EquivPruner: Boosting Efficiency and Quality in LLM-Based Search via Action Pruning

Anonymous ACL submission

Abstract

Large Language Models (LLMs) excel at complex reasoning through search algorithms, yet current strategies often suffer from massive token consumption due to redundant exploration of semantically equivalent steps. Existing semantic similarity methods struggle to accurately identify such equivalence in domain-specific contexts like mathematical reasoning. To address this, we propose *EquivPruner*, a simple yet effective approach that identifies and prunes semantically equivalent actions during LLM reasoning search. We also introduce MathEquiv, the first dataset we created for mathematical statement equivalence, which enables the training of a lightweight equivalence detector. Extensive experiments across various models and tasks demonstrate that *EquivPruner* significantly reduces token consumption, improving searching efficiency and often bolstering reasoning accuracy. For instance, when applied to Qwen2.5-Math-7B-Instruct on GSM8K, *EquivPruner* reduced token consumption by 48.1% while also improving accuracy. Our code is available at `https://anonymous.4open.science/r/EquivPruner-2364/`.

1 Introduction

Large Language Models (LLMs) are increasingly demonstrating remarkable capabilities, yet their performance scaling during pretraining faces potential constraints due to data limitations (Lightman et al., 2023). Consequently, enhancing LLM capabilities at inference time has become a critical research frontier (Snell et al., 2024). A prominent direction involves leveraging search algorithms, particularly reward-guided tree search, to improve complex reasoning (Ke et al., 2025). These methods typically expand the search space by generating multiple reasoning steps (e.g., via chain-of-thought prompting (Wei et al., 2022)) and employ various techniques to navigate this space, from foundational algorithms like depth-first search (Yao et al., 2023) to more advanced heuristics like beam search (Kang et al., 2024) and Monte Carlo Tree Search (MCTS) (Chen et al., 2024; Zhang et al., 2024a), to identify high-quality solutions.

However, current search strategies exhibit significant inefficiencies (Damani et al., 2024). A common practice involves sampling multiple candidate reasoning steps and exploring them, often allocating computational resources uniformly across these candidates (Yao et al., 2023; Long, 2023; Besta et al., 2024). This approach overlooks the potential semantic equivalence among textually distinct candidates. Treating semantically identical reasoning steps as unique branches leads to redundant exploration of the search space, incurring substantial computational overhead through unnecessary token generation and processing. Moreover, for search algorithms that incorporate preference learning based on intermediate steps (e.g., Xie et al., 2024; Jiang et al., 2024), learning preferences from pairs of equivalent steps may provide noisy or conflicting signals, hindering the learning of effective reasoning policies. This challenge is particularly acute in domains like mathematical reasoning, where numerous textual formulations can represent the same underlying logical operation or state. Addressing this redundancy via standard Semantic Textual Similarity (STS) techniques (Majumder et al., 2016) proves inadequate as illustrated in Figure 1. Existing embedding models, such as SBERT (Reimers and Gurevych, 2019), predominantly trained on general text, often fail to capture the nuanced structural and logical equivalence specific to mathematical statements. Even domain-specific models like MathBERT (Peng et al., 2021), which enhance mathematical text representation, along with other embedding models MATH-Similarity (Steinfeldt and Mihaljević, 2024), lack optimization for identifying functional equivalence between mathematical sentences. This limitation is further exacerbated by the lack of specialized benchmark datasets de-

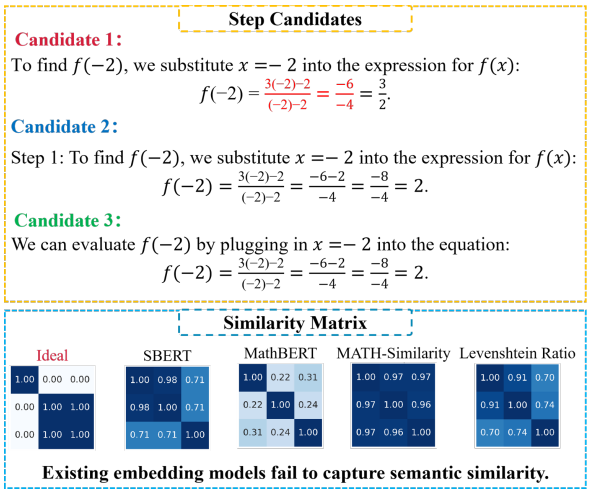

Figure 1: Illustration of the mathematical statement equivalence challenge during reasoning search. Given multiple candidate steps generated by an LLM, standard methods like embedding similarity or Levenshtein Ratio may incorrectly assess candidate 1 and candidate 2 as highly similar due to surface features, while failing to recognize the true semantic equivalence between candidate 2 and candidate 3, which represent the identical logical operation.

signed for mathematical statement equivalence. Although large-scale generative models can achieve satisfactory performance in few-shot scenarios for such judgment tasks, their substantially higher computational complexity results in significantly slower inference speeds compared to embedding models (Brown et al., 2020). The consequent latency renders them impractical for high-throughput applications requiring real-time processing.

To overcome these limitations, we introduce *EquivPruner*, a simple yet effective approach that centers on identifying and pruning semantically equivalent actions during LLM reasoning search. We create MathEquiv, the first dataset specifically designed for mathematical statement equivalence. Leveraging this dataset, we trained a lightweight yet effective equivalence detection model. This model serves as a dynamic pruner integrated into the LLM's search process. When the LLM generates multiple candidate reasoning steps at a given expansion point, the pruner identifies sets of semantically equivalent candidates among these siblings. For each set of equivalent steps, it retains only a single representative node for further exploration, effectively pruning the redundant branches and significantly reducing the search space.

While the proposed pruning framework is potentially generalizable, this paper focuses on its validation within mathematical reasoning due to the significant research community attention (Ke et al., 2025) and the availability of well-developed open-source process reward models (Shao et al., 2024). We conduct extensive experiments across various models, including Mistral-7B-SFT (Shao et al., 2024) and the Qwen2.5-Math-Instruct series (Yang et al., 2024), using two widely recognized math reasoning benchmarks: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). Our proposed EquivPruner demonstrates compelling improvements across these settings. For instance, when applied to Qwen2.5-Math-7B-Instruct on GSM8K—where the model already achieves a very high baseline accuracy of 96.44%—EquivPruner not only cuts token consumption by a substantial 48.1% but also further boosts accuracy to 96.59%. This demonstrates EquivPruner's ability to significantly enhance searching efficiency.

