

FANS: Formal Answer Selection for LLM Natural Language Math Reasoning Using *Lean4*

Jiarui Yao¹, Ruida Wang¹, Tong Zhang¹

¹ University of Illinois Urbana-Champaign
{jiarui14, ruidaw, tozhang}@illinois.edu

Abstract

Large Language Models (LLMs) have displayed astonishing abilities in various tasks, especially in text generation, classification, question answering, etc. However, the reasoning ability of LLMs still faces many debates, especially in math reasoning. The inherent ambiguity of Natural Language (NL) limits LLMs' ability to perform verifiable reasoning, making the answers lack coherence and trustworthy support. To tackle the above challenges, we propose a novel framework named **FANS: *F*ormal *A*nswer *S*election for LLM Natural Language Math Reasoning Using *Lean4***. It is a pioneering framework that utilizes *Lean4* to enhance LLMs' NL math reasoning ability. In particular, given an NL math question and LLM-generated answers, FANS first translates it into *Lean4* theorem statements. Then it invokes another *Lean4* prover LLM to produce proofs, and finally verifies the proofs by *Lean4* compiler. Answers are selected based on the verifications. It enhances LLMs' NL math ability in providing a computer-verifiable solution for its correct answer and proposes an alternative method for answer selection beyond the reward model based ones. Our experiments demonstrate the effectiveness of FANS with an improvement of nearly 2% across several math benchmarks, and even higher further based on reward models or in subfields such as algebra and number theory that *Lean4* is better at. The code is available in <https://github.com/MaxwellJryao/FANS>.

1 Introduction

Math reasoning and problem-solving capability have been a hot topic in LLM's research field. Researchers are trying to develop new LLMs (Shao et al., 2024; Yang et al., 2024b; Yu et al., 2023), increasingly harder benchmarks (Cobbe et al., 2021; Hendrycks et al., 2021), and advanced reward model methods for evaluation of answers (Dong et al., 2023; Yang et al., 2024b). However, there have always been debates on whether LLMs are

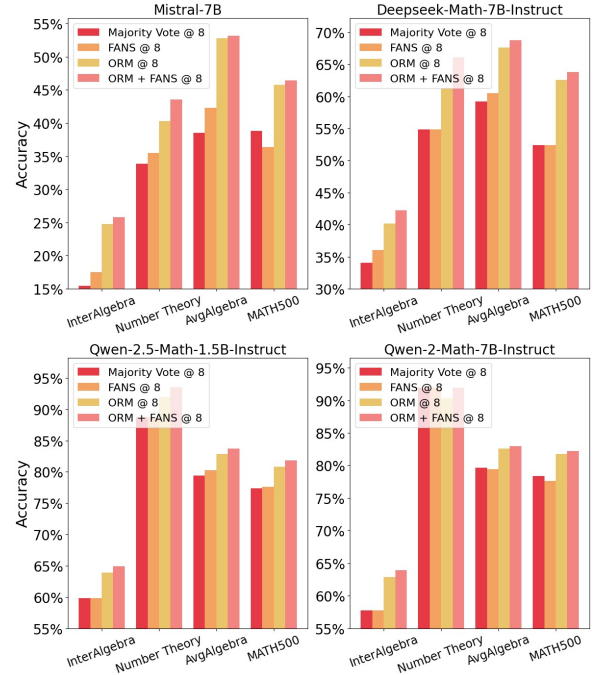


Figure 1: Comparison between FANS and majority vote, together with ORM and ORM + FANS method. From the results, we could see FANS based on ORM achieve the highest accuracies consistently across different base models and different test sets. In particular, FANS performs well on the sub-fields of number theory and algebra, which are better supported by *Lean4* with its existing libraries.

able to perform real reasoning or simply do pattern matching (Saparov et al., 2023). Consequently, researchers have begun to consider using symbolic languages to perform computer-verifiable formal reasoning in mathematics. Consequently, many formal mathematical languages have been proposed such as Lean (De Moura et al., 2015; Moura and Ullrich, 2021), Isabelle (Paulson, 1994), and HOL Light (Harrison, 2009). These Formal Languages (FLs) map existing mathematics into their formal kernels, which allows computers to automatically verify mathematical proofs. FL provides a clear

standard for evaluating answers to theorem proof and significantly impacts both mathematical and computer science communities.

