\texttt{h1} : 0 \leq (a^2 - 1)^2$$

is tokenized as

$$[\,\texttt{h1},\ :,\ 0,\ \leq,\ \texttt{(},\ \texttt{a},\ \hat{},\ 2,\ \texttt{-},\ 1,\ \texttt{)},\ \hat{},\ 2\,]$$

Step 2: Alpha-renaming / canonicalization (Sec. 2.3). Local variable and hypothesis names are anonymized to prevent name leakage:

a	$\mapsto$	$\_x1$	b	$\mapsto$	$\_x2$
c	$\mapsto$	$\_x3$	h1	$\mapsto$	$\_h6$
ha, hb, hc, habc, h	$\mapsto$	$\_h1 \ldots \_h5$			

Thus the added hypothesis pretty-prints canonically as

$$\texttt{\_h6} : 0 \leq (\_x1^2 - 1)^2.$$

Step 3: Formatting normalization (Sec. 2.3). We normalize whitespace and numeric formats; there is no effect on the symbol sequence above.

Step 4: Goal/context split (Sec. 2.3). We split at the first ⊢. The new hypothesis belongs to the *context* (left of ⊢); the *goal* (right of ⊢) is unchanged in this step. There are no multiple-goal peculiarities here (multi-goal pooling is unnecessary).

Step 5: Token-level deltas (Sec. 2.4). Let $\Delta(w) = \text{count}_{\textit{after}}(w) - \text{count}_{\textit{before}}(w)$. The only positive deltas arise from inserting the hypothesis line _h6 $: 0 \leq (\_x1^2 - 1)^2$. The multiset of *added tokens* (with multiplicities) is:

$$\texttt{\_h6}^{\times 1}\quad :^{\times 1}\quad 0^{\times 1}\quad \leq^{\times 1}\quad \texttt{(}^{\times 1}\quad \texttt{\_x1}^{\times 1}$$
$$\hat{}^{\times 2}\quad 2^{\times 2}\quad \texttt{-}^{\times 1}\quad 1^{\times 1}\quad \texttt{)}^{\times 1}$$

Step 6: Typed edit indicators (Sec. 2.4). Regex-based structural edits are as follows:

$\Delta_{\texttt{ADD\_HYP}}$	(one hypothesis added)
$\Delta_{\texttt{ADD\_SYM\_-}}$	(a minus sign added)
$\Delta_{\texttt{ADD\_SYM\_}\leq}$	(a ≤ added)

There is no Δ_GOAL_SOLVED and no relation flip (≤↔≥) in this step.

Step 7: Final Δ-context C (Sec. 2.4). The training context is the multiset:

$$\{\texttt{TOK\_\_h6},\ \texttt{TOK\_:},\ \texttt{TOK\_0},\ \texttt{TOK\_}\leq,\ \texttt{TOK\_(},\ \texttt{TOK\_\_x1},\ \texttt{TOK\_\^{}},\ \texttt{TOK\_2},\ \texttt{TOK\_\^{}},\ \texttt{TOK\_2},\ \texttt{TOK\_-},\ \texttt{TOK\_1},\ \texttt{TOK\_)}\,\}$$
$$\cup\ \{\Delta_{\texttt{ADD\_HYP}},\ \Delta_{\texttt{ADD\_SYM\_-}},\ \Delta_{\texttt{ADD\_SYM\_}\leq}\}$$

(Repeated TOK_^ and TOK_2 indicate multiplicity.)

B Examples

Table B1 shows sample lean proofs with different number of tactics. Table B2 shows examples of State-Before, Tactic, State-After triples from the Lean Work-book Corpus.

Proof #	Tactics

1	`norm_num [ha, hb, hc, habc, h]`
	`have h1 := sq_nonneg (a^2 - 1)`
	`have h2 := sq_nonneg (b^2 - c^2)`
	`nlinarith`

2	`rintro a b c ⟨h1, h2, h3, h4, h5, h6⟩` `nlinarith [sq_nonneg (a - b), sq_nonneg (b - c), sq_nonneg (c - a)]`
3	`push_neg` `refine' ⟨1, 1, ⟨by norm_num, by norm_num⟩, by norm_num⟩`
4	`push_neg` `refine' ⟨1, 2, 3, by norm_num⟩`
5	`simp [Int.ModEq]` `omega`

Table B1: Sample proofs. Each line is one tactic.

State-Before	Tactic	State-After
x y z : ℤ ⊢ (x^2 + 1) * (y^2 + 1) * (z^2 + 1) = (x + y + z)^2 - 2 (x y + y z + z x) + (x y + y z + z x)^2 - 2 x y z (x + y + z) + x^2 y^2 z^2 + 1	ring	no goals
a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ ⊢ $a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8) \geq \frac{9 \cdot 3^{1/4}}{8}$	`norm_num [ha, hb, hc, habc, h]`	a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$
a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$	have h1 := `sq_nonneg (a^2 - 1)`	a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ h1 : $0 \leq (a^2-1)^2$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$
a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ h1 : $0 \leq (a^2-1)^2$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$	have h2 := `sq_nonneg (b^2 - c^2)`	a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ h1 : $0 \leq (a^2-1)^2$ h2 : $0 \leq (b^2-c^2)^2$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$
a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ h1 : $0 \leq (a^2-1)^2$ h2 : $0 \leq (b^2-c^2)^2$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$	nlinarith	no goals

Table B2: Sample sentences (before, tactic, after) from the Before-Tactic-After dataset.

UniMath-CoT: A Unified Framework for Multimodal Mathematical Reasoning with Re-Inference Affirmation

Zhixiang Lu[1], Mian Zhou[1], Angelos Stefanidis[1], Jionglong Su[1],

[1]School of Artificial Intelligence and Advanced Computing, Xi'an Jiaotong-Liverpool University

Correspondence: Jionglong.Su@xjtlu.edu.cn

Abstract

Large Language Models (LLMs) have achieved considerable success in text-based mathematical reasoning, yet their potential remains underexplored in the multimodal mathematics domain where joint text and image understanding is imperative. A key bottleneck hindering progress is the scarcity of high-quality, genuinely multimodal benchmarks. To address this gap, we construct a unified benchmark by consolidating and curating three public multimodal mathematics datasets. We subsequently propose the **UniMath-CoT** framework, which establishes a robust performance baseline by combining Chain-of-Thought (CoT) principles with efficient Supervised Fine-Tuning (SFT) based on Low-Rank Adaptation (LoRA). Furthermore, to bolster the model's reasoning robustness, we introduce an innovative verification mechanism, **AARI (Answer Affirmation by Re-Inference)**, which leverages a specialized re-inference protocol to have the model self-scrutinize and validate its initial conclusions. Our comprehensive experiments show that this integrated strategy substantially boosts performance, surpassing a wide range of open-source models and markedly closing the gap with leading proprietary systems.

1 Introduction

Mathematical reasoning is a cornerstone of human intelligence. Automating this process, particularly for problems presented in a multimodal format that integrates text with diagrams and figures, represents a significant frontier for artificial intelligence (Seo et al., 2015). This capability has profound implications for domains like personalized education, scientific discovery, and engineering, where complex information is often conveyed visually.

The advent of powerful Vision-Language Models (VLMs), such as GPT-4V, Gemini, and LLaVA (OpenAI, 2023; Team et al., 2023; Liu et al., 2024a), has opened new avenues for tackling this challenge. These models can, in principle, process interleaved text and images to produce human-like reasoning steps, moving beyond the text-only limitations of earlier models (Wei et al., 2022). However, the landscape of multimodal mathematical reasoning remains fragmented. Numerous datasets exist (Lu et al., 2024; Ling et al., 2023; Liu et al., 2024b; Wang et al., 2024), but they possess distinct formats, scopes, and a significant number of instances where the visual component is not essential for solving the problem. This fragmentation and lack of a unified, high-quality benchmark impede the rigorous evaluation of VLMs and hinder progress in the field. To address these challenges and systematically advance the state of multimodal mathematical reasoning, this paper makes three primary contributions:

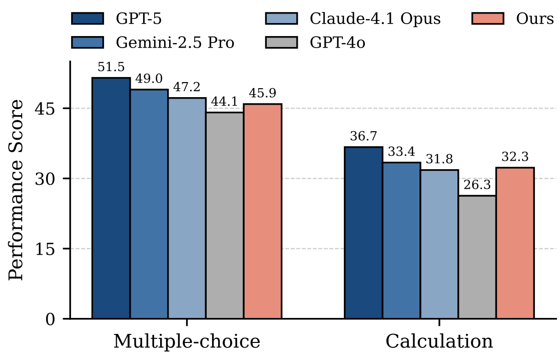


Figure 1: Performance comparison of our model against leading proprietary VLMs on Multiple-choice and Calculation problems. "Ours" refers to the Qwen2.5-VL 7B model fine-tuned with UniMath-CoT and enhanced with AARI.

1. **A Unified and Curated Multimodal Mathematics Dataset.** We construct a new benchmark by amalgamating three prominent datasets: MathViTa, MathVision, and CMM-MATH. We apply a rigorous filtering process to remove unimodal instances, ensuring that

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 176–185

every problem is genuinely multimodal. The resulting dataset serves as a more challenging and realistic testbed for evaluating VLM reasoning capabilities.

2. **An Effective Reasoning Framework (UniMath-CoT).** We leverage and validate UniMath-CoT, a structured Chain-of-Thought approach tailored for the complexities of joint visual and textual understanding. Implemented via parameter-efficient fine-tuning (LoRA) (Hu et al., 2021), this framework guides the model to generate more coherent and accurate reasoning paths than conventional methods.

3. **A Novel Answer Verification Technique (AARI).** We introduce AARI, a lightweight, post-hoc strategy to boost model performance. This technique directs the model to re-evaluate its predicted answer through a secondary, condensed inference pass, acting as a powerful self-correction mechanism that significantly reduces errors and improves final accuracy.

Table 1: Distribution of samples by split, source, and language. Each cell shows the absolute count and its proportion relative to the split's total count.

Split	Dataset	Language	
		Chinese	**English**
Train	MathVision	0 (0.0%)	3157 (23.8%)
	MathVista	357 (2.7%)	5428 (40.9%)
	EduChat-Math	4255 (32.1%)	61 (0.5%)
	Total	**4612 (34.8%)**	**8646 (65.2%)**
Test	MathVision	0 (0.0%)	187 (18.7%)
	MathVista	45 (4.5%)	311 (31.1%)
	EduChat-Math	455 (45.5%)	2 (0.2%)
	Total	**500 (50.0%)**	**500 (50.0%)**

2 Related Works

Multimodal Math Datasets The development of capable models for mathematical reasoning is intrinsically linked to the availability of high-quality datasets. Early efforts in this domain often focused on text-based problems, with benchmarks like GSM8K (Cobbe et al., 2021) and MathQA (Amini et al., 2019) becoming standard testbeds for evaluating the reasoning capabilities of LLMs.

The frontier has recently shifted towards multimodal problems that require understanding both text and images. This has led to the creation of several key benchmarks. For instance, Geometry3K (Lu et al., 2021) provides high-school level geometry problems with formally annotated diagrams. More recently, a new wave of diverse datasets has emerged, including **MathVista** (Lu et al., 2024), a comprehensive benchmark aggregating problems from 28 different sources; **MathViTa** (Ling et al., 2023), which focusing on visual instruction tuning for math; **MathVision** (Wang et al., 2024), designed to test reasoning-intensive math problems; and **CMM-MATH** (Liu et al., 2024b), a benchmark specifically for Chinese multimodal mathematics.

While these datasets have been invaluable, their varied formats, scopes, and annotation styles create a fragmented landscape. This makes it challenging to perform unified and fair evaluations of different models. Our work addresses this gap directly by curating and unifying three of these recent, diverse datasets into a single, filtered benchmark, ensuring all instances are genuinely multimodal and providing a more robust foundation for analysis.

Vision-Language Models (VLMs) The advent of powerful VLMs, pioneered by models like LLaVA (Liu et al., 2024a) and further advanced by open-source models like Qwen-VL (Bai et al., 2023) and proprietary systems like GPT-4V (OpenAI, 2023) and Gemini (Team et al., 2023), has enabled end-to-end multimodal reasoning. The primary challenge has since shifted to effectively eliciting their latent reasoning abilities.

The foundational technique for this is Chain-of-Thought (CoT)(Wei et al., 2022) prompting (Wei et al., 2022), which significantly improves performance by instructing models to generate step-by-step solutions. This paradigm has been extended to the multimodal domain, with methods like Multimodal-CoT (Zhang et al., 2023) that prompt the model to integrate information from both modalities in its reasoning steps. While effective, these zero-shot prompting methods can be sensitive to prompt formulation and may not be optimal for a specific domain. An alternative is to instill reasoning capabilities through training. Our *UniMath-CoT* framework aligns with this direction, employing supervised fine-tuning (SFT) (Ouyang et al., 2022) to teach the model a specialized reasoning structure for multimodal math, aiming for a more robust and replicable performance than prompting alone.