Our main contributions are:

- To the best of our knowledge, this work is the *first* to identify and address the problem of action equivalence in LLM-based reasoning search.
- We introduce *EquivPruner*, a simple yet effective approach that centers on identifying and pruning semantically equivalent actions during LLM reasoning search.
- We release MathEquiv, the first benchmark dataset specifically designed for mathematical statement equivalence. It serves as a versatile resource applicable to a variety of mathematical tasks and scenarios.
- Extensive experiments demonstrate the effectiveness of *EquivPruner*. When applied to Qwen2.5-Math-7B-Instruct on GSM8K, *EquivPruner* not only cuts token consumption by a substantial 48.1% but also further boosts accuracy in a very high baseline.

2 Related Work

LLM Reasoning via Search Strategies Efforts to improve LLM problem-solving capabilities have moved beyond simple prompting. Chain-of-Thought prompting (Wei et al., 2022) demonstrated the value of intermediate reasoning steps. Building on this, structured search methods like Tree-of-Thoughts (Yao et al., 2023) and Graph-of-Thoughts (Besta et al., 2024) explore multiple

reasoning paths, enhancing performance on complex tasks requiring exploration and backtracking. Further advancing this direction, a particularly powerful paradigm integrates LLMs with sophisticated search algorithms. Among these, the synergy between LLMs and Monte Carlo Tree Search (MCTS) (Chen et al., 2024; Zhang et al., 2024a) has garnered significant attention for tackling complex reasoning problems. MCTS, renowned for its ability to balance exploration and exploitation in vast search spaces, becomes exceptionally potent when guided by an LLM's generative capabilities to propose candidate steps and a reward model to estimate state values (Yao et al., 2023; Long, 2023; Besta et al., 2024). This LLM-MCTS approach, alongside other advanced search integrations like LLM-guided beam search (Chen et al., 2024) and depth-first-search (Yao et al., 2023), has consistently achieved state-of-the-art results in demanding areas such as science tasks (Yang et al., 2024), coding (Dainese et al., 2024; Zhang et al., 2023), and mathematical reasoning (Zhang et al., 2024b; Luo et al., 2024). However, despite the remarkable success of these advanced search strategies, a significant challenge emerges, especially prevalent in mathematical reasoning when employing methods like LLM-MCTS: the substantial token cost (Chen et al., 2024). While LLM-MCTS explores many branches effectively, it often wastes resources evaluating syntactically distinct but semantically equivalent states. This redundancy unnecessarily expands the search space, consuming tokens without yielding novel solutions, thus limiting efficiency and scalability.

Mathematical Equivalence Detection Effective detection of mathematical statement equivalence is crucial for efficient LLM-Based search tree pruning, yet current methodologies exhibit significant shortcomings. For instance, rudimentary sequence comparison metrics like Levenshtein similarity (Yujian and Bo, 2007) are fundamentally ill-suited, as they fail to capture the deep semantic and hierarchical structures inherent in mathematical language, leading to unreliable equivalence assessments. Standard Semantic Textual Similarity models, such as SBERT (Reimers and Gurevych, 2019), trained predominantly on general language corpora, are designed to capture semantic relatedness rather than strict mathematical equivalence. Even domain-specific models like MathBERT (Peng et al., 2021), which enhance mathematical text representation, along with other embedding models MATH-Similarity (Steinfeldt and Mihaljević, 2024), lack optimization for identifying functional equivalence between mathematical sentences. Their capacity to accurately recognize semantically equivalent mathematical sentences is thereby constrained, as illustrated by the examples in Figure 1. While LLMs like GPT-4o (Hurst et al., 2024) has the ability to recognize mathematical equivalence, their complex architectures introduce significant latency. This high time overhead renders them impractical for real-time pruning scenarios. Consequently, there is an urgent need to enable efficient pruning in LLM-based search.

3 Methodology

3.1 Define Semantic Equivalence in Mathematics

Simply equating statements based on identical outcomes can be superficial and misleading, as it may overlook critical differences in conceptual articulation, structural formulation, symbolic interpretation, and methodological pathways. To address this, we propose a definition of semantic equivalence specifically attuned to these multifaceted aspects. Accordingly, in our framework, two mathematical statements are considered semantically equivalent if and only if they rigorously satisfy the following criteria:

- **Conceptual Consistency:** The statements must articulate identical mathematical concepts, definitions, or propositions without ambiguity.
- **Structural Equivalence:** Their logical formulations, encompassing assumptions, derivations, and conclusions, must be fully aligned.
- **Notational Precision:** All variables, symbols, and mathematical expressions must be employed consistently, maintaining identical meanings across the statements.
- **Methodological Congruence:** Semantic equivalence necessitates an alignment in the underlying methodology and reasoning. Statements yielding the same final result via disparate approaches are not considered fully equivalent.

Our approach to semantic equivalence thus mandates a comprehensive assessment. It scrutinizes