In the context of formal reasoning, most existing works focus on how to use NL to enhance the FL capability of the model (Xin et al., 2024b; Wang et al., 2024b; Lin et al., 2024; Wang et al., 2025b). Research has also shown that formal language solutions to a math problem can be translated back to a valid natural language proof with very few errors (Jiang et al., 2022). Despite extensive research on formal languages (FLs), few have successfully leveraged Lean4 or other formal languages to enhance LLMs’ natural language based mathematical reasoning. On the other hand, for complex natural language math reasoning problems, most LLMs struggle to find a correct solution in limited attempts, but could eventually provide the correct solution through multiple attempts. This leads to a natural idea of scaling the test-time computation resources in NL reasoning to get the correct answer. Dong et al. (2024); Shao et al. (2024); Yang et al. (2024b) shown that a model with scale around 7B without advanced fine-tuning methods could achieve a pass@n accuracy around 90% on MATH500 (Lightman et al., 2023) after a sufficient number of trials. Much recent literature has revealed the effectiveness of test-time scaling for improving the model’s performance (Muennighoff et al., 2025; Chen et al., 2024; Snell et al., 2024; Wu et al., 2024a; Guan et al., 2025).

However, this raises an important question of how to select the correct answer from outputs from the model, which is highly non-trivial especially in the case that we do not have the ground truth answer. Fundamentally speaking, this partially originates from mathematical reasoning being a task with a solid logical foundation, but natural language’s inherent ambiguity makes the generated results not trustworthy enough. To solve this problem, many answer-selection methods have been proposed, ranging from majority vote to best-of-N ranking built upon various reward models (Dong et al., 2024; Yang et al., 2024b). But these methods are either too simple to fulfill the pass@n potential, or still pure LLM-based, failing to make use of the solid formal foundation of math reasoning. In addition, it is hard for reward models to generalize to out-of-distribution (OOD) domains or models due to the distribution shift.

To solve the above challenges, we propose our novel FANS framework. It contains the following

stages that work synergistically to enhance answer selection and provide a verifiable formal foundation for NL math reasoning:

(a) NL to FL translation: Based on the outstanding reasoning of Long Chain-of-Thought (CoT), we propose the training method for the Long CoT translator named *LeanTranslator*. It can translate general math question-answering problems into Lean4 provable formal statements. Applying the translator to question-answer pairs generated by LLMs in NL math reasoning, we can obtain Lean4 statements for further processing.

(b) Lean4 proof writing and verification: We pass the generated statement to an advanced prover to generate the proofs for the FL statement, then use the Lean4 verifier to judge the correctness of the proofs. Final answer selection will be based on the verification results. **(c) Answer selection and output:** After we obtain the verification results, we can combine them with existing answer selection methods jointly. Since our answers are formally provable, it conveys a more solid foundation and trustworthiness than direct NL reasoning.

We summarize our contributions as follows:

1. We propose FANS, a pioneering framework that applies formal mathematical language, Lean4, in enhancing the capability of LLMs to solve NL math problems.
2. In our framework, we propose a new method to train the Long CoT translator that translates NL question-answer pairs into Lean4 statements. To the best of our knowledge, this is the first work that proposes such methods.
3. Extensive experiments indicate that enhanced by our framework, LLMs are able to perform better answer selection. It can improve the accuracy on MATH500 by at most **1.91%** and AMC23 by at most **8.33%**. In some particular subfields that Lean is better at, we can even select **all** correct solutions. The qualitative studies also indicate that our framework can make the NL solutions generated by the model have a solid basis using formal language, enhancing the trustworthiness of the answer.