Self-Correction and Answer Verification A key limitation of LLMs, even when using CoT, is their propensity for making logical or computational errors within a reasoning chain. To mitigate this, research has explored methods for self-correction and answer verification (Lu et al., 2025; Li et al., 2025). One approach involves training a separate verifier model to score or judge the correctness of solutions generated by a primary model (Cobbe et al., 2021). Another popular direction is self-refinement, where the model iteratively critiques and improves its own output based on feedback (Madaan et al., 2023).

While powerful, these methods can be computationally expensive, requiring either the training of an additional model or multiple inference passes. Our proposed AARI technique is situated within this line of research but designed for efficiency. It is a lightweight, single-pass verification step that prompts the model to perform a final, focused check on its answer. AARI functions as a post-hoc self-correction mechanism that improves reliability without the overhead of multi-step refinement or external verifiers.

3 Methodology

Our methodology is designed to systematically enhance and evaluate the mathematical reasoning capabilities of Vision-Language Models (VLMs). It is founded on three core components: a newly curated benchmark, a fine-tuned reasoning framework, and a novel inference strategy for answer verification as shown in Figure 2.

3.1 UniMath Benchmark Construction

A robust evaluation requires a high-quality benchmark. To this end, we construct **UniMath**, a unified and curated benchmark derived from three recent and diverse multimodal math datasets: MathVista (Lu et al., 2024), MathVision (Wang et al., 2024), and CMM-MATH (Liu et al., 2024b). The construction process involves four critical steps:

1. **Schema Unification:** All problems from the source datasets are converted into a standardized JSON format. Each entry contains a unique ID, the raw question text, an image, the ground-truth answer, and, where available, human-annotated reasoning steps. This creates a consistent data structure for all subsequent processing.

2. **Answer Normalization:** We develop and apply a rigorous parsing function, $\phi(\cdot)$, to normalize all answers. This function extracts numerical values, correctly interprets fractions and percentages, removes units, and standardizes multiple-choice options (e.g., converting '(A)' to 'A'). This ensures that evaluation is based on mathematical correctness, not superficial formatting differences.

3. **Genuinely Multimodal Filtering:** A key contribution of UniMath is its focus on problems requiring genuine multimodal reasoning. We employ a systematic process to filter out instances where the image is redundant or extraneous, ensuring that a correct solution can only be derived by integrating both visual and textual information.

4. **Problem Type and Scope Curation:** To maintain evaluation clarity and objectivity, we further refine the benchmark by problem type and answer scope. Specifically, we only select two types of problems: multiple-choice questions and non-multiple-choice problems that possess a unique, definitive solution. Furthermore, to avoid ambiguity with extremely large numbers or complex symbolic expressions, we constrain the answers for all non-multiple-choice problems to be rational numbers within the range of `[-10,000, 10,000]`.

The final UniMath benchmark, shaped by this rigorous curation process, serves as the foundation for our experiments, providing a challenging and standardized testbed for multimodal mathematical reasoning.

3.2 UniMath-CoT Reasoning Framework

To move beyond the limitations of zero-shot prompting, we propose UniMath-CoT, a framework for teaching a VLM to generate structured, step-by-step reasoning for multimodal math problems through supervised fine-tuning (SFT).

The goal of UniMath-CoT is to train the model to produce a specific, decomposable reasoning chain that mirrors an ideal problem-solving process (see Figure 3). This chain consists of several stages: (1) *Visual Grounding*, where key information from the image is extracted; (2) *Problem Formulation*, where visual and textual information are integrated; (3) *Step-by-Step Planning*; (4) *Execution* of the plan with calculations; and (5) the *Final Answer*.

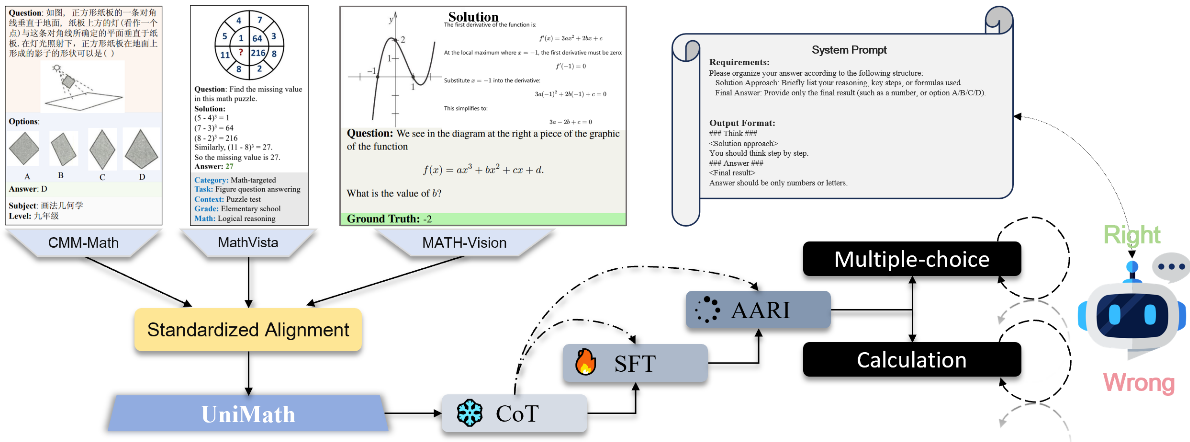


Figure 2: The overall framework of our work, encompassing the entire pipeline from data construction to model training and inference-verification. We first integrate three public datasets (CMM-Math, MathVista, MATH-Vision) through standardized alignment to build the unified **UniMath** benchmark. Subsequently, this benchmark is used to generate CoT formatted samples for SFT of the base VLM. During inference, the preliminary answer from the SFT model is passed through our proposed **AARI** module for re-inference and verification to produce a more reliable final result.

UniMath-CoT for Multimodal Mathematical Reasoning

Input: You are an expert skilled in mathematical reasoning. Your task is to analyze the given math problem and its type, and then provide the solution. The problem type is: {Question type} {Question image}
Requirements:
1. All mathematical symbols and formulas must be written using LaTeX syntax.
2. Absolutely no factual mistakes are allowed (for example, errors like $\ln 1 = 1$ will result in termination of cooperation).
3. Please organize your answer according to the following structure:
Solution Approach: Briefly list your reasoning, key steps, or formulas used.
Final Answer: Provide only the final result (such as a number, or option A/B/C/D).
Output Format:
Think
<Solution approach (in LaTeX format)>
- Step 1: $formula/derivation$
- Step 2: $formula/derivation$
- ...
■ Note: If it is a multiple-choice question, include the verification process. For example, if you derive that the answer is 20 and option B is 20, then the final Answer must be "B".
Answer
<Final result, only numbers or letters>
IMPORTANT:
- If the final answer is a number, it must be output as a floating-point number, not as an expression or fraction (e.g., `$\frac$`, `{frac} = 0.5`, or `1/2` are forbidden).
- If the answer is for a multiple-choice question, it must be a capital letter such as "C", not "C.9".
- Violating these rules in the final answer will result in termination of cooperation!
[Examples]
■ Calculation question (correct output):
Think
...(derivation leading to 1)
Answer
1
■ Multiple-choice question (correct output):
Think
...(derivation showing B is correct)
Answer
B
Now strictly follow these rules to handle the problem.

Figure 3: The structured reasoning generation prompt template for UniMath-CoT.

Formally, we define a problem instance as a tuple $P = (I, Q)$, where I represents the image and Q is the corresponding textual question. The desired output is a coherent reasoning chain $R = (s_1, s_2, \ldots, s_n)$ that culminates in the final answer A. For training purposes, we concatenate these elements into a single target sequence $Y = (s_1, \ldots, s_n, A)$, which the model must learn to generate autoregressively.

Our primary objective is to fine-tune a base Vision-Language Model (VLM), parameterized by θ, to maximize the conditional likelihood of generating this ground-truth sequence Y given the problem instance P. To achieve this in a computationally efficient manner, we employ Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning.

The training process minimizes the standard cross-entropy loss, which is equivalent to minimizing the negative log-likelihood of the target sequence. The loss function, $\mathcal{L}_{\text{SFT}}$, over our curated training dataset $\mathcal{D}_{\text{train}}$ is formally expressed as the expectation of this loss across all data samples:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(P,Y)\sim\mathcal{D}_{\text{train}}}\left[\log p_\theta(Y|P)\right] \quad (1)$$

where the log-likelihood of a single sequence is decomposed autoregressively as:

$$\log p_\theta(Y|P) = \sum_{t=1}^{|Y|} \log p_\theta(y_t|y_{<t}, P) \quad (2)$$

Algorithm 1 Answer Affirmation by Re-Inference

Require: Image I, question Q, initial answer A, prompt P, VLM $\mathcal{M}$, mode $\mu \in \{t_1, t_2\}$
Ensure: Final answer A^*

if $\mu = t_1$ **then** ▷ Multiple-Choice
  $\mathcal{O} = \{O_1, \ldots, O_k\}$
  for $i = 1, \ldots, k$ **do**
    $P \leftarrow$ "Is 'O_i' correct for Q given I?"
    if $\mathcal{M}(P) =$ "Correct" **then**
      return O_i
  return Fallback($\mathcal{O}$)
else if $\mu = t_2$ **then** ▷ Calculation
  $\mathcal{A}_+ \leftarrow \emptyset$
  for $j = 1, \ldots, N$ **do**
    $P \leftarrow$ "Is 'A_j' correct for Q given I?"
    if $\mathcal{M}(P) =$ "Correct" **then**
      $\mathcal{A}_+ \leftarrow \mathcal{A}_+ \cup \{A_j\}$
  if $|\mathcal{A}_+| > 0$ **then**
    return $\arg\max_{A \in \mathcal{A}_+} \#\{\, i : A_i = A \,\}$
  else
    return Fallback($\mathcal{A}$)

This training paradigm compels the model not only to predict the final answer but also to articulate the underlying step-by-step derivation. By optimizing the entire reasoning path, this process endows the model with a specialized and robust capability for structured, multimodal problem-solving.

3.3 Answer Affirmation by Re-Inference

Even with a strong reasoning framework, VLMs can produce fallacious conclusions from otherwise sound reasoning chains. To address this, we introduce a novel lightweight inference technique: AARI, which is a two-stage process designed to verify and self-correct the model's initial conclusion.

Stage 1: Candidate Generation Given a problem $P = (I, Q)$, the fine-tuned model first generates a candidate solution, which includes the reasoning chain R_1 and a preliminary answer A_1. This is the standard generative process:

$$(R_1, A_1) = \underset{R,A}{\operatorname{argmax}}\ P_\theta(R, A | I, Q) \quad (3)$$

Stage 2: Affirmation via Re-Inference Next, instead of immediately accepting A_1, we formulate a new verification prompt, P', which contains the original problem, the generated reasoning, and the candidate answer: $P' = (I, Q, R_1, A_1)$. We then task the model with assessing the validity of A_1 given R_1. Let V be a latent variable representing the affirmation of the answer, where $V = 1$ signifies correctness and $V = 0$ signifies an error. The model implicitly computes the posterior probability $P_\theta(V = 1|P')$. The final answer, A_f, is determined by this affirmation step:

$$A_f = \begin{cases} A_1 & \text{if } p_\theta(V = 1|P') > 0.5 \\ A_2 & \text{otherwise} \end{cases} \quad (4)$$

where $A_2 = \arg\max_A p_\theta(A|P', V = 0)$. In practice, this is implemented by prompting the model with a question like, "Based on the provided reasoning, is the answer 'A_1' correct? Re-examine the steps and confirm." If the model's response is affirmative, we accept A_1. If it is negative, we prompt it to provide the corrected answer, A_2. This re-inference step forces the model to perform a final, focused consistency check, effectively reducing unforced errors without requiring external verifiers or complex iterative refinement.

4 Experiments

4.1 Experimental Setup

We conduct a comprehensive set of experiments to evaluate our proposed framework. All evaluations are performed on our newly constructed UniMath benchmark, which covers a diverse range of multimodal mathematics problems. The primary goals are to (1) assess the individual contributions of the UniMath-CoT fine-tuning strategy and the AARI inference mechanism through ablation studies, and (2) compare our final model's performance against state-of-the-art open-source and proprietary baselines. Our implementation is built upon PyTorch 2.11+. We leverage the Transformers library (v4.53+) for handling the underlying Vision-Language Models. To manage the significant computational requirements of fine-tuning, we employ `DeepSpeed` (Rajbhandari et al., 2020) for distributed training and memory optimization. For our parameter-efficient fine-tuning approach, we use `LoRA` with 64 lora rank, which is implemented with the `bitsandbytes` library (v0.42). The development environment requires `Python 3.11` and `CUDA 12.6`. All fine-tuning experiments were conducted on a server equipped with NVIDIA A100 80GB GPUs, a 32-core AMD EPYC processor, and 128GB of DDR4 memory. This configuration supported a per-device batch size of 8 during the LoRA fine-tuning process.

Table 2: Comprehensive benchmark on the UniMath dataset. We first establish a zero-shot performance leaderboard across proprietary and open-source models (<10B). We then conduct a detailed ablation study on the top-performing open-source model, Qwen2.5-VL, demonstrating the progressive performance gains from Chain-of-Thought (CoT), a standard SFT (LoRA), and finally our AARI method. Our approach, highlighted in gray and marked with a star (★), achieves state-of-the-art results among its peers, rivaling top proprietary systems.