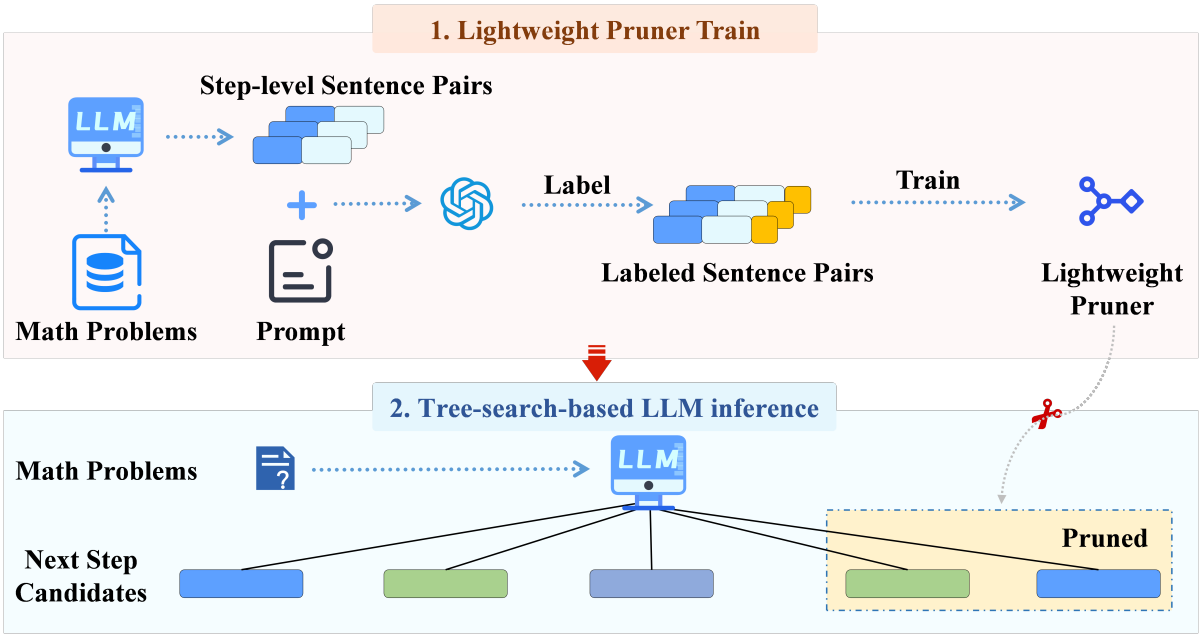

Figure 2: The EquivPruner framework. Top: Training the lightweight equivalence pruner from labeled step-level sentence pairs. Bottom: Applying the trained lightweight pruner during tree-search-based LLM inference to remove semantically equivalent candidates generated by the LLM.

the congruence of conceptual foundations, logical structures, notational usage, and methodological approaches. Two mathematical statements are judged completely equivalent only when they demonstrate indivisible identity across all these critical facets.

3.2 The MathEquiv Dataset

Recognizing the absence of dedicated datasets for mathematical statement equivalence, we constructed and released MathEquiv to bridge this gap. The MathEquiv dataset was curated by initially employing a Step-level Beam Search algorithm (Chen et al., 2024) to gather action candidates. These candidates were subsequently formulated into step-level sentence pairs.

For the task of equivalence scoring, we implemented a five-tiered classification system. This granular approach was adopted to enhance the stability of the GPT model's outputs, as preliminary experiments with binary classification (equivalent/non-equivalent) revealed inconsistencies in judgments. The five-tiered system yielded significantly more consistent and reliable assessments:

- **Level 4 (Exactly Equivalent):** The statements are mathematically interchangeable in all respects, exhibiting identical meaning and form.
- **Level 3 (Likely Equivalent):** Minor syntactic differences may be present, but the core mathematical content and logic align.
- **Level 2 (Indeterminable):** Insufficient information is available to make a definitive judgment regarding equivalence.
- **Level 1 (Unlikely Equivalent):** While some partial agreement may exist, critical discrepancies in logic, definition, or mathematical structure are observed.
- **Level 0 (Not Equivalent):** The statements are fundamentally distinct in their mathematical meaning, derivation, or resultant outcomes.

The MathEquiv dataset was labeled via an iterative refinement process. Initially, GPT-4o labeled a data subset, followed by human expert review. For discrepancies, the human-adjudicated label and its rationale were incorporated into GPT-4o's prompt as few-shot examples. This cycle was repeated until model outputs for a randomly sampled subset consistently aligned with human consensus. Subsequently, the collection of few-shot examples was pruned to a minimal, representative set sufficient to maintain this model-human alignment. This iterative calibration process yielded the MathEquiv dataset, characterized by high-quality labels and an accurate assessment of semantic equivalence.

The final prompt is detailed in Figure 4. We show examples of the contents of MathEquiv in Figure 5 and Figure 6. The MathEquiv dataset is available at https://anonymous.4open.science/r/MathEquiv_dataset-B2B2/.

3.3 Lightweight Pruner for Tree Search

To facilitate dynamic, real-time pruning within our tree search algorithm, we developed and trained a dedicated Lightweight Pruner. The data collection process for training this pruner and its integration into the broader Tree-search-based LLM inference pipeline are illustrated in Figure 2.

3.3.1 Data Complexity in Pruner Training

The MathEquiv dataset, suitable for assessing overall statement equivalence, presents specific challenges for training the Lightweight Pruner. The dataset's step-level sentence pairs often consist of multiple sentences. A key difficulty is that step pairs labeled as non-equivalent at a macro-level may nevertheless contain sub-pairs of sentences that are semantically equivalent. This characteristic, common in data derived from intermediate mathematical problem-solving steps, can introduce ambiguity and hinder the pruner's ability to learn fine-grained distinctions if not appropriately addressed. The true equivalence status of these sub-sentence pairs can be viewed as a latent aspect of the data.

3.3.2 Pruner Training via Expectation-Maximization (EM)

To effectively train the Lightweight Pruner amidst this data complexity, we employ the Expectation-Maximization (EM) algorithm, which is effective for handling the unobserved equivalence status of sub-sentence pairs within larger, complex training instances. The algorithm iteratively maximizes the expected complete-data log-likelihood by alternating between two steps:

1. E-step (Expectation Step): Given the model parameters $\theta^{(t)}$ at iteration t, the pruner predicts the equivalence probability of each sub-sentence pair in multi-sentence samples. Sub-sentence pairs with probabilities exceeding a threshold are treated as high-confidence equivalents and removed from samples to refine the dataset for the next step.

2. M-step (Maximization Step): The model parameters are updated to $\theta^{(t+1)}$ by maximizing the likelihood of the observed data, conditioned on the expectations derived in the E-step.

By training on samples that have been simplified or where latent equivalences have been accounted for, the model can better focus on learning more subtle or challenging distinctions necessary for effective pruning.