2 Methodology

High-level speaking, FANS framework can be decomposed into three stages. Firstly, we introduce methods for translation for Natural Language (NL)

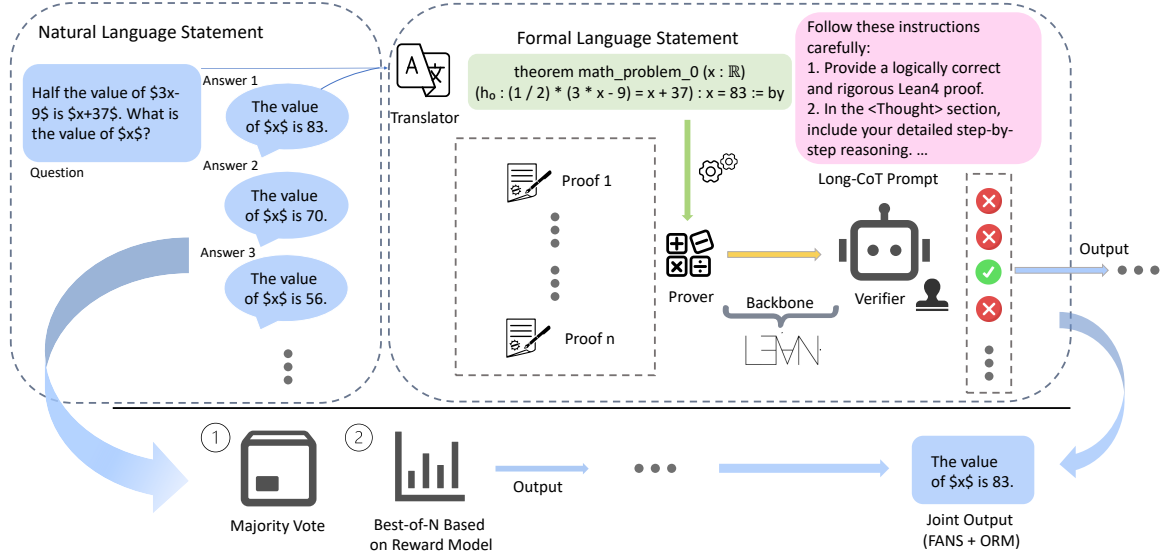


Figure 2: **FANS Framework:** The framework shown in the upper part first passes the Natural Language math questions and the LLM-generated answers to our Long CoT NL-to-FL translator. Subsequently, it invokes a prover to prove the translated Lean4 statements and uses the verifier to check whether the proofs are correct. The correct outputs are used for further answer selection as a verifiable foundation. **Existing Methods:** majority vote and best-of-N ranking based on reward models are shown in the lower part of the figure.

question-answer pairs into their corresponding Formal Language (FL) statements in Section 2.1. Subsequently, we detail how we write FL proofs and verify it using the Lean4 verifier in Section 2.2. Finally, we introduce further usage for the verified proofs in Section 2.3. The general idea for FANS is to bridge the gap between FL and NL using the verified FL proofs as a solid foundation for LLM’s NL reasoning.

2.1 From Natural Language to Formal Language

To obtain accurately aligned FL statements from the original NL question-answer pairs, we first train a Long Chain-of-Thought (Long-CoT) (OpenAI, 2024) translator using pair-wise NL-FL statements. We try to teach a prover to analyze the NL statement in its Long CoT and then translate it into the FL statement. However, since there is no Long CoT data available for NL-to-FL statement translation, inspired by transfer learning methods for Lean prove Long CoT in MA-LoT (Wang et al., 2025b), we introduce the following training method for FL statement translator training.

Firstly, we collect the NL-FL aligned statement translation data from Lean-Workbook (Ying et al., 2024). Based on such data, as versatile examples and use analysis-then-generate method inspired by

Wang et al. (2023), we generate NL statements for the DeepSeek-Prover-v1 dataset’s Lean4 theorem statement using Qwen-2.5-72B. Altogether, we obtain a dataset containing 162,181 NL-FL aligned theorem statements without Long CoT. Among them, 21,967 are generated from DeepSeek-v1 and 140,214 are from Lean-Workbook.

Based on the aligned dataset, we train a Long CoT translator using the LoT-Solver (Wang et al., 2025b) as the base model. Since there is no Long CoT annotated theorem statement translation dataset, we apply the transfer learning technique. During training, we explicitly instruct the model in the system prompt to answer without using Long CoT and provide the empty Long CoT content. During inference, we instruct the model to use Long CoT to answer the question. In the inference, the model learns from the basic Long CoT ability in LoT-Solver to accurately translate the NL statement into the FL statement. Details of training and inference data examples can be found in Appendix A.