Model / Method	**Problem Type**		**Language**		**Overall**
	Multiple-choice	**Calculation**	**Chinese**	**English**	**Accuracy (%)**
Proprietary Models (Zero-shot)					
GPT-5 (OpenAI, 2024)	**51.5**	**36.7**	**43.5**	**44.7**	**44.1**
Gemini-2.5 Pro (Reid and Team, 2024)	49.0	33.4	40.1	42.3	41.2
Claude-4.1 Opus (Anthropic, 2024)	47.2	31.8	38.8	40.2	39.5
GPT-4o (OpenAI, 2023)	44.1	26.3	33.0	37.4	35.2
Open-Source Models (Zero-shot)					
InternVL3 8B (Chen et al., 2024)	29.1	13.1	21.3	20.9	21.1
+ CoT Prompting	35.0	19.8	28.1	26.7	27.4
+ SFT (LoRA)	36.5	21.9	29.5	28.9	29.2
Qwen2.5-VL 7B (Bai et al., 2024)	30.5	13.7	23.8	20.4	22.1
+ CoT Prompting	35.3	19.5	28.2	26.6	27.4
+ SFT (LoRA)	37.2	21.5	31.4	30.9	31.2
DeepSeek-VL 7B (DeepSeek, 2024)	28.8	11.7	20.9	19.7	20.3
+ CoT Prompting	34.8	19.2	28.0	25.9	27.0
+ SFT (LoRA)	35.8	20.4	28.3	27.9	28.1
Our Method (Based on Qwen2.5-VL 7B)					
+ UniMath-CoT with SFT (LoRA)	39.6	26.8	34.5	31.9	33.2
★ + AARI	45.9	32.3	40.2	38.0	39.1

4.2 Evaluation Metric

Our primary evaluation metric is `Accuracy`, defined as the percentage of correctly solved problems. An answer is considered correct if the model's prediction, after applying a normalization function $\phi(\cdot)$, exactly matches the normalized ground-truth answer. Formally, for a test set $\mathcal{D}_{\text{test}}$, accuracy is defined as:

$$\text{Accuracy} = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{i=1}^{|\mathcal{D}_{\text{test}}|} \mathbb{I}(\phi(\hat{A}_i) = \phi(A_i)) \quad (5)$$

where $\hat{A}_i$ is the predicted answer, A_i is the ground-truth answer, and $\mathbb{I}(\cdot)$ is the indicator function.

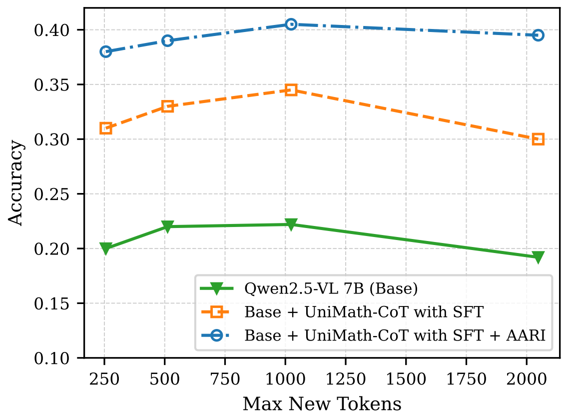


Figure 4: Performance of different model versions under varying maximum new token lengths.

4.3 Results and Analysis

Overall Performance As shown in Table 2, our full approach that combining the UniMath-CoT fine-tuning with the AARI inference strategy, achieves a final accuracy of 39.1%. This result establishes a new state-of-the-art among open-source models on this challenging task. Notably, our model significantly surpasses the powerful proprietary model GPT-4o (35.2%) and becomes highly competitive with Claude-4.1 Opus (39.5%).

The Impact of Each Component To isolate the contributions of our proposed methods, we conducted a detailed ablation study. We evaluated the effectiveness of our fine-tuning framework. Starting from a standard SFT Qwen2.5-VL 7B model which achieves 31.2% accuracy, employing our UniMath-CoT framework boosts the performance to 33.2%, which validates that our specialized approach of training on structured, decomposable

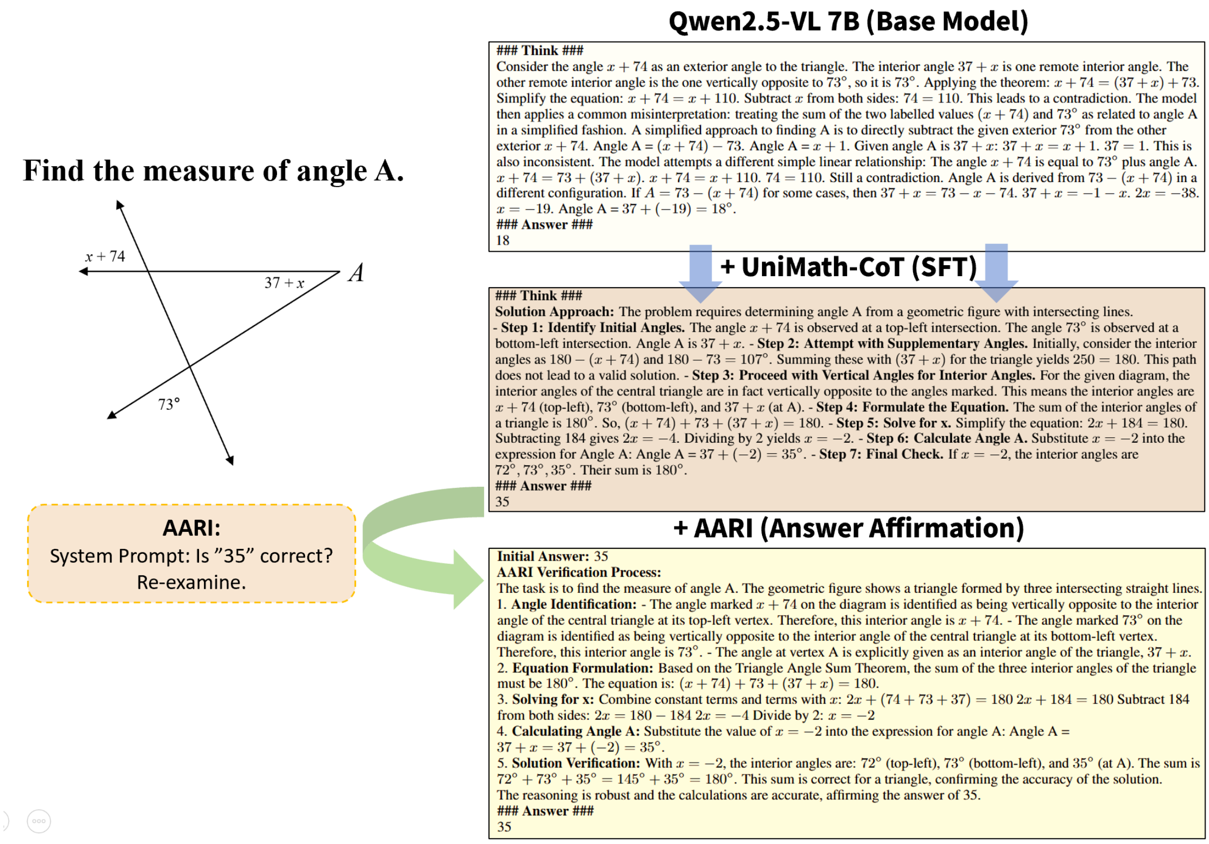


Figure 5: Qualitative Comparison of Model Outputs on a Sample Problem.

reasoning paths is more effective than generic instruction tuning. We also assessed the impact of our inference strategy. Applying AARI on top of the UniMath-CoT model elevates the accuracy from 33.2% to 39.1%. This represents a remarkable +5.9% increase, which corresponds to a 17.8% relative error reduction, underscoring the efficacy of the self-verification mechanism. The benefits of AARI are robust and consistent across all problem categories, boosting accuracy on multiple-choice (+6.3%) and calculation (+5.5%) problems, as detailed in Table 2. This demonstrates that AARI is a broadly applicable technique that enhances model reliability.

Effect of Generation Length We also investigated the impact of the maximum generation length (`Max New Tokens`) on performance. Figure 4 reveals two key findings. First, the performance hierarchy remains consistent across all token lengths, with the base model being outperformed by UniMath-CoT, which is in turn outperformed by the full model with AARI. This visually confirms our ablation results. Second, all model variants achieve their peak performance around 1024 max new tokens, suggesting this length provides an optimal balance between allowing for complete reasoning and avoiding excessive, potentially noisy, output.

Comprehensive Performance Profile To provide a more holistic, multi-dimensional view of our model's capabilities, we present a comparative radar chart in Figure 6. This chart visualizes the trade-offs between model size, inference speed, and performance on different sub-tasks (Chinese, English, Multiple-choice, Calculation) for our model and the baselines.

The chart clearly illustrates the well-rounded and highly efficient profile of our approach. Compared to its base model, **Qwen2.5-VL 7B**, our model shows a significantly larger and more balanced performance polygon. This expansion across all accuracy axes: Chinese, English, Multiple-choice, and Calculation, visually confirms that the gains from UniMath-CoT and AARI are comprehensive and not limited to a single domain.

When compared against leading proprietary models, our model demonstrates remarkable competitiveness despite its significantly smaller parameter size. For instance, while models like **GPT-5** and **Gemini-2.5 Pro** exhibit the largest performance polygons overall, our 7B model achieves an accuracy profile that is notably competitive with, and in some areas superior to, much larger models like **GPT-4o**. This highlights the efficiency of our approach: we have successfully closed a substantial portion of the performance gap with state-of-the-art systems while operating at a fraction of the computational scale. The radar chart thus underscores our primary contribution: a clear and effective methodology for developing highly capable and efficient open-source models for complex multimodal reasoning.

Qualitative Analysis To showcase the differences in multi-step reasoning, Figure 5 provides a qualitative comparison of different models on a geometry problem that requires robust spatial interpretation. As shown, the base model fails due to a misinterpretation of geometric relations, while the UniMath-CoT model, despite an initial flawed step, successfully self-corrects to find the correct solution (Wei et al., 2022). This comparison highlights the critical role of structured reasoning and verification mechanisms, like those in the CoT and AARI models, in achieving reliable and accurate mathematical problem-solving.

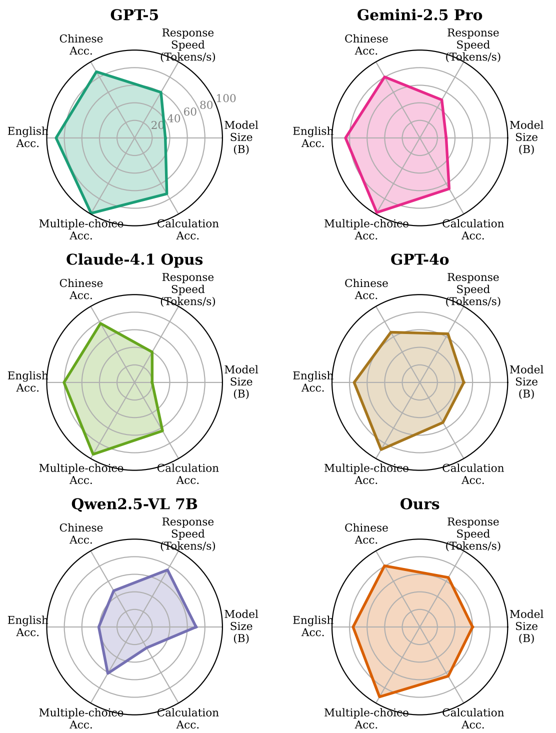


Figure 6: Multi-dimensional performance radar chart comparison. (All accuracy scores are multiplied by 2; parameter sizes for proprietary models are estimates.)

5 Conclusion and Limitation

In this work, we addressed key obstacles in multimodal mathematical reasoning, namely benchmark fragmentation and the need for robust verification. We introduced a tripartite contribution: (1) UniMath, a unified and rigorously curated benchmark for genuinely multimodal math problems; (2) UniMath-CoT, a fine-tuning framework that teaches structured reasoning; and (3) AARI, a novel, lightweight inference-time strategy for answer verification. Our experiments validate the power of this integrated approach. Our 7B model, enhanced by UniMath-CoT and AARI, achieves 39.1% accuracy on the UniMath benchmark, setting a new state-of-the-art for open-source models and outperforming strong proprietary systems like GPT-4o. A key finding is the remarkable effectiveness of AARI, which alone contributes a 5.9 % improvement, drastically reducing errors through its efficient self-verification mechanism.

Our work provides a clear roadmap for building powerful, open-source reasoning systems that rival proprietary models. Crucially, AARI demonstrates that inference-time self-correction is a highly effective strategy for boosting model factuality and reliability, a principle with strong potential for generalization beyond mathematics to other complex domains. Future work can extend this foundation by exploring iterative self-correction mechanisms and expanding the UniMath benchmark to new modalities like video-based challenges.