3.4 Pruner's Impact on Different Search Algorithms

The pruner enhances search performance across various algorithms by eliminating state redundancy, with its contribution tailored to the architecture of each search strategy. For heuristic-driven algorithms like Monte Carlo Tree Search (MCTS), it optimizes the allocation of the computational budget (i.e., simulation rollouts) by collapsing equivalent states. This concentrates exploration on unique paths, which improves the accuracy of the guiding policy statistics and accelerates convergence. In Step-level Beam Search (SBS), the pruner directly improves search quality by diversifying the beam. By removing equivalent nodes *prior* to the top-k selection, it ensures that the beam's limited slots are occupied by unique candidates, mitigating the risk of optimal paths being prematurely pruned. For systematic algorithms such as Depth-First Search (DFS), it yields a direct efficiency gain by preventing the repeated exploration of identical subtrees.

4 Experiments

In this section, we present a series of comprehensive experiments designed to validate the efficacy of EquivPruner.

4.1 MathEquiv Dataset Generation

We constructed the MathEquiv dataset for mathematical statement equivalence. The foundation of this dataset consists of 7,500 mathematical problems sourced from the MATH training set (Hendrycks et al., 2021). To prevent data leakage between training, validation, and test phases of EquivPruner, these 7,500 problems were first split into training, validation, and test sets using an 8:1:1 ratio, respectively. For each problem in these distinct sets, we generated candidate reasoning step pairs using the Qwen2.5-Math-7B-Instruct model (Yang et al., 2024) via Step-level Beam Search. These pairs were subsequently filtered based on Levenshtein distance, and a balanced sample from each set was then annotated for equivalence by GPT-4o. This process resulted in distinct training, validation, and test sets of annotated step pairs for EquivPruner. The specific parameters for step pair

Table 1: Performance comparison of Vanilla MCTS and MCTS + EquivPruner across three language models on the MATH and GSM8K datasets. EquivPruner significantly reduces token consumption (Tokens, Ratio) while generally maintaining or improving accuracy (Acc, %). Best results within each model-dataset block are in **bold**.

Methods	MATH			GSM8K		
	Acc	Tokens	Ratio	Acc	Tokens	Ratio
Qwen2.5-Math-7B-Instruct:						
Vanilla MCTS	83.40	106773	100.00%	96.44	34826	100.00%
+ EquivPruner	**84.00**	**74194**	**69.49%**	**96.59**	**18071**	**51.89%**
Mistral-7b-sft:						
Vanilla MCTS	36.60	49251	100.00%	83.78	20217	100.00%
+ EquivPruner	**37.40**	**38265**	**77.69%**	**85.06**	**12537**	**62.01%**
Qwen2.5-Math-1.5B-Instruct:						
Vanilla MCTS	75.60	91811	100.00%	**91.05**	39337	100.00%
+ EquivPruner	**75.60**	**71878**	**78.29%**	90.75	**23752**	**60.38%**

generation, filtering criteria, and the final dataset sizes are detailed in Appendix A.1.

4.2 Experimental Setup

4.2.1 Models and Datasets

For inference, we utilized several LLMs: Qwen2.5-Math-7B-Instruct (Yang et al., 2024), Mistral-7B-SFT (Shao et al., 2024), and Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024). Given that existing open-source PRMs are predominantly tailored for mathematical reasoning, our current investigation is confined to mathematical tasks. Nevertheless, the EquivPruner framework is designed for generalizability and can be readily extended to other domains like code generation and commonsense reasoning. The Process Reward Model (PRM) employed for guiding the Monte Carlo Tree Search (MCTS) was Math-Shepherd-Mistral-7B-PRM (Shao et al., 2024). As EquivPruner was trained on data generated by Qwen2.5-Math-7B-Instruct, the Mistral-7B-SFT and Qwen2.5-Math-1.5B-Instruct models serve as out-of-distribution (OOD) models in our experiments.

Our evaluation was conducted on two standard benchmark datasets:

- **MATH** (Hendrycks et al., 2021): Featuring challenging competition-level mathematics problems. Due to computational demands, our evaluation on the MATH dataset was performed on the MATH-500 subset, identical to the test partition used in Lightman et al. (2023).
- **GSM8K** (Cobbe et al., 2021): Consisting of grade school mathematics word problems. Its test set has 1319 problems. Since EquivPruner was trained on data derived from MATH dataset problems, GSM8K is considered an OOD dataset.
- **AIME 2024** (Codeforces): The AIME 2024 dataset consists of 30 challenging mathematics problems spanning various domains, including algebra, geometry, and number theory. AIME 2024 is also considered an OOD dataset.

4.2.2 Implementation Details

The EquivPruner model itself is a fine-tuned Longformer-base (Beltagy et al., 2020), chosen for its efficiency suitable for real-time pruning. We report evaluation results of pruner in Appendix A.2. During the MCTS inference phase, the determination of equivalence between two reasoning step nodes involves a two-stage process. First, the Levenshtein ratio between the steps is calculated. If the ratio is less than or equal to 0.75, the nodes are immediately considered non-equivalent, acting as a fast filter. Only if the Levenshtein ratio is greater than 0.75 is the EquivPruner model invoked to make the final equivalence prediction. This hierarchical check balances speed and accuracy in the pruning process. The maximum number of newly generated tokens by the LLMs (max_new_tokens) was set to 1024, and the generation temperature was 0.7. All experiments were conducted on NVIDIA GeForce RTX 3090 GPUs. Further details are available in Appendix A.3.

4.3 Evaluation Metrics

We adopted a vanilla MCTS (Chen et al., 2024) as the baseline for comparison. The evaluation of

EquivPruner focuses on two primary aspects:

- **Effectiveness**: Measured using solution accuracy (Acc), the percentage of problems solved correctly.
- **Efficiency**: Assessed through the average number of tokens generated (Tokens) and a token ratio (Ratio), defined as the ratio of tokens generated by the EquivPruner-enhanced search to those generated by the baseline MCTS.

4.4 Main Results

Table 1 presents our main experimental findings, comparing vanilla MCTS against MCTS augmented with EquivPruner on MATH dataset and GSM8K dataset. We also conducted experiments on AIME 2024 dataset in the Appendix A.4. The results consistently demonstrate that EquivPruner substantially enhances computational efficiency across different language models and datasets, primarily by reducing token generation while largely preserving or even improving solution accuracy. Due to page limitations, the detailed ablation studies are provided in Appendix A.5.