Besides our own Long CoT translator, recently, Wang et al. (2025a) also released an NL-FL translator, and we also utilize it as the translator in the following experiments. Since it is not guaranteed that the translated FL statements are perfectly consistent with the original NL statements, we introduce

another procedure to check whether the translations are faithful by prompting the LLMs themselves or a stronger one, QwQ-32B (Qwen Team, 2025) in our case, to generate a decision.

2.2 Formal Language Proofs and Verification

After receiving the translated formal language statements, we utilize the open-sourced Lean4 provers (Lin et al., 2025; Ren et al., 2025) to produce potential proofs written in Lean4 as well. To adapt the prover on the formal language statements from natural language math problems, we use a few-shot prompt for better alignment with the formats of classic formal language proof problems. Since the natural language statements have been translated into the standard formal language statements format, provers fine-tuned on formal language statements-proofs could be immediately applied without any modification. For this very reason, FANS could generalize among different backbone provers easily, with a highly disentangled formulation of its three-stage procedure.

The core backbone for FANS to be rigorously verified stems from the formal proof process in the functional programming level, which could be implemented by retrieving the compilation results from the verifier. At its core, Lean4 operates within the calculus of inductive constructions, ensuring that every proof is mechanically verified through a type-checking system that enforces strict logical consistency. This could eliminate human errors commonly found in formal reasoning, getting rid of unstable reasoning based on intuition and vague derivation. Therefore, once a theorem is proven in Lean4, it is mathematically indisputable, providing a robust foundation for tasks such as answer selection in math problem solving.

2.3 Beyond Formal Language Itself

Though the provers could achieve a 60% to 80% success rate on standard formal language proofs datasets like miniF2F (Zheng et al., 2021), there are still chances that all the tries of formal proofs fail because of the gap between the formal language problems used to train the prover and the formats of translated formal language statements from the natural math QA questions. In this case, FANS will fall back to other alternate methods to select the best output. For example, we could turn to the vanilla majority vote, or take advantage of reward models and output the answer with the highest score. For the former, we first conduct a majority

vote on those answers successfully verified by the Lean4 verifier, and if there the number of votes for the winner is below one pre-specified threshold, we discard the answer and return to the majority vote directly to mitigate the impact of false positives due to mis-translation.

If the reward models are accessible, they could serve as a metric to determine which problems are more difficult by comparing the scores on specific questions and model outputs. Intuitively, harder problems might lead to more erroneous attempts, rendering the best-of-N based on reward models ineffective. Under this scenario, we resort to verifiable formal language proofs to accurately identify the correct answer among multiple wrong options.

3 Experiments

We conduct extensive experiments on the MATH500 (Hendrycks et al., 2021) and AMC23 (Yang et al., 2024b) to evaluate the effectiveness of FANS on enhancing the NL mathematical reasoning using FL methods. We quantitatively measure its usage in answer selection in Section 3.3 and qualitatively show its capability in enhancing the trustworthiness of NL math reasoning by providing a formal backbone in Section 3.5. Additionally, we conduct a thorough ablation study to validate the importance of each module of our proposed framework.

3.1 Experiment Setup

3.1.1 Dataset and Task

In this work, we evaluate FANS’s capability in enhancing LLMs’ NL reasoning on several challenging datasets. The first is MATH500 (Hendrycks et al., 2021), a dataset containing 500 high-school-level math problems ranging in 7 major fields, including precalculus, algebra, number theory, etc. Another dataset we use is AMC23 from the repo of Qwen-2.5-Math (Yang et al., 2024b). This dataset contains 40 high-school-level math competition questions, ranging from similar fields as MATH500 but harder and more versatile in the form of question types. Other datasets include Minerva Math (Lewkowycz et al., 2022) and Olympiad Bench (He et al., 2024).

The goal of FANS is to use the FL method to enhance the existing answer selection methods like Majority Vote (MV) and Optimized Reward Model (ORM), and use formal reasoning to provide verifiable support for NL reasoning.