Ethics Statement

Our work leverages Large Language Models (LLMs) for complex mathematical problem-solving, specifically geometric reasoning, rather than direct text generation. While this application domain typically presents fewer immediate ethical concerns related to content generation biases, it is crucial to acknowledge the broader ethical landscape of LLM usage. Recent investigations have indicated that advanced prompting techniques, such as Chain-of-Thought (CoT) prompting, may inadvertently introduce or amplify ethical biases within LLM reasoning processes, even in non-generative tasks (Shaikh et al., 2023). Therefore, future work will involve a thorough examination of potential biases that might emerge from our method's reliance on CoT and answer affirmation techniques, ensuring fairness and robustness in mathematical reasoning applications.

References

Aida Amini, Mark Hopkins, Tony Tung, Cissy Le, Michael Huth, and Richard Socher. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2957–2967.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. `https://www.anthropic.com/news/the-claude-3-model-family-opus-sonnet-haiku`.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *Preprint*, arXiv:2308.12966.

Jinze Bai, Zhenting Tu, Shusheng Yang, Qian Ma, Yichang Zhu, Shijie Wang, Peng Wang, Sinan Tan, and Chang Zhou. 2024. Qwen2: A family of powerful open-source language models. *arXiv preprint arXiv:2406.04741*.

Zhe Chen, Weiyun Zhang, Wen Wang, Yiliang Liu, Zhaoyang Zhang, Jian Wang, Jie Luo, Yu Qiao, and Wenhai Wang. 2024. Internvl 1.5: A general vision-language model. *arXiv preprint arXiv:2404.16821*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek. 2024. Deepseek-vl: Towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Yulong Li, Zhixiang Lu, Feilong Tang, Simin Lai, Ming Hu, Yuxuan Zhang, Haochen Xue, Zhaodong Wu, Imran Razzak, Qingxia Li, and 1 others. 2025. Rhythm of opinion: A hawkes-graph framework for dynamic propagation analysis. *arXiv preprint arXiv:2504.15072*.

He Ling, Sen Liu, Jian Liu, Yixuan Li, Guangda Shi, Leyang Zhou, Zhaoyang Hu, Yixuan Sun, Xing Su, Jiaxuan Yu, and 1 others. 2023. Mathvita: A visual instruction tuning toolkit for general-purpose multi-modal llms. *arXiv preprint arXiv:2308.03720*.

Haotian Liu, Chunyuan Li, Feiyu Li, and Michihiro Yasunaga. 2024a. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36.

Wentao Liu, Qianjun Pan, Yi Zhang, Zhuo Liu, Ji Wu, Jie Zhou, Aimin Zhou, Qin Chen, Bo Jiang, and Liang He. 2024b. Cmm-math: A chinese multimodal math dataset to evaluate and enhance the mathematics reasoning of large multimodal models. *Preprint*, arXiv:2409.02834.

Pan Lu, Swaroop Mishra, Tony Xia, Liangcheng Huang, Ramana Al-Tawy, Yixin Jia, Wen-haw Zhang, Ping Yang, Mohamed Abdel-Hady, Bassem Majumder, and 1 others. 2024. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*.

Pan Lu, Ran Zhang, Jivko Sashank, AI Wang, Mohit Singh, Hongxin Zhu, and Yi Su. 2021. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2028–2041.

Zhixiang Lu, Peichen Ji, Yulong Li, Ding Sun, Chenyu Xue, Haochen Xue, Mian Zhou, Angelos Stefanidis, Jionglong Su, and Zhengyong Jiang. 2025. Advancing low-resource machine translation: A unified data selection and scoring optimization framework. In *Advanced Intelligent Computing Technology and Applications*, pages 482–493, Singapore. Springer Nature Singapore.

Aman Madaan, Niket Tandon, Prakhar Gupta, Kevin Hall, Luyu Gao, Sreyan Majumder, Julian McAuley, Yiming Narayan, and Gena Sim. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

OpenAI. 2023. Gpt-4v(ision) system card. `https://cdn.openai.com/papers/GPTV_System_Card.pdf`.

OpenAI. 2024. Gpt-4o. `https://openai.com/index/hello-gpt-4o/`.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. *Preprint*, arXiv:1910.02054.

M. Reid and Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Google AI*.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. *arXiv preprint arXiv:1509.04232*.

Omar Shaikh, Hongxin Zhang, William Held, Michael Bernstein, and Diyi Yang. 2023. On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4454–4470, Toronto, Canada. Association for Computational Linguistics.

Gemini Team and 1 others. 2023. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. 2024. Measuring multimodal mathematical reasoning with math-vision dataset. *Preprint*, arXiv:2402.14804.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ben Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Zhuosheng Zhang, Aston Li, Vladislav Lialin, Hai Lu, Hanjung Lee, Mohammad Hosseini, Xiang Li, Haotian Liu, Chunyuan Li, Mu Li, and Anima Anandkumar. 2023. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*.

An in-depth human study of the mathematical reasoning abilities in Large Language Models

Carolina Dias-Alexiou, Edison Marrese-Taylor, Yutaka Matsuo
Graduate School of Engineering, The University of Tokyo
{carolina.dias,emarrese,matsuo}@weblab.t.u-tokyo.ac.jp

Abstract

We study the generalization capabilities of large language models (LLM) through the lens of mathematical reasoning, asking if these models can recognize that two structures are the same even when they do not share the same nomenclature. We propose a human study to evaluate if LLMs reproduce proofs that they have most likely seen during training, but when the symbols do not match the ones seen. To test this in a controlled scenario, we look at proofs in *propositional calculus*, foundational for other logic systems, semantically complete and widely discussed online. We replace the implication operator ($\rightarrow$) with an unrelated, arbitrary symbol ($\spadesuit$) and ask experts to evaluate how the output of a selection of LLMs changes in terms of compliance, correctness, extensiveness and coherence. Our results show that nearly all our tested models produce lower quality proofs in this test, in particular open-weights models, suggesting the abilities of these LLMs to reason in this context have important limitations.

1 Introduction

Mathematical reasoning is a key aspect of human intelligence that encompasses pattern recognition and logical entailment. The development of artificial intelligence systems capable of tasks such as solving applied and theoretical mathematical problems has been a long-standing focus of research in the fields of machine learning and natural language processing, dating back to the 1960s (Feigenbaum and Feldman, 1963; Bobrow, 1964).

Our interest in this topic rises from recent attempts to use language models for theorem proving by means of ITPs' programming languages and databases of theorems with their proofs. Deep learning models can be trained in one of these many programming languages, and then used to generate mathematical proofs. Data sources for neural theorem proving in ITPs include interactive learning environments that interface with ITPs, and datasets derived from proofs in ITP libraries.

When it comes to mathematical reasoning, and in particular to the ability of models to understand logical statements, we note that despite the abundance of studies, previous works generally assume that only "variables" are the ones not constant throughout problems. However, we see that in mathematics different nomenclatures are used in different areas, as well as in different time periods, to express the same ideas. Examples of this fact include the use of $\supset$ and $\rightarrow$ to denote implication; Leibniz's $\frac{\mathrm{d}f}{\mathrm{d}x}$, Lagrange's f', and Newton's $\dot{f}$ notation in differential calculus; the different notation for the inner product for physicists $\langle\cdot|\cdot\rangle$ and for mathematicians $\langle\cdot,\cdot\rangle$; prefix $f(x)$ and postfix $(x)f$ notations for functions; and different notations for the von Neumann generated algebra, such as: $W^*(\cdot,\cdot)$ and $\cdot \vee \cdot$, to name a few.

As we attempt to use LLMs to tackle proof generation tasks, for example using the "informal theorem proving" approach (please see §2 for more details on this), we think researchers and practitioners need to take this fact into consideration. Furthermore, and in contrast to all these models, current state-of-the-art models in NLP are trained on large datasets of text extracted from the web. In this case, we have limited or no control on the kinds of expressions and/or operators that the models are exposed to during training. We can, however, still assume the models have been exposed mostly to the standard nomenclature.

Given this scenario, we ask: *can these models recognize that two structures are the same even if they do not share the same nomenclature?* For example, can models reproduce proofs that they have most likely seen during training, but when the symbols do not match the ones seen? If so, to what extent are these proofs plausible and correct? In order to answer these questions, we perform

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 186–194
November 9, 2025

an in-depth human evaluation to assess the quality of the output generated by LLMs when prompted to generate proofs similar to the ones seen during training but with expressions that use a different notation.

2 Related Work

Automated Theorem Proving The first proofs using computers started appearing as early as the 1950s, soon after electronic computers became available. This played a big role in the development of the field of automated reasoning, which itself led to the development of AI. Most of the early work on computer-assisted proof was devoted to *automated theorem proving* (ATP) (Harrison et al., 2014), in which machines were expected to prove assertions fully automatically. The increased availability of interactive time-sharing computer operating systems in the 1960s allowed the development of *interactive theorem provers* (ITPs) in which the machine and the user work together to produce a formal proof. While ATPs include proof-search algorithms to generate whole proofs, ITPs usually check the validity of human input statements, although they may also include reduced automated tools.

More recently, work on "informal" theorem proving has presented an alternative medium for theorem proving, in which statements and proofs are written in a mixture of natural language and symbols used in "standard" mathematics (e.g., in LaTeX), and are checked for correctness by humans. Here we find the work of Welleck et al. (2021) who developed NaturalProofs, a large-scale dataset of 32k informal mathematical theorems, definitions, and proofs, and provided a benchmark for premise selection via retrieval and generation tasks. Most of the data is taken from websites like `proofwiki.com`, and though this enables more flexibility when proving, the task is approached in a way similar to ITPs.

Mathematical Reasoning in LLMs Early works attempting to study the ability of models to recognize patterns in mathematical expressions focused on the manipulation of simple expressions using standard notation. For example, Allamanis et al. (2017) trained models on datasets in which pairs of examples contain Boolean logic and arithmetic expressions which are known to be equivalent. For example, expressions like c^2 and $(c\cdot c)+(b-b)$ are equivalent. However, expressions with the same structure, but different variables, such as $c\cdot(a\cdot a+b)$ and $f\cdot(d\cdot d+e)$, are not. They showed that such models were capable of relating non-paired expressions, like $a-(b-c)$ and $b-(a+c)$, as negations of each other.

Evans et al. (2018) studied the ability of neural networks to understand logical entailment via training models on synthetic datasets of logical statements and their evaluations (True/False). Concretely, they generated datasets of triples of the form $(A, B, A \vDash B)$, where A and B are formulas of propositional logic, and $A \vDash B$ is 1 if A entails B, and 0 otherwise. They concluded that, from the models available at the time, those with a tree structure seemed to be better for domains with unambiguous syntax.

In these works, expressions are generated automatically starting from a set of simple rules plus a set of arbitrary combinations. This allows to scale and control the types of expressions that are shown to the model during training/inference. Later Cobbe et al. (2021) and Rein et al. (2023) shifted to a question-answer format using natural language, with the release of the GSM8K and GPQA datasets, respectively, while also extending the class of questions to other areas of mathematics, like calculus and probability, where the ability to control for the type of expressions is reduced.

3 Proposed Approach

To study the posed research questions under a controlled scenario, we look at proofs in *propositional calculus*, a branch of formal logic that deals with propositions, which can be true or false, and relations between propositions, including the construction of arguments based on them (Wrenn, 2025). Propositional calculus, also known as zeroth-order logic, does not deal with quantifiers over non-logical objects (unlike first-order or higher-order logic). There are several reasons that we think make this the ideal scenario for our study: (1) All the machinery of propositional logic is included in first-order logic, higher-order logic, and all mathematics. In this sense, propositional logic is the foundation of other logic systems; (2) Propositional calculus is semantically complete, i.e. any tautology (true formulas) can be proved with the formal axioms and the rules of inference of the system; (3) Being the subject of common undergraduate courses, demonstrations in this context have been widely discussed online (for example, in fora such as Math Stack Exchange) so we can reasonably assume that LLMs have been exposed to these types of proofs, and (4) It is a minimalist setup which allows us to include the entire logical structure in the prompt, thus reducing the amount of assumptions needed.

Propositional calculus is typically studied with a formal system, which contains a formal language and a deductive system. The language is composed of a set of well-formed formulas, which are strings of symbols from an alphabet (composed of propositional variables and propositional connectives) formed by a formal grammar (formation rules). The deductive system, in turn, contains the rules of inference, a function which takes premises and returns conclusions. To assess how models generalize in this scenario, we compared proofs generated by these models using "usual" and "unusual" symbols for connectives.

We use a standard proof system usually referred to as a Hilbert system. This is a deductive system that generates theorems from axioms (tautologies taken as starting point for further reasoning) and *modus ponens*. *Modus ponens* can be summarized as: If P implies Q and P is known to be true, one can conclude that Q must also be true. It is generally expressed as $\{P \rightarrow Q,\ P\} \vdash Q$, where the turnstile symbol ($\vdash$) denotes derivability, i.e. there is a formal derivation of a theorem from the axioms. As for connectives, we limit it to the logical and ($\wedge$), logical or ($\vee$), the negation operator ($\neg$), and the implication operator ($\rightarrow$). For the axioms, we use a common set of 14 axioms used in undergraduate courses, shown in Figure 1.