Efficiency Gains EquivPruner achieves significant reductions in token counts across all configurations. For instance, with Qwen2.5-Math-7B-Instruct on GSM8K, tokens were reduced by approximately 48.11% (Ratio: 51.89%), and on MATH, by 30.51% (Ratio: 69.49%). Similar substantial token savings were observed for Mistral-7B-SFT (e.g., 37.99% reduction on GSM8K) and Qwen2.5-Math-1.5B-Instruct (e.g., 39.62% reduction on GSM8K). These figures highlight EquivPruner's effectiveness in pruning the search space.

Accuracy Impact and Resource Optimization Crucially, these efficiency improvements are generally accompanied by maintained or enhanced accuracy. Qwen2.5-Math-7B-Instruct saw accuracy gains of +0.60% on MATH and +0.15% on GSM8K. With Mistral-7B-SFT, an OOD model relative to EquivPruner's training data source, accuracy improved by +0.80% on MATH and +1.28% on GSM8K (also an OOD dataset for EquivPruner). This suggests that by eliminating redundant explorations, EquivPruner enables MCTS to allocate its search resources more effectively. For Qwen2.5-Math-1.5B-Instruct (another OOD model), accuracy was maintained on MATH and saw a minor dip of -0.30% on GSM8K, which is a reasonable trade-off given the nearly 40% token reduction.

Table 2: Performance comparison of SBERT, MATH-Similarity, Levenshtein ratio (threshold = 0.95), and EquivPruner for pruning across two language models on the MATH datasets. EquivPruner significantly reduces token consumption (Tokens) while improving accuracy (Acc, %).

Methods	Acc	Tokens
Qwen2.5-Math-7B-Instruct:		
Vanilla MCTS	83.40	106773
+SBERT Pruner	82.60	70142
+MATH-Similarity Pruner	82.00	76693
+Levenshtein Ratio Pruner	83.00	105118
+EquivPruner (Ours)	84.00	74194
Mistral-7b-sft:		
Vanilla MCTS	36.60	49251
+SBERT Pruner	33.00	30876
+MATH-Similarity Pruner	34.40	33212
+Levenshtein Ratio Pruner	35.50	46608
+EquivPruner(Ours)	37.40	38265

Generalization The positive outcomes on OOD models (Mistral-7B-SFT, Qwen2.5-Math-1.5B-Instruct) and the OOD dataset (GSM8K) underscore EquivPruner's generalization capabilities. It effectively identifies and removes equivalent reasoning steps, allowing MCTS to conduct a more focused and efficient search across varied settings.

4.5 Baseline Comparison with Alternative Pruning Methods

To further evaluatethe effectiveness of our proposed EquivPruner, we compare it against alternative pruning methods based on semantic similarity (SBERT, MATH-Similarity) and string similarity (Levenshtein Ratio, threshold = 0.95). We also explored the Formula Extraction-based method in the Appendix A.6 considering the challenges in consistently parsing mathematical expressions.

Table 2 shows that these baselines are suboptimal. Embedding-based pruners reduce tokens but at the cost of accuracy; for instance, they caused an accuracy drop of up to 3.6% with the Mistral-7B-sft model. This indicates semantic similarity is a poor proxy for mathematical equivalence, leading to the erroneous pruning of valid reasoning paths. Similarly, the Levenshtein Ratio pruner, while maintaining accuracy, provided a negligible token reduction of only 1.5% with the Qwen model.

Table 3: Performance of EquivPruner with Step-level Beam Search (SBS) using the Qwen2.5-Math-7B-Instruct model on MATH, GSM8K and AIME 2024. EquivPruner enhances accuracy (Acc, %) by promoting diversity among selected nodes, with token counts (Tokens, Ratio) remaining largely stable.

Methods	Acc	Tokens	Ratio
MATH:			
SBS	82.00	21341	100.00%
+ EquivPruner	82.20	20952	98.18%
GSM8K:			
SBS	96.06	8004	100.00%
+ EquivPruner	96.13	7927	99.04%
AIME 2024:			
SBS	10.00	46672	100.00%
+ EquivPruner	13.33	43391	92.97%

In contrast, our EquivPruner improves both efficiency and performance. It achieved a 30.4% token reduction with Qwen2.5-Math-7B-Instruct while also increasing accuracy on both models (e.g., from 36.6% to 37.4% on Mistral-7B-sft). This confirms that checking for functional equivalence is superior to relying on semantic or textual similarity for pruning.

4.6 Effectiveness in Step-level Beam Search

To empirically validate the pruner's benefit for SBS, we integrated it into the search process using the Qwen2.5-Math-7B-Instruct model.

The results in Table 3 confirm its positive impact on search quality. EquivPruner increased accuracy on MATH from 82.00% to 82.20% (+0.20%), on GSM8K from 96.06% to 96.13% (+0.07%), and on AIME 2024 from 10.00% to 13.33% (+3.33%). As hypothesized, these gains were achieved without significant changes in token generation, with ratios of 98.18% (MATH), 99.04% (GSM8K), and 92.97% (AIME 2024). These findings show that by ensuring the top-k candidates are distinct, the pruner fosters a more diverse and effective exploration of the solution space, validating its role as a versatile component for improving LLM reasoning search strategies.

4.7 Effectiveness in Depth-First-Search

To validate the versatility of our EquivPruner beyond heuristic search, we integrated it into a Depth-First Search (DFS) framework using the Qwen2.5-Math-7B-Instruct model. The results in Table 4 show that the pruner is highly effective even on systematic search algorithms.

Table 4: Performance of EquivPruner with Depth-First-Search (DFS) using the Qwen2.5-Math-7B-Instruct model on MATH and GSM8K. EquivPruner significantly reduces token consumption (Tokens, Ratio) while improving accuracy (Acc, %).