Models	MATH(total)	MATH-Algebra	MATH-Number Theory	AMC23
Mistral-MV	33.80	42.74	33.87	12.50
Mistral-FANS	36.40	45.97	35.48	15.00
Improvement(%)	+7.69	+7.56	+4.75	+20.00
DeepSeek-Math-ORM	62.60	82.26	61.29	30.00
DeepSeek-Math-ORM+FANS	63.80	82.26	66.13	32.50
Improvement(%)	+1.91	-	+7.90	+8.33
Qwen-2.5-Math-ORM	80.80	96.77	91.94	70.00
Qwen-2.5-MATH-ORM+FANS	81.80	98.39	93.55	70.00
Improvement(%)	+1.25	+1.67	+1.75	-

Table 1: Accuracies on math benchmarks (%). For all baselines, we apply ORM@8 for comparison to using FANS based on ORM or using FANS compared with majority voting. The result for FANS is denoted as FANS-model.

Models	MATH500	Minerva Math	Olympiad Bench	AMC23	Average
Llama-3.2-3B-Instruct	56.60	24.26	21.93	35.00	34.45
FANS w/ self check	57.60	22.06	21.89	32.50	33.51
FANS w/ external check	61.00	24.26	24.15	35.00	36.10
FANS remove	61.80	27.21	25.48	40.00	38.62
Deepseek-Math-7B-Instruct	54.00	27.57	20.74	32.50	33.70
FANS w/ self check	55.40	26.47	22.37	27.50	32.94
FANS w/ external check	57.00	27.21	22.67	30.00	34.22
FANS remove	58.60	30.88	23.85	32.50	36.46
Qwen2.5-Math-1.5B-Instruct	79.00	33.46	44.30	55.00	52.94
FANS w/ self check	79.20	33.82	44.44	55.00	53.12
FANS w/ external check	80.00	33.82	44.74	57.50	54.02
FANS remove	80.20	36.40	45.63	60.00	55.56
Qwen2.5-Math-7B-Instruct	87.40	41.18	50.22	72.50	62.83
FANS w/ self check	87.40	40.07	50.52	72.50	62.62
FANS w/ external check	88.00	41.18	50.67	72.50	63.09
FANS remove	88.80	42.65	51.85	72.50	63.95

Table 2: Results (%) with new translator (Wang et al., 2025a) and current SOTA prover (Ren et al., 2025).

3.1.2 Baselines

To demonstrate how FANS works across diverse base models with different levels of answering mathematical questions, we select Mistral-7B (Jiang et al., 2023), DeepSeek-Math-7B (Shao et al., 2024) and Qwen-2.5-Math-1.5B / 7B (Yang et al., 2024b). The former two are relatively weaker models, leaving more space for them to be improved by FANS itself. In contrast, for the latter stronger models, which could answer more problems correctly, FANS majorly focus on providing trustworthiness and a verifiable foundation to the generated solutions. For ORM methods, we uniformly select Qwen-RM-72B because it is a larger model and has the best performance.

3.2 Implementation Details

In FANS, we use LoT-Solver as the base model for our Long CoT translator training using the transfer learning method we proposed. We train the translator on 162,181 records of NL-FL aligned statement data. To stabilize training, we also use block training and curriculum data sorting techniques in Wang et al. (2024b). Besides, Kimina-Autoformalizer (Wang et al., 2025a) is also tested as the translator. We use DeepSeek-Prover-v1.5 (Xin et al., 2024b), Goedel Prover (Lin et al., 2025) and DeepSeek-Prover-v2 (Ren et al., 2025) as our provers and Santos et al. (2025) as the verifier, which significantly reduces the verification overhead. We use NVIDIA H200 for model training and inference. Lean4 verification is conducted on CPUs.

3.3 Results

Table 1 demonstrates that FANS consistently improves the answer selection accuracy across all base models when compared to the baselines. On the two sub-fields, algebra and number theory, where lean4 performs better compared to other fields due to well-developed support from the language libraries, FANS could achieve an accuracy gain up to 7.90%. On stronger base models like Qwen-2.5-Math (Yang et al., 2024b), FANS increases the accuracy for answer selection by 1.75% as well. On the harder dataset AMC23, FANS helps weaker models to select the correct answer successfully, confirming that FANS working upon the verifiable proof process would be more helpful to identify the right answer from multiple wrong ones.