For our study, we propose to replace the implication operator ($\rightarrow$) with an unrelated, arbitrary symbol (♠). In order to produce a significant perturbation in the input token distribution, we specifically select the unicode representation of the symbol (U+2660) for the replacement. Alternative replacements are left for future work. We select two common theorems from propositional calculus extracted from Rossegger (2019), as shown in Figure 2 and test models in two different scenarios, as follows.

Full Context (FC) Our first evaluation scheme is intended to simulate a noisy retrieval step prior to the proof generation. Concretely, we offer the model the complete set of axioms together with the selected rule of inference, *modus ponens*. Thus, in this scenario, we can also test the model's ability to identify and retrieve only the axioms that are needed to prove the selected theorem.

Axioms

$$
\begin{aligned}
(\text{Ax } 1) &: ((\phi \wedge \psi) \rightarrow \phi)\\
(\text{Ax } 2) &: ((\phi \wedge \psi) \rightarrow \psi)\\
(\text{Ax } 3) &: (\phi \rightarrow (\psi \rightarrow (\phi \wedge \psi)))\\
(\text{Ax } 4) &: (\phi \rightarrow (\phi \vee \psi))\\
(\text{Ax } 5) &: (\phi \rightarrow (\psi \vee \phi))\\
(\text{Ax } 6) &: ((\phi \rightarrow \chi) \rightarrow ((\psi \rightarrow \chi) \rightarrow ((\phi \vee \psi) \rightarrow \chi)))\\
(\text{Ax } 7) &: (\phi \rightarrow (\psi \rightarrow \phi))\\
(\text{Ax } 8) &: ((\phi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\phi \rightarrow \psi) \rightarrow (\phi \rightarrow \chi)))\\
(\text{Ax } 9) &: ((\phi \rightarrow \psi) \rightarrow ((\phi \rightarrow \neg\psi) \rightarrow \neg\phi))\\
(\text{Ax } 10) &: (\neg\phi \rightarrow (\phi \rightarrow \psi))\\
(\text{Ax } 11) &: (\phi \vee \neg\phi)\\
(\text{Ax } 12) &: ((\phi \wedge \neg\phi) \rightarrow \psi)\\
(\text{Ax } 13) &: ((\phi \rightarrow (\psi \wedge \neg\psi)) \rightarrow \neg\phi)\\
(\text{Ax } 14) &: (\neg\neg\phi \rightarrow \phi)\\
(\text{MP}) &: \{P \rightarrow Q, P\} \vdash Q
\end{aligned}
$$

Figure 1: Portion of the prompt provided to the LLMs showing the content of the full context provided, namely, the axioms and rules of inference we allow the models to use.

Selected Context (SC) We assume that the relevant axioms for the requested proof have already been selected by an oracle, and we offer only these axioms and rule of inference to the model input. For each question, we manually select the axioms required (Axioms 6, 7, 10 for Question A; Axioms 7, 8 for Question B). In practice, we reassign identifier numbers to them, always starting from 1, to avoid ambiguity.

A key point of our study is to ensure that the generated proofs are checked by mathematicians. Previous work has stressed the need to rely on experts for evaluation of theorem proving systems, including the work of Welleck et al. (2022), who carried on an in-depth annotation where an expert annotator is presented with the theorem, proof-so-far, and a generated next-step. Frieder et al. (2023) also highlight that human evaluation of advanced mathematics that approaches research level is expensive and requires experts. The evaluation of the output of the language model for their work was performed by the authors, who are all mathematicians.

To perform the evaluation, we concretely rely on a volunteer (one of the authors of this paper) who has a Master's degree in mathematics. We design an annotation interface where for each case, we show the annotator the exact input fed to the model,

Question A

Question: Prove that

$$\vdash ((\neg P \vee Q) \to (P \to Q))$$

Answer:

$$((\neg P \to (P \to Q)) \to ((Q \to (P \to Q)) \to$$
$$((\neg P \vee Q) \to (P \to Q)))) \quad \text{(Ax 6)}$$
$$(\neg P \to (P \to Q)) \quad \text{(Ax 10)}$$
$$((Q \to (P \to Q)) \to ((\neg P \vee Q) \to (P \to Q))) \quad 1, 2 \text{ MP}$$
$$(Q \to (P \to Q)) \quad \text{(Ax 7)}$$
$$((\neg P \vee Q) \to (P \to Q)) \quad 3, 4 \text{ MP}$$

Question B

Question: Prove that

$$\{(P \to Q), (Q \to R)\} \vdash (P \to R)$$

Answer:

$$(P \to Q) \quad \text{(hyp)}$$
$$(Q \to R) \quad \text{(hyp)}$$
$$((Q \to R) \to (P \to (Q \to R))) \quad \text{(Ax 7)}$$
$$(P \to (Q \to R)) \quad 2, 3 \text{ MP}$$
$$((P \to (Q \to R)) \to ((P \to Q) \to (P \to R))) \quad \text{(Ax 8)}$$
$$((P \to Q) \to (P \to R)) \quad 4, 5 \text{ MP}$$
$$(P \to R) \quad 1, 6 \text{ MP}$$

Figure 2: Details of the question (top: Question A, bottom: Question B) utilized for our study, also showing a possible proof which we allow our annotators to see.

as well as the output generated by it. Below, we list the tasks that we require our annotator to perform.

- First, we ask the annotators if the output of the model contains a proof.
- We present the steps of a correct proof and ask the annotators to judge if each step appears in the output of the model. Additionally, if the step invokes an axiom or rule of inference, we ask them to do the following.
  - If the step invokes an axiom, we ask to check if variables were substituted correctly.
  - If the step uses a rule of inference, we ask to check if the rule is properly invoked.
  - We ask the annotator to judge whether the step contributes to the proof in the sense that it stirs the overall flow of the proof in the right direction towards conclusion.
- With respect to the clarity of the overall text, we ask the annotators to rate the output on a scale of 0 to 4 via the following labels: "Very Incoherent, Incoherent, Neither Coherent nor Incoherent, Coherent, Very Coherent".
- In order to evaluate the compliance to the task, we ask whether the proof attempts to use information other than the necessary elements that were provided to the model as input. Concretely, we ask the annotators to indicate if the proof uses an additional axiom, if this axiom was provided in the input, and if it uses an additional rule of inference, or an additional hypothesis.
- Finally, we also allow the annotators to freely provide us specific feedback by highlighting spans of the model output that calls their attention, to which they can add free-text comments.

The above questions were designed to incorporate most, if not all, aspects that one would take into consideration when grading the same questions in an exam.

4 Results

For this study, we consider the following models: (1) API-based LLMs, including ChatGPT (*gpt-3.5-turbo-0125*) (Brockman et al., 2023), Claude 3 Opus (*claude-3-opus-20240229*) (Anthropic, 2023), (2) Open-weights models, including Llama 3 (*Meta-Llama-3-8B-Instruct*) (Grattafiori and the Llama 3 Team, 2024; AI@Meta, 2024), Llama 3.1 (*Llama-3.1-8B-Instruct*) (Int) and Gemma 2 (*gemma-2-9b-it*) (Team, 2024). The latter models are obtained from HuggingFace, and quantized to 4-bits (Dettmers et al., 2023) to fit our GPU memory. For each input, we obtain 3 outputs from each model using a different random seed. We computed the following metrics to summarize model behavior and measure performance.

- Percentage of times the model generated output that contains a proof and where the model did not utilize a hypothesis, axiom, or rule of inference other than those provided. We consider this a measure of the model compliance with our instruction (Compliance).
- Percentage of steps from the gold standard proof that appear in the generated proof, i.e., to what extent the model used the needed axioms and *modus ponens* (Extensiveness).

Model	Ctx.	Compliance →	Compliance ♠	Extensiveness →	Extensiveness ♠	Correctness →	Correctness ♠	Flow →	Flow ♠	Clarity →	Clarity ♠
GPT-3.5-turbo	SC	66.67%	**83.33%**	**62.38%**	49.52%	**30.00%**	6.67%	**33.33%**	20.00%	**2.67**	2.33
	FC	50.00%	**100%**	**36.19%**	30.48%	**10.00%**	6.67%	**26.67%**	16.67%	**3.00**	2.60
Claude 3 Opus	SC	**100%**	**100%**	**95.24%**	93.14%	**86.67%**	72.00%	**93.33%**	84.00%	**3.83**	3.40
	FC	**100%**	83.33%	**85.71%**	75.71%	**66.67%**	33.33%	**80.00%**	50.00%	**3.67**	3.00
Gemma 2 (9B)	SC	0.00%	0.00%	**11.90%**	0.00%	0.00%	0.00%	3.33%	0.00%	**4.00**	-
	FC	0.00%	0.00%	**16.67%**	0.00%	0.00%	0.00%	3.33%	0.00%	**4.00**	-
Llama 3 (8B)	SC	50.00%	**66.67%**	**49.52%**	41.43%	**3.33%**	**3.33%**	**16.67%**	6.67%	**1.83**	1.80
	FC	33.33%	**83.33%**	**35.71%**	21.90%	**10.00%**	3.33%	**13.33%**	0.00%	**2.17**	1.60
Llama 3.1 (8B)	SC	33.33%	**100%**	**42.38%**	40.95%	0.00%	**3.33%**	**6.67%**	**6.67%**	**1.17**	**1.17**
	FC	66.67%	**50.00%**	**32.38%**	24.29%	**3.33%**	0.00%	**6.67%**	**6.67%**	**2.00**	1.50

Table 1: Summary of the results of our human evaluation study, where Ctx. is short for context. Bold numbers indicate the best score for each pair of (→, ♠) prompts for a given model, and we underline the best score across the FC and SC scenarios for each model.

- Percentage of steps that appear in the generated proof and were correctly applied. In other words, the steps that are correct (Correctness).
- Percentage of steps that appear in the output providing an expression which is a step towards finalizing the proof, even if it was not correctly deduced (Flow).
- Average clarity score reported for the overall text output from a given model (Clarity).

Table 1 summarizes the results of our evaluation. With the exception of Compliance, all metrics show a similar trend of decrease in model performance when the questions are perturbed with ♠. The same is also true when we compare the Full Context to the Selected Context. Most interestingly, the score for Flow was always higher than the Correctness, indicating that models tend to recall or retrieve parts of the right answer from memory, even if they cannot follow the correct sequence of logical steps.

We also observe that not only Compliance did not follow the decrease trend seen for other metrics, but also that most models showed an increase in this metric when prompted with the arbitrary symbol. In fact, this variation in Compliance appears to strongly correlate with our ♠-based replacement scenario. Overall, the average compliance with the original symbol (→) is 43.85%, which compares unfavorably against 71.89% for our replacement scenario (♠). We note that such a significant gap is not evident when looking at performance differences due to variations in context, where we observe an average Compliance of 60.22% for SC and 63.39% for FC. Since invoking facts outside of the givens in the prompt is a big factor in how we compute the Compliance criteria, we believe this counterintuitive increase in value can be at least partially explained by the changes in the semantics induced by our symbol replacement. Intuitively, the semantic similarity between these expressions and the examples seen during training should be lower, making it more unlikely for the model to retrieve relevant content seen during training, even if not useful for the proof.

The model with the best performance across all measures was Claude 3 Opus. Like for most models, its Extensiveness measure was subject to a decrease when one compares the easiest scenario (SC with →) to the hardest scenario (FC with ♠), but its lowest value (75.71%) was still higher than the highest Extensiveness measure of any other model. However, it did experience a sharp decrease in both Correctness (from 86.67% to 33.33%) and Flow (from 93.33% to 50.00%), with the first being the largest difference in performance of any metric for any model. As table 1 shows, there is also a big difference in performance between the API-based LLMs and the Open-weights models. The Gemma 2 model refused to provide a proof in most cases, as it can be seeing by its Compliance measure, making it impossible to draw conclusions about its abilities. We think these results suggests open-weights models are significantly behind APIs, which is well-aligned with performance measured in popular automatic benchmarks.

To delve further into the data, we split the Clarity into intervals, as seen in Table 3, and looked at the other metrics' behavior in each range (here we excluded the cases in which the model did not pro-

Model	Metric	Scenario Δ	
		→ to ♠	SC to FC
GPT-3.5-turbo	Compliance	33.33%	0.00%
	Extensiveness	-9.29%	-22.62%
	Correctness	-13.33%	-10.00%
	Flow	-11.67%	-5.00%
	ExtraHyp	-25.00%	8.33%
	ExtraAxiom	0.00%	83.33%
	ExtraROI	-41.67%	8.33%
	Clarity	-0.38	0.32
Claude 3 Opus	Compliance	-9.09%	-8.33%
	Extensiveness	-6.84%	-13.57%
	Correctness	-25.76%	-30.00%
	Flow	-21.21%	-24.09%
	ExtraHyp	0.00%	0.00%
	ExtraAxiom	27.27%	25.00%
	ExtraROI	0.00%	0.00%
	Clarity	-0.57	-0.3
Gemma 2 (9B)	Compliance	0.00%	0.00%
	Extensiveness	-14.29%	2.38%
	Correctness	0.00%	0.00%
	Flow	-3.33%	0.00%
	ExtraHyp	-41.67%	8.33%
	ExtraAxiom	0.00%	0.00%
	ExtraROI	-8.33%	-8.33%
	Clarity	-4	0
Llama 3 (8B)	Compliance	33.33%	0.00%
	Extensiveness	-10.95%	-16.67%
	Correctness	-3.33%	3.33%
	Flow	-11.67%	-5.00%
	ExtraHyp	-25.00%	8.33%
	ExtraAxiom	-16.67%	83.33%
	ExtraROI	-33.33%	0.00%
	Clarity	-0.3	0.09
Llama 3.1 (8B)	Compliance	25.00%	-8.33%
	Extensiveness	-4.76%	-13.33%
	Correctness	0.00%	0.00%
	Flow	0.00%	0.00%
	ExtraHyp	-25.00%	-8.33%
	ExtraAxiom	16.67%	83.33%
	ExtraROI	-8.33%	8.33%
	Clarity	-0.25	0.58

Table 2: Difference in model performance, measured by our introduced metrics, when the model is presented with the perturbed input versus the original symbol (denoted as "→ to ♠"), and when the model is presented with the Selected Context versus the Full Context (denoted as "SC to FC"). Values in green denote an increase in performance and values in red denote a decrease in performance.

vide a proof). We can see that models like Claude 3 Opus and Gemma 2 were more consistent in the clarity of their outputs. Although we cannot guarantee that the gold standard proof is the only possible proof, we can see from Table 3 that Correctness and Flow align well with the clarity score, i.e., they all decrease as the clarity decreases. This indicates that as the model struggles to complete the proof, the output becomes more convoluted and difficult to understand.