Methods	Acc	Tokens	Ratio
MATH:			
DFS	81.40	44229	100.00%
+EquivPruner	81.60	30943	69.96%
GSM8K:			
DFS	95.60	14821	100.00%
+EquivPruner	95.68	9648	65.10%

On both MATH and GSM8K benchmarks, EquivPruner significantly reduces token consumption by over 30% and 34.9% respectively, while also marginally improving accuracy. This efficiency gain is achieved by preventing the redundant traversal of identical subtrees after merging equivalent states. The slight accuracy improvement suggests that by eliminating redundant paths, the search can explore the solution space more effectively, confirming that EquivPruner is a robust, algorithm-agnostic optimization.

5 Conclusion

In this paper, we introduce *EquivPruner*, a simple yet effective approach to address inefficient token usage in LLM reasoning search by identifying and pruning semantically equivalent actions. We also introduce MathEquiv, the first dataset specifically designed for mathematical statement equivalence, which enables the training of an effective lightweight equivalence detector. Extensive experiments demonstrate that *EquivPruner* significantly reduces token consumption—for example, by 48.1% for Qwen2.5-Math-7B-Instruct on GSM8K—while maintaining or often improving reasoning accuracy across various models and tasks. Our findings underscore the substantial benefits of managing semantic redundancy in reasoning search, offering a valuable direction for enhancing the efficiency and effectiveness of LLMs.

Limitations

There are some limitations with our paper, which we reserve for future work. Firstly, due to computational constraints, *EquivPruner* was not evaluated on language models significantly larger than the 7B

parameter scale. Secondly, our work focused on *EquivPruner*'s application at inference time, and its potential integration with iterative LLM training or refinement strategies remains an area for future exploration. Lastly, while designed for generalizability, our empirical validation was primarily within mathematical reasoning, influenced by the availability of suitable process reward models (PRMs). Extending evaluation to other domains such as science tasks or commonsense reasoning, contingent upon broader PRM availability, offers a promising direction for future research.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: Process supervision without process. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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*.

MAA Codeforces. American invitational mathematics examination-aime 2024, 2024.

Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. 2024. Generating code world models with large language models guided by monte carlo tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. 2024. Learning how hard to think: Input-adaptive allocation of lm computation. *arXiv preprint arXiv:2410.04707*.

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*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jinhao Jiang, Zhipeng Chen, Yingqian Min, Jie Chen, Xiaoxue Cheng, Jiapeng Wang, Yiru Tang, Haoxiang Sun, Jia Deng, Wayne Xin Zhao, and 1 others. 2024. Technical report: Enhancing llm reasoning with reward-guided tree search. *arXiv preprint arXiv:2411.11694*.

Jikun Kang, Xin Zhe Li, Xi Chen, Amirreza Kazemi, Qianyi Sun, Boxing Chen, Dong Li, Xu He, Quan He, Feng Wen, and 1 others. 2024. Mindstar: Enhancing math reasoning in pre-trained llms at inference time. *arXiv preprint arXiv:2405.16265*.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, and 1 others. 2025. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *arXiv preprint arXiv:2504.09037*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Jieyi Long. 2023. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, and 1 others. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Goutam Majumder, Partha Pakray, Alexander Gelbukh, and David Pinto. 2016. Semantic textual similarity methods, tools, and applications: A survey. *Computación y Sistemas*, 20(4):647–665.

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*.

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.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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*.

Christian Steinfeldt and Helena Mihaljević. 2024. Evaluation and domain adaptation of similarity models for short mathematical texts. In *International Conference on Intelligent Computer Mathematics*, pages 241–260. Springer.

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.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, and 1 others. 2024b. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. 2023. Planning with large language models for code generation. In *The Eleventh International Conference on Learning Representations*.

A Experimental Details

A.1 MathEquiv Dataset Generation Details

The MathEquiv dataset was constructed as follows:

Problem Sourcing and Splitting: We selected 7,500 problems from the MATH training set (Hendrycks et al., 2021). These problems were divided into three distinct sets for EquivPruner: a training set (6,000 problems, 80%), a validation set (750 problems, 10%), and a test set (750 problems, 10%). This initial split of problems ensures no data leakage between the subsequently generated step-pair datasets for EquivPruner. **Step Pair Generation**: For each problem within these three sets, we generated candidate reasoning steps using the Qwen2.5-Math-7B-Instruct model (Yang et al., 2024). This generation was performed via a Step-level Beam Search with the following parameters: beam size (k) = 8, temperature = 0.7, maximum search tree width (tree_max_width) = 10, maximum search tree depth (tree_max_depth) = 50, and maximum new tokens for generation (max_new_tokens) = 1024. **Filtering**: The generated step pairs from each set were then filtered based on their Levenshtein ratio. Only pairs with a ratio between 0.75 and 0.99 (inclusive) were retained. This filtering aimed to capture meaningful variations while excluding nearly identical or overly dissimilar steps. **Sampling and Annotation**: From the filtered pairs of each set, we randomly sampled a large number for annotation: Training set: 80,000 pairs were annotated. Validation set: 10,000 pairs were annotated. Test set: 10,000 pairs were annotated. This process resulted in the final training, validation, and test sets for the MathEquiv dataset, with no overlap in the underlying problems from which the step pairs were derived.

A.2 Evaluation Results of Pruner

We evaluated the final EquivPruner model on the held-out test set to assess the impact of incorporating the Expectation-Maximization (EM) algorithm. As shown in Table 5, the experimental results confirm that integrating EM into the training process improves the pruner's performance.

A.3 Implementation Environment and MCTS Parameters

All experiments were conducted using PyTorch version 2.4.0. The GPU infrastructure consisted of eight NVIDIA GeForce RTX 3090 GPUs, each

Table 5: Performance comparison of the EquivPruner on the test set, with and without the Expectation-Maximization (EM) algorithm. The best results are highlighted in bold.

Method	Without EM	With EM
F1	83.88%	**86.20%**
Accuracy	79.40%	**82.97%**
Precision	84.01%	**85.45%**
Recall	83.75%	**86.97%**

with 24GB, utilizing CUDA version 12.1. The central processing unit was an Intel(R) Xeon(R) Platinum 8255C CPU equipped with 96 cores.