Table 2 displays the results with the translator from Wang et al. (2025a) and the current SOTA prover from Ren et al. (2025) at 7B scale. Here “FANS remove” means we remove all false positive translations by eliminating those items with incorrectly selected answers but successfully verified FL proofs. Therefore, it could be regarded as a kind of upper bound for FANS, while an exact one since, if the translations are correct, provers could prove more FL statements. To ensure better consistency and fidelity of translated FL statements with the original NL statements, we introduce an extra stage to check whether the translations are correct or not, by invoking the base models themselves, or an external stronger model, for example, QwQ-32B (Qwen Team, 2025) here.

After the verification of the translations from the external model, FANS could achieve uniformly better final performance compared to all baselines, while the verification from the same base models could also achieve slight improvements on some datasets. This implies the potential of integrating FL to assist the answer selection in LLMs’ math reasoning, especially by providing a verifiable backbone when the translation from natural language to formal language is faithful enough.

3.4 Ablation Studies

3.4.1 Dropping Long CoT in Translator

To validate the effectiveness of the Long CoT translator in FANS, we conduct the study on dropping the Long CoT translator and replacing it with other models. Choices include the fine-tuned translator without Long CoT and GPT-4o-mini. We use few-shot prompts for all models. The results are

Dataset	FANS (GPT-4o-mini)	FANS (w/o Long CoT)	FANS
Number Theory	30.65	37.10	37.10
Prealgebra	57.32	59.76	60.98
Inter Algebra	16.49	14.43	17.53
Algebra	46.77	48.39	50.81
Precalculus	14.29	21.43	19.64
MATH500-Full	33.80	35.40	37.40

Table 3: Comparison among different translators with Mistral-7B as the base model.

Dataset	ORM (Mistral)	ORM (Deepseek)	ORM (Qwen2.5 -Math)	ORM (Qwen2.5 -Math) + FANS
Precalculus	30.36	23.21	26.79	26.79
Prealgebra	64.63	65.84	73.17	73.17
Interalgebra	25.77	16.49	24.74	25.77
Algebra	56.45	52.42	60.48	60.48
Number Theory	32.26	30.65	40.32	43.55
MATH500	42.40	38.80	45.80	46.40

Table 4: Comparison among different reward models with Mistral-7B as the base model.

presented in Table 3. The results show that the accuracy for MATH500 without the Long CoT translator drops significantly, confirming the effectiveness of Long CoT in providing a more faithful translation. Comparison to the results on GPT-4o-mini shows that existing closed-source LLMs have sub-optimal performance on FL translation, aligned with previous studies (Wang et al., 2024b).

3.4.2 Using different reward models

To test the effectiveness of reward models in FANS, we conduct the ablation study on switching between different kinds of Reward models. In the experiment, we use different reward models to select data generated by Mistral-7B. The results are presented in Table 4. It indicates that the Mistral reward model has the best performance on Mistral generated data while other reward models all suffer from suboptimal performances. This experiment shows that reward models may be unable to generalize to OOD model or data. Indicating the need for our generalizable answer selection methods.

3.5 Qualitative Studies

A concrete example through FANS pipeline is displayed in 1. From the example, we could see that the translator translates the NL statement into its corresponding FL statement correctly, maintaining the original semantic meaning and mathematical formulation without inappropriate modification. In

the proof stage, the prover successfully proves the FL statement with detailed step-by-step explanations preceding each line, which are omitted here for brevity. In the example, the question asks for the solution to a system of equations. Since the original natural language statement is already in a quite standard math format, our translator simply transforms the natural language description to a formal language expression. The prover generates a rigorous proof for the statement, which is successfully verified by the verifier.

4 Related Work

Reasoning models, both proprietary like OpenAI-O1, Google Gemini Flash Thinking, Kimi-k1.5 (Kimi Team, 2025) and open-sourced ones such as Qwen Math (Yang et al., 2024b) and Deepseek-R1 (Xin et al., 2024a), begin to substitute general language models in the core of LLM research. This stems from not only the representative of (math) reasoning ability for evaluating the intelligence of LLMs but also the various downstream tasks it could be applied to and the potential it reveals about the underlying immense abilities of LLMs to solve complicated problems that can only be resolved by humans in the past. Zhou et al. (2024) proposed a similar framework to FANS, but their method is heavily prompt-based, built upon weaker base models, while we train another Long CoT translator and utilize more advanced models to generate proofs and translation verification.