Model	Clar.	Ext.	Corr.	Flow
GPT-3.5-turbo	0∼1	60.00%	60.00%	0.00%
	1∼2	57.14%	0.00%	0.00%
	2∼3	32.00%	0.00%	8.00%
	3∼4	49.64%	16.25%	33.75%
Claude 3 Opus	2∼3	60.00%	20.00%	20.00%
	3∼4	88.44%	66.36%	79.09%
Gemma 2 (9B)	3∼4	34.29%	0.00%	8.00%
Llama 3 (8B)	1∼2	41.43%	3.33%	0.00%
	2∼3	33.85%	4.62%	7.69%
	3∼4	67.62%	13.33%	40.00%
Llama 3.1 (8B)	0∼1	20.00%	0.00%	5.00%
	1∼2	37.50%	0.00%	2.50%
	2∼3	42.54%	4.44%	13.33%
	3∼4	25.71%	0.00%	0.00%

Table 3: Summary of our results grouping by Clarity value bins, where Clar., Ext. and Corr. are short for Clarity, Extensiveness and Correctness, respectively.

To better understand the impact of the different scenarios on models, we compute the difference in model performance when the model is presented with the perturbed input versus the original symbol and when the model is presented with the Selected Context versus the Full Context. For this study, in addition to the metrics introduced before, we also compute:

- The percentage of times the model uses a hypothesis that was not necessary (ExtraHyp), as per our proof of reference.
- The percentage of times the model uses an axiom that was not necessary (ExtraAxiom), as per our proof of reference.
- The percentage of times the model uses a rule of inference that was not provided in the input (ExtraROI).

Table 2 summarizes these findings. The introduction of a perturbation in the input had a bigger impact on Clarity than the change in context for all models. Looking at "→ to ♠" for both Llama models and ChatGPT, → has a higher percentage of steps, but also more unnecessary steps. The symbol replacement seems to be curbing all types of steps. For ChatGPT, we noticed that there is a bigger drop in Extensiveness for "SC to FC" than "→ to ♠", but the drop in Correctness and Flow is smaller for "SC to FC" than "→ to ♠". This suggests that ChatGPT struggles to retrieve the necessary axioms from the givens, but the change in context does not seem to significantly impact its Correctness. For

Claude, both changes seem equally challenging, with the change in context having slightly bigger impact on Extensiveness, Correctness, and Flow. Claude 3 Opus was the only model to have a decrease across all metrics for both differences.

There are three different types of steps that could appear in our studied proof: Hypotheses, Axioms, and Rules of Inference. With that in mind, we split each of our metrics into these three cases and summarize the results on Table 4. We noticed that models tend to be better at manipulating axioms than at listing hypotheses or applying rules of inference. In particular, both Llama models struggled with rules of inference (ROIs), with no Extensiveness value being higher than 25% and all Correctness values being 0%. For these two models we also noticed that while mistakes on substitution (Correctness) seem to lead to fewer contributions to the final proof (low Flow), mistakes on rules of inference still lead to contributions. We think this indicates that these models use rule of inference to justify steps they "know" are needed, even though the rule never correctly justifies said step.

We also observed that Gemma 2 never attempted to use an axiom (all values are 0%), and its best performance was on Extensiveness for hypotheses. ChatGPT had the biggest gap on Extensiveness for axioms (SC vs FC), and on the Full Context scenario, it always made mistakes on substitution. Claude 3 Opus was slightly worse at applying ROIs than making substitutions on easiest scenario (SC with $\rightarrow$), but it did much worse at substitution than on ROIs on the hardest scenario (FC with ♠). Its only use of unnecessary steps was on the hardest scenario and it used extra axioms.

We also analyze the spans highlighted by the annotators, alongside the comments left by them on each location. We collected a total of 304 annotations derived from highlighted spans, out of which, 170 were unique. We note that more than 99% of these comments denote errors made my the model, often related to the questions that were presented to the annotators. Based on this insight, we use this information to empirically estimate "when" models tend to make their first mistake as they generate the requested proof. Concretely, we compute the position in terms of number of characters, normalized by answer length, of the start of the spans highlighted by the annotator. These relative positions are then averaged for all the answers for each model.

As shown in Table 5, we see models tend to make their first mistake relatively early in their generation, which is true even for the more advanced black-box models. The table also shows how differently each model behaves in terms of verbosity, where we see models Llama 3 and 3.1 generating answers that are up to 3,000 characters long, while API-based models like GPT-3.5 and Claude 3 Opus are significantly more concise. The relatively lower value shown by Gemma is due to the model often not generating an actual proof, which artificially brings this number down. Compared to our gold standard proofs, which contain 478 and 452 characters, these results show that models tend to favor verbosity instead of precision when generating these kinds of proofs without specialized prompts.

Finally, we analyze the *content* of the annotations for each of the highlighted spans. We think these may provide additional insights on the nature of the mistakes by the models. In order to perform this analysis, we encode the annotations with Sentence-BERT (Reimers and Gurevych, 2019), via the *all-MiniLM-L6-v2* model, using the SentenceTransformers[1] package. We then perform clustering using k-means via its scikit-learn implementation (Pedregosa et al., 2011). We compute clusters varying parameter k, the number of clusters, with $k = 1, \ldots, 15$, and select the top 3 results as based on silhouette scores (Rousseeuw, 1987). Finally, we visually analyze these best results by performing PCA on the embedded annotation comments, and plotting them on a 2D chart, coloring the examples by cluster. We find the that $k = 7$ offers meaningful results, highlighting four distinct behaviors models often engage in the following cluster, which here we represent by the instance closest to the centroid: (Cluster 0) "*Inability to recognize the Unicode symbol used to replace the implication operator*", (Cluster 1) "*Does not follow from Axiom*", (Cluster 3) "*Substitution Issue*", (Cluster 5) "Incorrect usage of *modus ponens*".

Discussion It is known that if we have a theorem in propositional calculus, then it has arbitrarily many proofs. One cannot even guarantee there is a unique "minimal length" proof. Hence, one may question our comparison of model outputs against a fix proof. However, since we study changes, we need a baselines to compare against. This also makes it easier to evaluate the output systematically and consistently. We also note that if we

[1] github.com/UKPLab/sentence-transformers

Model	Cxt.	Hypotheses				Axioms								Rules of Inference							
		Ext.		ExtraHyp		Ext.		Corr.		Flow		ExtraAxiom		Ext.		Corr.		Flow		ExtraROI	
		→	♠	→	♠	→	♠	→	♠	→	♠	→	♠	→	♠	→	♠	→	♠	→	♠
GPT-3.5-turbo	SC	83%	67%	17%	17%	72%	69%	42%	14%	25%	14%	17%	0%	42%	28%	17%	0%	42%	28%	33%	0%
	FC	67%	100%	50%	0%	22%	6%	0%	0%	8%	0%	83%	100%	44%	31%	17%	11%	44%	31%	50%	0%
Claude 3 Opus	SC	100%	100%	0%	0%	92%	100%	92%	73%	92%	80%	0%	0%	94%	83%	83%	70%	94%	83%	0%	0%
	FC	100%	100%	0%	0%	75%	69%	44%	17%	75%	36%	0%	50%	83%	67%	83%	42%	83%	67%	0%	0%
Gemma 2 (9B)	SC	67%	0%	33%	0%	0%	0%	0%	0%	0%	0%	0%	0%	6%	0%	0%	0%	6%	0%	17%	0%
	FC	100%	0%	50%	0%	0%	0%	0%	0%	0%	0%	0%	0%	6%	0%	0%	0%	6%	0%	0%	0%
Llama 3 (8B)	SC	67%	33%	17%	0%	67%	83%	8%	8%	17%	8%	0%	0%	25%	6%	0%	0%	17%	6%	33%	17%
	FC	50%	33%	33%	0%	56%	42%	19%	8%	14%	0%	100%	67%	14%	0%	0%	0%	14%	0%	50%	0%
Llama 3.1 (8B)	SC	67%	0%	50%	0%	64%	86%	0%	8%	8%	6%	0%	0%	6%	6%	0%	0%	6%	6%	33%	0%
	FC	100%	0%	17%	17%	19%	50%	6%	0%	0%	0%	67%	100%	22%	11%	0%	0%	14%	11%	17%	33%

Table 4: Results disaggregated for different categories: Hypotheses, Axioms, and Rules of Inference. For the sake of presentation, values in this table are rounded to the closest integer.

Model	Answer Len.	Rel. Ann. Loc.
GPT-3.5 turbo	1,310 ± 500	15.573 % (10.34)
Claude 3 Opus	1,577 ± 594	14.126 % (10.18)
Gemma 2 (9B)	890 ± 43	28.247 % (11.90)
LLama 3 (8B)	2,181 ± 751	17.549 % (09.70)
Llama 3.1 (8B)	3,747 ± 5,024	33.592 % (21.69)

Table 5: Average length of model answers (**Answer Len.**), in characters, with their respective standard deviations, and average relative location of the comments left by the annotators (denoted as **Rel. Ann. Loc.**), normalized be answer length. For the latter, numbers in parenthesis show the standard deviation.

Figure 3: Clusters of annotations related to model behavior recognized as a mistake by our annotators.

had access to the training data, we could ensure we use the same proof used during training as our gold standard, but this is difficult or impossible in practice. To try to minimize this issue, we employ mathematicians to evaluate, which allow us to also collect comments about any conflict that could arise from this assumption. Finally, we would like to note that, ultimately, the difficulty of the model to follow the steps of the gold standard correlates with the decrease of overall clarity of the output.

5 Conclusions

This paper studies the generalization capabilities of LLM through the lens of mathematical reasoning. We perform an in-depth human evaluation of the output of LLMs when they are prompted to produce basic proofs in propositional calculus, comparing their answers when we replace the implication operator (→) with an unrelated, arbitrary symbol (♠). Our results show that nearly all our tested models produce lower quality proofs in this test, in particular open-weights models, suggesting the abilities of these LLMs to reason in this context have important limitations. For future work we would like to extend this study to incorporate more proofs, models, and multiple annotators. We would also like to analyze how models react to other input perturbations, for example using other replacement symbols, and/or alternative representations for them.

Limitations

While our study may provide valuable insights into the mathematical reasoning abilities of large language models, it is subject to several limitations. First, our analysis is constrained to a finite and small set of tasks, which do not capture the full breadth of mathematical reasoning scenarios. Second, human evaluation, while essential for assessing nuanced reasoning steps, is inherently subjective and may introduce variability in judgments. We would like to improve on this in future work by working with multiple annotators. Third, the models examined represent a snapshot of current architectures and training paradigms. Finally, our study focuses on English-language prompts leaving open questions about performance across languages.

References

Introducing Llama 3.1: Our most capable models to date. https://ai.meta.com/blog/meta-llama-3-1/.

AI@Meta. 2024. Llama 3 model card.

Miltiadis Allamanis, Pankajan Chanthirasegaran, Pushmeet Kohli, and Charles Sutton. 2017. Learning Continuous Semantic Representations of Symbolic Expressions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 80–88. PMLR.

Anthropic. 2023. Introducing Claude. Anthropic.

Daniel G Bobrow. 1964. Natural language input for a computer problem solving system. *AI Technical Reports*.

Greg Brockman, Atty Eleti, Elie Georges, Joanne Jang, Logan Kilpatrick, Rachel Lim, Luke Miller, and Michelle Pokrass. 2023. Introducing ChatGPT and Whisper APIs. OpenAI Blog.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc.

Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. 2018. Can Neural Networks Understand Logical Entailment? In *International Conference on Learning Representations*.

Edward A Feigenbaum and Julian Feldman. 1963. Computers and thought.

Simon Frieder, Martin Trimmel, Rashid Alawadhi, and Klaus Gy. 2023. LLM vs ITP. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*.