A.3.1 EquivPruner Training

The EquivPruner model, a fine-tuned Longformer-base (Beltagy et al., 2020), was trained using hyperparameters selected via Bayesian optimization. The optimization aimed to maximize the 'eval/f1' metric over a maximum of 10 trials. The hyperparameter search spaces are detailed in Table 6.

Table 6: Hyperparameter search space for EquivPruner using Bayesian optimization.

Hyperparameter	Value or Range
Learning Rate	[1e-6,5e-5]
Batch Size	4
Training Epochs	Discrete Values {2, 3, 5}
Weight Decay	[0.0, 0.1]

A.3.2 MCTS Parameters

The Monte Carlo Tree Search (MCTS) based evaluation hyperparameters are detailed in Table 7. These MCTS parameters (temperature, tree_max_width, tree_max_depth, simulations, PUCT values) were kept consistent across baseline and EquivPruner-enhanced evaluations unless otherwise specified.

Table 7: Monte Carlo Tree Search (MCTS) hyperparameters.

Hyperparameter	Value
Number of Simulations	20
LLM Generation Temperature	0.7
LLM max_new_tokens	1024
Search Tree Maximum Width	10
Search Tree Maximum Depth	50
PUCT values	1.25

A.3.3 SBS Parameters

The Step-level Beam Search (SBS) based evaluation hyperparameters are detailed in Table 8. These SBS parameters (beam size, temperature, tree_max_width, tree_max_depth) were kept consistent across baseline and EquivPruner-enhanced evaluations unless otherwise specified.

Table 8: Step-level Beam Search (SBS) hyperparameters.

Hyperparameter	Value
Beam Size	3
LLM Generation Temperature	0.7
LLM max_new_tokens	1024
Search Tree Maximum Width	10
Search Tree Maximum Depth	50

A.3.4 DFS Parameters

The Depth-First-Search (DFS) based evaluation hyperparameters are detailed in Table 9. These DFS parameters (num_sequence, temperature, tree_max_width, tree_max_depth) were kept consistent across baseline and EquivPruner-enhanced evaluations unless otherwise specified.

Table 9: Depth-First-Search (DFS) hyperparameters.

Hyperparameter	Value
num_sequence	3
LLM Generation Temperature	0.7
LLM max_new_tokens	1024
Search Tree Maximum Width	10
Search Tree Maximum Depth	50

A.4 Experiments on AIME 2024 dataset

We conducted additional experiments on the AIME dataset using both the Mistral-7B-sft and Qwen2.5-Math-7B-Instruct models. As shown in Table 10, these results on AIME consistently show that our EquivPruner effectively reduces token consumption across different models and search algorithms without compromising performance.

A.5 Ablation Study

To investigate the individual contributions of the key components of our EquivPruner—specifically, the fine-tuning process and the use of the EM algorithm—we conducted an ablation study. The experiments were performed using the Qwen2.5-Math-7B-Instruct model on the MATH dataset. We

Table 10: Performance of EquivPruner with MCTS using the Qwen2.5-Math-7B-Instruct model and Mistral-7B-sft model on AIME 2024.

Methods	Acc	Tokens
Qwen2.5-Math-7B-Instruct:		
Vanilla MCTS	16.67	220446
+ EquivPruner	16.67	204654
Mistral-7B-sft:		
Vanilla MCTS	0.00	87002
+ EquivPruner	3.33	68832

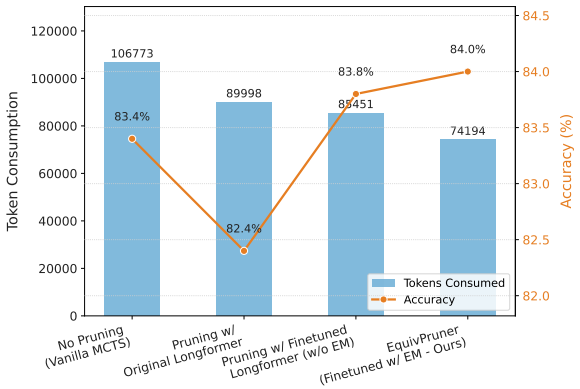

Figure 3: Ablation study of EquivPruner components. The plot illustrates the impact of different pruning strategies within a MCTS framework on Token Consumption (bars, left y-axis) and Accuracy (line, right y-axis).

compare our full method, EquivPruner (Finetuned w/ EM), against three variants: (1) No Pruning (vanilla MCTS baseline); (2) Pruning w/ Original Longformer (using a pre-trained Longformer-base without task-specific fine-tuning for equivalence); and (3) Pruning w/ Finetuned Longformer (w/o EM) (standard supervised fine-tuning without the EM algorithm).

The results in Figure 3 demonstrate the impact of each component. Using the Original Longformer-base for pruning (Setting 2) reduces tokens (106773 to 89998) compared to No Pruning (Setting 1), but at the cost of a accuracy drop (83.4% to 82.4%), indicating that a generic model is insufficient. Standard fine-tuning without EM (Setting 3) improves accuracy to 83.8% (surpassing No Pruning) while improve token efficiency to Setting 2 (89998 to 85451), underscoring the necessity of task-specific training. Critically, our full EquivPruner method with EM-based fine-tuning (Setting 4) achieves both the highest accuracy (84.0%) and the most significant token reduction (106773 to 74194). This highlights that both the fine-tuning process and specifically the EM algorithm are vital for maximizing EquivPruner's effectiveness in improving accuracy and token efficiency.

Table 11: Performance comparison of the formula extraction-based pruning method and EquivPruner on the GSM8K dataset (Model: Qwen2.5-Math-7B-Instruct).

Methods	Acc	Tokens
Vanilla MCTS	96.44	34826
+ Formula Extraction with String Match	96.44	25632
+ Formula Extraction with Embedding Model	96.44	25336
+EquivPruner (Ours)	96.59	18071

A.6 Analysis of Formula Extraction-based Pruning

An intuitive pruning approach in mathematical reasoning involves extracting arithmetic equations from the generated text and directly checking for equivalence. We investigated this formula extraction-based method to evaluate its feasibility and performance.