Aaron Grattafiori and the Llama 3 Team. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

John Harrison, Josef Urban, and Freek Wiedijk. 2014. History of Interactive Theorem Proving. In *Handbook of the History of Logic*, volume 9, pages 135–214. Elsevier.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. *Preprint*, arxiv:2311.12022.

Dino Rossegger. 2019. *Introduction to Mathematical Logic*.

Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Gemma Team. 2024. Gemma 2: Improving Open Language Models at a Practical Size. *Preprint*, arXiv:2408.00118.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021. Naturalproofs: Mathematical theorem proving in natural language. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. 2022. Naturalprover: Grounded mathematical proof generation with language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Chase B. Wrenn. 2025. Naturalistic Epistemology: Internet Encyclopedia of Philosophy.

Synthetic Proofs with Tool-Integrated Reasoning: Contrastive Alignment for LLM Mathematics with Lean

Mark Obozov
Research Center of the Artificial Intelligence Institute
Innopolis University
Innopolis, Russia
obozovmark9@gmail.com

Michael Diskin
HSE University
Moscow, Russia
michael.s.diskin@gmail.com

Aleksandr Beznosikov
Innopolis University
Innopolis, Russia

Alexander Gasnikov
Innopolis University
Innopolis, Russia

Serguei Barannikov
Skoltech, CNRS
Moscow, Russia

Abstract

Modern mathematical reasoning benchmarks primarily focus on answer finding rather than proof verification, creating a gap in evaluating the proving capabilities of large language models (LLMs). We present a methodology for generating diverse mathematical proof tasks using formal tools. Our approach combines Lean-based synthetic problem generation with a Tool-Integrated Reasoning (TiR) framework for partial (sampling-based) proof validation, and it uses contrastive preference optimization to align the model's proof outputs. Experiments on the Qwen-2.5 family of models demonstrate meaningful improvements in mathematical reasoning, particularly for smaller models. Our aligned models achieve up to a 57% higher success rate than baselines on the MiniF2F benchmark (across 0.5B, 1.5B, and 7B parameter models). These results highlight the potential of synthetic data and integrated validation for advancing LLM-based mathematical reasoning.

1 Introduction

Mathematical reasoning is a fundamental challenge for artificial intelligence. Despite significant advances in large language models (Li et al., 2024; Yang et al., 2024a; DeepSeek-AI, 2024), achieving robust proof-solving capabilities comparable to human mathematicians remains elusive. A core difficulty lies in the vast search space of proofs: any given statement can spawn an enormous graph of conjectures and implications. Exhaustive search in this space is infeasible — formal proof attempts like AlphaProof (Hubert et al., 2024) *can be* computationally expensive at scale for complex problems.

Recent approaches using LLMs show promise in their flexibility and reasoning ability. However, even models that excel at answer-focused tasks (e.g., the MATH benchmark (Hendrycks et al., 2021)) often struggle with formal proofs and higher-level mathematical reasoning (Tsoukalas et al., 2024; Glazer et al., 2024). This gap calls for methodologies that combine the adaptability of LLMs with the rigor of structured verification.

The contributions of this paper are as follows:

- **Tree-Based Conjecture Generation**: A tree-based algorithm for synthetic conjecture generation using Lean (de Moura et al., 2015) to produce diverse, valid mathematical problems.
- **Tool-Integrated Reasoning**: A Tool-Integrated Reasoning (TiR) framework for partial proof validation that does not require full formalization of the proofs.
- **Contrastive Alignment of Proofs**: A contrastive preference optimization approach (SimPO) to align the model's proof generation with preferred (correct) solutions.
- **New Proof Benchmark Dataset**: A synthetic dataset of 30,000 generated problems and proofs for benchmarking mathematical reasoning in LLMs.

In our pipeline, **Lean** primarily serves as a *generation and structuring environment*: it provides statements, neighborhood relations, and hints that scaffold synthesis. Model outputs are *informal proofs*. We rely on TiR and an LLM judge as *probabilistic* filters at scale; comprehensive formal checking in Lean is used only in a limited way here and is left to future work.

Our experiments with Qwen-2.5 models (Yang et al., 2024b) (0.5B–7B parameters) demonstrate significant improvements in mathematical reasoning, particularly for smaller models. These results highlight the potential of combining structured generation with flexible validation to advance mathematical reasoning in AI systems.

Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025), pages 195–202
November 9, 2025

2 Background

Formal theorem proving has traditionally relied on search-based methods in proof assistants. Tools like Lean's Aesop tactic (Limperg and From, 2023) and the HyperTree proof search algorithm (Poesia et al., 2024a) can systematically explore proofs but struggle with the combinatorial explosion of possibilities in complex problems. More recently, large language models have been applied to theorem proving (Li et al., 2024; Yang et al., 2024a), but purely data-driven approaches still face challenges unless guided by formal structure. Synthetic data generation has shown promise: for instance, AlphaGeometry (Trinh et al., 2024) trained an LLM on two million procedurally generated geometry problems, yielding impressive problem-solving performance.

Aligning LLM outputs with desired solutions often requires learning from preferences. Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) showed that human feedback can effectively steer model behavior, and Direct Preference Optimization (DPO) (Rafailov et al., 2023) achieved this without a separate reward model. Recent contrastive approaches led to the Simple Preference Optimization (SimPO) (Meng et al., 2024) framework, which introduces a margin γ to ensure the model scores a preferred output higher than a dispreferred one by at least a fixed gap. This margin-based criterion is well-suited for mathematical proofs, where distinguishing correct reasoning from subtly flawed reasoning is essential; alternative preference objectives and recipes include KTO (Ethayarajh et al., 2024), LMSI (Huang et al., 2022), ORPO (Hong et al., 2024), and PPO (Schulman et al., 2017). Another complementary direction is integrating external tools into the reasoning loop. Tool-Integrated Reasoning (Gou et al., 2023) agents allow an LLM to call formal solvers or checkers during problem solving. We build on this idea by using formal environment checks to partially verify generated conjectures and proofs without requiring complete formalization.

Self-improving systems represent another promising direction, with recent work focusing on agents that enhance their mathematical capabilities by autonomously generating and filtering their own proofs (Lin et al., 2024; Huang et al., 2022; Poesia et al., 2024b), complemented by intrinsic motivation algorithms like Minimo (Poesia et al., 2024b) for exploring infinite action spaces without predefined goals.

Current evaluation methodologies present significant limitations; standard benchmarks (Liu et al., 2024; Hendrycks et al., 2021) typically rely on answer-matching approaches that fail to validate reasoning steps, while formal proof benchmarks such as PutnamBench and MiniF2F (Tsoukalas et al., 2024; Zheng et al., 2021) offer more rigorous assessment environments with varying difficulty distributions — PutnamBench featuring exceptionally challenging problems and MiniF2F providing a broader range suitable for evaluating models across different skill levels — though they still require bridging the gap between natural language reasoning and formal representations.

3 Synthetic Generation

3.1 Lean-Based Generation

We propose a tree-based algorithm to generate new conjectures from existing ones. The method treats mathematical statements as nodes in a proof graph and leverages known proofs to create harder related problems. Given an initial conjecture X_0 with a known proof Y, we perform a random walk of N steps starting at X_0, moving to a sequence of neighboring conjectures in the graph. This produces a trajectory of intermediate conjectures that lie progressively further from the root (axioms). We then reverse this trajectory and prepend the original proof Y, using X_0 as a lemma to derive a new conjecture. In essence, the algorithm finds a new target statement that is "beyond" X_0 in the proof tree and constructs a valid proof for it using Y. Algorithm 1 outlines the procedure.

The choice of neighbor selection in Algorithm 1 is crucial. We consider several strategies, including Lean hints and tooling (e.g., *Lean Copilot* (Song et al., 2024)), a formal environment (Peano (Poesia et al., 2024b)), and lightweight LLM-based heuristics such as symbol-overlap and predicted difficulty; the walk stops on a fixed budget or when predicted difficulty saturates.

3.2 Generating New Problems from Solutions

Another strategy for synthesis is to derive new problems from known solutions. Given a solved problem P_0 with solution S_0, we attempt to *invert* it: generate a new problem P_1 for which S_0 (or a minimally modified version) serves as a solution. This inverse-problem technique expands

Algorithm 1 Synthetic Conjecture Generation

Require: $T = (V, E)$, N, X_0, Y ▷ Proof graph, steps, initial conjecture, and its proof
Ensure: $X_0 \in V$ and $Y \subseteq V$ ▷ X_0 and all proof nodes in V

$X_{\text{current}} \leftarrow X_0$
$T_{\text{trace}} \leftarrow \{\}$ ▷ Initialize trace
for $i = 1$ to N **do**
  $X_{\text{next}} \leftarrow$ choose a neighbor of X_{current} ▷ $X_{\text{next}} \in N(X_{\text{current}})$
  $T_{\text{trace}} \leftarrow T_{\text{trace}} \cup \{X_{\text{next}}\}$
  $X_{\text{current}} \leftarrow X_{\text{next}}$
end for
reverse T_{trace}
$T_{\text{trace}} \leftarrow Y \cup T_{\text{trace}}$ ▷ Prepend original proof Y

our dataset with challenging new problems while retaining a valid solution path.

Illustration. If S_0 proves $u + v \geq 2\sqrt{uv}$ for $u, v \geq 0$, then substituting $u = a^2$, $v = b^2$ and adding a constraint (e.g., $a+b = 1$) yields a variant inequality where the same proof skeleton applies with adjusted premises.

3.3 Generating Rejected Values

For contrastive training, we need not only correct proofs but also plausible incorrect proofs as counterexamples. We employ three techniques to automatically generate such *rejected* outputs: (1) use a smaller or less capable LLM to answer the problem (weaker models often produce incorrect or suboptimal solutions); (2) if the main model produces a correct solution, prompt it to introduce a mistake into its reasoning (and if it fails to solve the problem, its failed attempt is used as R); (3) perturb the proof trajectory from our tree-based generator (Algorithm 1) to create a flawed solution path. These methods yield contrastive pairs where y_w is a correct proof and y_l is a similar but incorrect attempt.

3.4 Tool-Integrated Validation

We integrate a validation step using external tools to filter the generated data. Instead of asking the LLM to produce a fully formal proof, we prompt it to output a verification function f (e.g., a Python snippet) that returns 1 if a conjecture holds for given inputs and 0 otherwise. We then evaluate f on N randomly sampled inputs from a domain-specific environment to statistically test the conjecture; this is a *sampling-based* check and thus provides no absolute soundness guarantee. Algorithm 2 illustrates this process, which computes the fraction of inputs for which f returns true.

Algorithm 2 TiR-Based Validation

Require: $f : \{\dots\} \to \{1, 0\}$
Require: N ▷ Number of random trials
Require: $\mathcal{E}$ ▷ Input generator environment
Ensure: Empirical success rate of conjecture (fraction of inputs where f returns 1)

$success \leftarrow 0$
$failure \leftarrow 0$
for $i = 1$ to N **do**
  $x \leftarrow \mathcal{E}$.generate_input()
  $output \leftarrow f(x)$
  if $output = 1$ **then**
    $success \leftarrow success + 1$
  else
    $failure \leftarrow failure + 1$
  end if
end for
return $\frac{success}{N}$

If $f(x) = 1$ for all N sampled inputs (i.e., a success rate of 1.0), we consider the conjecture *validated for the sampled domain* $\mathcal{E}$; adversarial counterexamples may still exist (see Limitations). For example, given the inequality conjecture $a^2 + b^2 + c^2 \geq ab + ac + bc$, our TiR prompt produces a function $f(a, b, c)$ that computes $a^2 + b^2 + c^2$ and $ab + ac + bc$ and returns 1 if the inequality holds (0 otherwise). Running Algorithm 2 on this f with random numeric inputs quickly confirms the truth of the conjecture.

4 Training Framework

Using the synthetic data and validation techniques above, we construct training pairs for contrastive alignment. For each problem x, we obtain a correct proof y_w (verified by TiR) and a corresponding incorrect proof y_l (generated via the strategies in Section 3.3). We then fine-tune the model using a SimPO-based objective, which encourages the model to assign higher probability to y_w over y_l.

The training loss follows the SimPO formulation:

$$\mathcal{L}_{\text{SimPO}}(\pi_\theta) = \\ -\mathbb{E}_{(x,y_w,y_l)}\Big[\log\sigma\Big(\log\frac{\beta}{|y_w|}\log\pi_\theta(y_w|x) - \\ \log\frac{\beta}{|y_l|}\log\pi_\theta(y_l|x) - \gamma\Big)\Big], \quad (1)$$

where $|y|$ denotes the length (number of tokens) of output y. We set $\beta = 2.0$ (length normalization against overly short outputs) and $\gamma = 0.5$ (margin between preferred and rejected scores) based on validation experiments. Here β provides a length-normalization factor for the log-probabilities (penalizing overly short answers), and γ enforces a minimum margin between the model's scores for the preferred and rejected outputs.

5 Experiments

5.1 Setup

We fine-tuned three Qwen-2.5 models (with 0.5B, 1.5B, and 7B parameters) on our synthetic proof dataset. The SimPO objective (using $\beta = 2.0$, $\gamma = 0.5$) was implemented with a modified `torchtune` library (torchtune maintainers and contributors, 2024). Each model was trained for 3 epochs on two A100 80GB GPUs, and we selected the best checkpoint based on validation performance. Gradient checkpointing and offloading were used to manage GPU memory during training.

For the synthetic data generation, we employed both Lean and pure-LLM environments. We also experimented with the Peano formal environment, but it produced low-quality conjectures with our algorithm and was excluded from the final dataset. Our TiR validation framework was implemented to handle a variety of mathematical structures (e.g., arithmetic, algebra, group theory, graph theory) and was used primarily to filter out invalid conjectures or proofs before they were added to the training data.

We evaluated our aligned models on the MiniF2F test benchmark, which consists of formal math problems disjoint from those used in training. Since our model outputs are informal proofs (not directly checkable by an automatic theorem prover), we employed a separate judge model to assess solution correctness. Specifically, we used a 32B distilled model (DeepSeek-R1-32B-Qwen-Distilled) as an automated verifier: given a problem and our model's solution, it decides whether the solution is correct (binary accept/reject). We report the *judge-accepted pass rate* in $[0, 1]$ averaged over MiniF2F and compare it to the base model's success rate (baseline).

Model	Size	Aligned	Baseline
Qwen-2.5	0.5B	**0.22**	0.14
Qwen-2.5	1.5B	**0.37**	0.29
Qwen-2.5	7B	**0.53**	0.47

Table 1: MiniF2F *judge-accepted pass rate* (0–1). "Aligned": SimPO-trained on our synthetic pairs; "Baseline": original pretrained model. Higher is better. Verifier: DeepSeek-R1-32B-Qwen-Distilled. *Scope:* single family (Qwen-2.5).

5.2 Results and Discussion

Our alignment approach markedly improved the models' problem-solving success rates. Table 1 summarizes each model's performance versus its unaligned baseline.

The results reveal several insights:

- **Consistent Gains**: Across all model sizes, our aligned models outperform their baselines. The smallest model (0.5B) enjoys the largest relative gain (about +57% relative improvement).
- **Scaling Effects**: The benefit of synthetic training persists as model size grows, though the relative improvement is more pronounced for smaller models. This suggests that smaller models gain proportionally more from our additional training data and alignment.
- **Validation Efficacy**: The TiR filtering appears effective in removing invalid proofs from the training corpus, which helps ensure the model learns from mostly correct and verifiable examples.

6 Scope and External Validity

Our empirical scope is intentionally narrow (Qwen-2.5, MiniF2F) to isolate pipeline effects. The components of our method (problem synthesis, TiR filtering, contrastive alignment) are model-agnostic; we expect transfer across families with two practical considerations: (i) evaluation — complementing the LLM judge with selective formal checking on representative subsets; (ii) domain coverage — extending $\mathcal{E}$ and validators beyond algebra/number theory (e.g., geometry and graphs require domain-specific generators and metamorphic tests).

7 Conclusion

Our study demonstrates that combining synthetic proof generation with partial formal validation can substantially bolster an LLM's mathematical reasoning abilities. In particular, training on generated conjecture–proof pairs (with contrastive alignment) enabled even relatively small models to solve significantly more formal problems. This work represents a step toward AI systems that can not only generate and verify mathematical proofs but also gradually improve their own reasoning strategies.

To support reproducibility and accelerate progress in mathematical reasoning research, we will release our complete synthetic dataset of 30,000 problems and the Tool-Integrated Reasoning framework under open-source licenses. This includes the tree-based generation algorithms, validation mechanisms, and alignment methodology described in this paper.

In future work, we plan to enhance the synthetic generation process with chain-of-thought prompting to further improve conjecture quality, extend our framework to other domains such as geometry (which may require specialized validation techniques), strengthen the integration between LLM reasoning and formal verification (improving the TiR framework), and investigate more efficient training strategies to scale to larger models.

8 Limitations

1. **Sampling-based validation.** TiR offers high-precision but only *sampling-based* guarantees: a conjecture that passes N random trials in $\mathcal{E}$ may still be false (adversarial cases are possible).

2. **Domain coverage.** The synthetic set is skewed toward algebra/number theory; geometry, analysis, and combinatorics require domain-specific generators and validators that we do not cover here.

3. **LLM judge bias.** Evaluation relies on an automated LLM judge (binary accept/reject). Despite high spot-check agreement, residual bias may affect absolute scores.

4. **Empirical scope.** Experiments intentionally target one model family (Qwen-2.5) and one benchmark (MiniF2F) to isolate pipeline effects; cross-family/benchmark validation is left for follow-up work.

9 Acknowledgements

The study was supported by the Ministry of Economic Development of the Russian Federation (agreement No. 139-10-2025-034 dd. 19.06.2025, IGK 000000C313925P4D0002)

References

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The Lean theorem prover (system description). In *Automated Deduction – CADE-25*, volume 9195 of *Lecture Notes in Computer Science*, pages 378–388. Springer.

DeepSeek-AI. 2024. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, and et al. 2024. FrontierMath: A benchmark for evaluating advanced mathematical reasoning in AI. *arXiv preprint arXiv:2411.04872*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, and et al. 2023. ToRA: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, and et al. 2021. Measuring mathematical problem solving with the MATH dataset. In *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and et al. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Thomas Hubert, Rishi Mehta, Laurent Sartran, and Google DeepMind Team. 2024. AI achieves silver-medal standard solving International Mathematical Olympiad problems. *Nature*. News & Views.

Jia Li, Edward Shan, Xuefeng Wu, Yifei Ma, Rohan Murty, Himanshu Arora, and et al. 2024. Numina-Math: AI Mathematical Olympiad Progress Prize.

Jannis Limperg and Asta Halkjær From. 2023. Aesop: White-box best-first proof search for Lean. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '23)*.

Haohan Lin, Zhiqing Sun, Yiming Yang, and Sean Welleck. 2024. Lean-STaR: Learning to interleave thinking and proving. *arXiv preprint arXiv:2407.10040*.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, and et al. 2024. MathBench: Evaluating the theory and application proficiency of LLMs with a hierarchical mathematics benchmark. *arXiv preprint arXiv:2405.12209*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, and et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Gabriel Poesia, David Broman, Nick Haber, and Noah D. Goodman. 2024a. HyperTree proof search for neural theorem proving. *arXiv preprint arXiv:2205.11491*.

Gabriel Poesia, David Broman, Nick Haber, and Noah D. Goodman. 2024b. Learning formal mathematics from intrinsic motivation. *arXiv preprint arXiv:2407.00695*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. 2024. Towards large language models as copilots for theorem proving in Lean. *arXiv preprint arXiv:2404.12534*.

torchtune maintainers and contributors. 2024. torchtune: PyTorch's finetuning library.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. 2024. Solving Olympiad geometry without human demonstrations. *Nature*, 625:476–482.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, and et al. 2024. PutnamBench: Evaluating neural theorem-provers on the Putnam Mathematical Competition. *arXiv preprint arXiv:2407.11214*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, and et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, and et al. 2024b. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2021. miniF2F: A cross-system benchmark for formal Olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*.

A Examples of generated conjectures

Here are a few examples of generated problems with our algorithms.

InternLM step generator was strong in algebra problems generation with algorithm 1.

Basic problem

For $a, b, c \geq 0$ and $a+b+c=1$ prove that $1+12abc \geq 4(ab+bc+ac)$

We got a new problem and proof to it, which further on the G than starting one:

New problem

$1+12ab(1-a-b) \geq 4(ab+b(1-a-b)+(1-a-b)a)$

B GeoGen and AlphaGeometry synthetic

In our methods, we did not touch geometrical problems as they require slightly different approaches. Also, formalization of geometrical problems to Lean-like languages is a quite complicated task. Knowing this, we consider different approach to synthetical geometric task generation. We analyzed AlphaGeometry framework and examined its problems:

1. AlphaGeometry can't solve problems that require non-trivial additional constructions. For instance, median doubling.
2. Directed angles make it possible to solve really generalized problems.
3. Projective theorems (Pascal theorem, Pappus's theorem, etc.), usually can't be solved.
4. There is no numerical package that could help to calculate geometrical problems in coordinates.

Knowing these facts, we used a slightly modified version of GeoGen which included Humpy and Dumpy points and created several configurations to this algorithm. Then, we applied Qwen2.5 to translate problems to AlphaGeometry language. Finally, we searched for the solution with AlphaGeometry. With this algorithm pairs problem/solution might be conducted. Let's consider examples of generated problems:

While we were able to generate some qualitative geometrical problems with this algorithm, this framework is computationally heavy, so we do not highlight any alignment experiments related to the geometrical data.

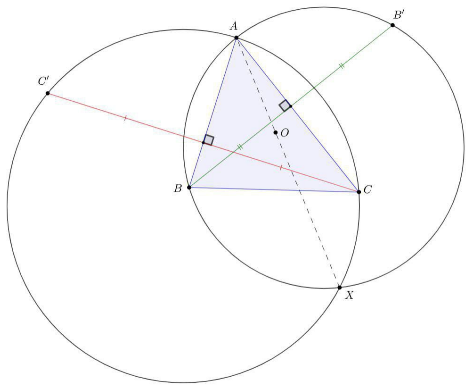


Figure 1: GeoGen example problem 1

C Real effects of small changes in evaluation

We examined the following question: Can we produce new evaluation sets by applying small changes to the questions similar to noise? For instance, given a simple task:

Original problem

On a circle, there are $2n$ points: n red and n blue. A red frog starts on one of the red points, and a blue frog starts on one of the blue points. Each minute, the red frog jumps clockwise to the next red point, and simultaneously, the blue frog jumps counterclockwise to the next blue point. Prove that for any initial positions of the frogs, it is always possible to draw a line such that the frogs are on opposite sides of the line at all times.

Problem (With changes)

On a circle, there are $2n$ points: n red and n blue. A red **turtle** starts on one of the red points, and a blue **turtle** starts on one of the blue points. Each **second**, the red **turtle** **hops** clockwise to the next red point, and simultaneously, the blue **turtle hops** counterclockwise to the next blue point. Prove that for any initial positions of the **turtles**, it is always possible to draw a line such that the **turtles** are on opposite sides of the line at all times.

With recent works there was an active attention to such mechanism as a potential way to produce new evaluation frameworks. In our work we show that such changes are inefficient. Changes of the results from table 1 are bounded within $[-1, +1]$. Therefore, such a method of creating extra evaluation data is not really working and affecting LLM reasoning abilities.

D Examples of TiR validation function

In this section we provide a few examples of functions that were generated during the TiR conjecture validation procedure.

Conjecture 1 **(Inequality)** $a^2 + b^2 + c^2 >= ab + ac + bc$

```
def f(a, b, c):
    """
    Evaluates the conjecture a^2 + b^2 +
        c^2 >= ab + ac + bc.

    Args:
        a, b, c: Real numbers.

    Returns:
        1 if the conjecture holds, 0
            otherwise.
    """
    lhs = a**2 + b**2 + c**2
    rhs = a*b + a*c + b*c
    if lhs >= rhs:
        return 1
    else:
        return 0
```

Conjecture 2 **(Functional equation), IMO 2024** A function $f : \mathbb{Q} \to \mathbb{Q}$ is called aquaesulian if the following property holds: for every $x, y \in \mathbb{Q}$,

$$f(x + f(y)) = f(x) + y \quad \text{or} \quad f(f(x) + y) = x + f(y). \tag{2}$$

Show that there exists an integer c such that for any aquaesulian function f, there are at most c different rational numbers of the form $f(r)+f(-r)$ for some rational number r, and find the smallest possible value of c.

```
from fractions import Fraction

def aquaesulian_validator(x, y):
    """
    A validator for an aquaesulian
        function. This function will
        return 1 if the conjecture
    holds for the given x and y, and 0
        otherwise.

    Args:
        x: Rational number (Fraction)
        y: Rational number (Fraction)

    Returns:
        1 if conjecture holds, 0
            otherwise.
    """
    # Define the function f (this is a
        placeholder; define actual f
        based on the conjecture)
    # For example, assume f(x) = x (
        identity function for simplicity
         in testing)
    def f(z):
        return z  # Identity function

    # Check if f(x + f(y)) = f(x) + y or
         f(f(x) + y) = x + f(y)
    if f(x + f(y)) == f(x) + y:
        return 1  # First condition
            holds
    elif f(f(x) + y) == x + f(y):
        return 1  # Second condition
            holds
    else:
        return 0  # Neither condition
            holds
```

Conjecture 3 (**Graph theory**). There is a node in a planar graph with *degree* less than 5.

```
def f(graph):
    """
    Validator function for the
        conjecture: in a planar graph,
        there exists at least one node
    with degree less than 5.

    Parameters:
        graph (dict): A dictionary
            representing the graph as an
             adjacency list.
                      Each key is a node
                          , and the
                          value is a
                          list of its
                          neighbors.

    Returns:
        1 if there is a node with degree
             less than 5, 0 otherwise.
    """
    # Iterate over each node and check
        its degree
    for node, neighbors in graph.items()
        :
        if len(neighbors) < 5:
            return 1  # Conjecture holds
                : there exists a node
                with degree < 5

    return 0  # Conjecture does not hold
        : no node with degree < 5
```

Author Index