A primary challenge of this approach is its dependency on the language model's output format. The style of arithmetic equations can be highly diverse across models, making consistent extraction via simple rules difficult. However, we observed that for simpler problems, such as those in the GSM8K dataset, the Qwen2.5-Math-7B-Instruct model often wraps formulas in consistent delimiters . This formatting consistency enabled a targeted experiment. In our experiment, we first extracted formulas from the generated solutions and then applied two techniques to check for equivalence: (1) direct string matching of the extracted formulas and (2) comparing the formulas using an embedding model (MATH-Similarity).

As show in Table 11, the results indicate that both formula-based pruning methods successfully reduced the token count compared to Vanilla MCTS—by 26.4% for string matching and 27.2% for the embedding model. However, neither method yielded an improvement in accuracy, achieving the same 96.44% as the vanilla baseline. In contrast, our EquivPruner achieved a significantly greater token reduction of 48.1% while also increasing accuracy to 96.59%. This experiment highlights the limitations of the formula extraction approach. Even when formatting is consistent

enough for reliable extraction, it fails to outperform EquivPruner in either efficiency or effectiveness. Its reliance on specific delimiters makes the method brittle and not generalizable to other models or more complex problem types where output may be less structured. This justifies our focus on the more robust and versatile execution-based approach implemented in EquivPruner.

Please determine whether the following two sentences are semanticly equivalent, and return 0: Not equivalent at all; 1: May not be equivalent; 2: Can't judge; 3: May be equivalent; 4: Exactly equivalent. Please explain the reason, reflect, and provide a more accurate result. Please output in the following Python dictionary format:
{ "reasoning_step": "The reasoning process of the model", "result": "Final result" (int) }
Question:
Sentence1: The inner sum is a geometric series with the first term $\frac{1}{2^{k^2-k+1}}$ and common ratio $\frac{1}{2}$, and it has $2k$ terms. The sum of a geometric series is given by:

$$\sum_{n=a}^{a+b-1} ar^n = a\frac{1-r^b}{1-r}$$

Sentence2: The inner sum $\sum_{n=k^2-k+1}^{k^2+k} \frac{1}{2^n}$ is a geometric series with the first term $\frac{1}{2^{k^2-k+1}}$ and common ratio $\frac{1}{2}$, and it has $2k$ terms. The sum of a geometric series is given by:

$$\sum_{n=a}^{a+r-1} ar^n = a\frac{1-r^r}{1-r}$$

Output:
{ "reasoning_step": "Sentence1 uses the formula $\sum_{n=a}^{a+b-1} ar^n = a\frac{1-r^b}{1-r}$, while Sentence2 uses $\sum_{n=a}^{a+r-1} ar^n = a\frac{1-r^r}{1-r}$. The variables and the structure of the formulas are different, which affects the semantic equivalence. Additionally, the limits of the summation in Sentence2 are explicitly given as $\sum_{n=k^2-k+1}^{k^2+k} \frac{1}{2^n}$, which is not present in Sentence1. These differences indicate that the sentences are not semantically equivalent.", "result": 0 }
Question:
Sentence1: The amplitude is the distance from the midline to the maximum value (or the minimum value). Therefore, we can calculate the amplitude as follows:

$$Amplitude = Maximumvalue - Midline = 3 - 1 = 2$$

Since the amplitude is $|a|$, we have:|a| = 2. Since a is a positive constant, we conclude:a = 2
Sentence2: The amplitude of the sine wave is half the distance between the maximum value and the minimum value. Therefore, we can calculate the amplitude as follows:

$$Amplitude = \frac{Maximumvalue - Minimumvalue}{2} = \frac{3-(-1)}{2} = \frac{3+1}{2} = \frac{4}{2} = 2$$

Since the amplitude is $|a|$, we have:|a| = 2. Since a is a positive constant, we conclude: a = 2
Output: { "reasoning_step": "Sentence1 defines the amplitude as the distance from the midline to the maximum value, calculating it as $Amplitude = Maximumvalue - Midline = 3 - 1 = 2$. Sentence2 defines the amplitude as half the distance between the maximum and minimum values, calculating it as $Amplitude = \frac{Maximumvalue-Minimumvalue}{2} = \frac{3-(-1)}{2} = 2$. While both methods yield the same result ($a = 2$), the definitions and calculations are fundamentally different. This difference in methodology means the sentences are not semantically equivalent.", "result": 0 }
Question:
Sentence1: {sentence1}
Sentence2: {sentence2}
Output:

Figure 4: Complete prompt for labeling.

Sentence1: Thus, the sum of the elements in the n-th row of Pascal's triangle is 2^n. The function $f(n)$ is the base-10 logarithm of this sum, so we have:
Sentence2: This means that the sum of the elements in the n-th row of Pascal's triangle is 2^n. Therefore, $f(n)$ is the base-10 logarithm of 2^n:
result: 4
reasoning_step: Both sentences convey the same mathematical property of Pascal's triangle, specifically that the sum of the elements in the n-th row is 2^n. Sentence1 states this as a conclusion, while Sentence2 presents it as a clarification. They both express the same final idea regarding the function $f(n)$ being the base-10 logarithm of this sum. Therefore, they are semantically equivalent.

Figure 5: An Exactly Equivalent Example on MathEquiv Dataset.

Sentence1: Since $u \geq \sqrt{3}$ and $2 \geq \sqrt{3}$, $u = 2$ is a valid critical point. We evaluate $g(u)$ at $u = 2$:
Sentence2: Since $u \geq \sqrt{3}$ and $2 \geq \sqrt{3}$, $u = 2$ is a valid critical point. We evaluate $g(u)$ at this critical point and at the boundary $u = \sqrt{3}$:
result: 0
reasoning_step: Sentence1 explicitly evaluates $g(u)$ at $u = 2$ only, while Sentence2 mentions that $g(u)$ is evaluated at both the critical point (which is also $u = 2$) and at the boundary $u = \sqrt{3}$. The additional evaluation at the boundary in Sentence2 introduces a difference that affects the overall semantic message, making them not semantically equivalent.

Figure 6: A Not Equivalent Example on MathEquiv Dataset.