4.1 Formal Language Reasoning

The Formal Languages (FL) for math reasoning express mathematical statements in verifiable first-order logic. By solving math problems using FL, we can not only verify the correctness of the problem by the final answer like MATH (Hendrycks et al., 2021) or GSM8k (Cobbe et al., 2021) but also explicitly verify the correctness of each intermediate steps, making math reasoning has a solid foundation. Typical FL are like Isabelle (Paulson, 1994), Coq (Coq, 1996), Metamath (Megill and Wheeler, 2019), HOL Light (Harrison, 2009), and Lean (De Moura et al., 2015; Moura and Ullrich, 2021). Following Yang et al. (2024c), we choose Lean4, the latest and most actively studied FL, as the language we use.

Traditional studies on FL all focused on how to annotate more data to boost the performance of LLMs on FL. Representative works like Lean-

Dojo (Yang et al., 2024c) use retrieval methods to select tactics; Wang et al. (2024b) tries to use LLM to translate NL proof to FL proof; MA-LoT (Wang et al., 2025b) proposed Long CoT and multi-agent framework to solve FL questions; DeepSeek-Prover (Xin et al., 2024b,a), Godel-Prover (Lin et al., 2025), and InternLM-Step-Prover (Wu et al., 2024b) applies massive data annotation to provide better foundation models. However, all of the above works focus on solving FL problems and ignore the potential of using FL to enhance the performance of NL math reasoning.

4.2 Natural Language Math Reasoning

Recent efforts to enhance the mathematical reasoning capabilities of large language models (LLMs) have spurred the development of diverse math problem-solving techniques. Many works in this field focus on developing advanced foundation models for solving math word problems such as DeepSeek-Math (Shao et al., 2024), Qwen-2.5-Math (Yang et al., 2024b), Mistral-Math (Yu et al., 2023), and Llemma (Azerbayev et al., 2023). Other focus on the inference methods to query the existing models to write better answers, typical methods like traditional Chain-of-Thought (CoT) (Wei et al., 2022; Yao et al., 2025; Xiong et al., 2025; Zhang et al., 2025a), tree-search methods (Yao et al., 2023; Jiang et al., 2024) and recent proposed Long-CoT approaches (OpenAI, 2024; Wang et al., 2025b; Open-Source-O1, 2024). However, none of them jump beyond the context of natural language in math reasoning, leading to LLMs’ math reasoning without formal foundations.

4.3 Reward Modeling

Reward models play a crucial role in assessing the quality of LLMs’ outputs, serving as an essential component in downstream applications such as reinforcement learning from human feedback (RLHF). These models function as evaluative mechanisms that guide optimization processes, ensuring that generated responses align with human preferences and ethical considerations. Several studies have incorporated reward models into their RLHF frameworks, highlighting their significance in fine-tuning LLMs (Dong et al., 2024; Bai et al., 2022; Ouyang et al., 2022; Dong et al., 2023; Wang et al., 2024a). Depending on the design of how to model the rewards, reward models can be categorized into both explicit and implicit frameworks. The former includes outcome-based or process-based ones

Demonstration of FANS

Natural Language Math Problem

Find the product CD of the integers C and D for which $\frac{C}{x-3} + \frac{D}{x+8} = \frac{4x-23}{x^2+5x-24}$ for all real values of x except -8 and 3 .

Answer -5 .

Translated Formal Language Statement

```
theorem math_problem_0 (C D : ℤ) (h : ∀ x : ℝ, x ≠ 3 ∧ x ≠ -8 → C / (x - 3) + D / (x + 8) = (4 * x - 23) / (x ^ 2 + 5 * x - 24)) : C * D = -5 := by
```

Formal Language Proof

```
theorem math_problem_0 (C D : ℤ) (h : ∀ x : ℝ, x ≠ 3 ∧ x ≠ -8 → C / (x - 3) + D / (x + 8) = (4 * x - 23) / (x ^ 2 + 5 * x - 24)) : C * D = -5 := by
  have h1 := h 0 (by norm_num)
  have h2 := h 1 (by norm_num)
  have h3 := h (-1) (by norm_num)
  field_simp at h1 h2 h3
  ring_nf at h1 h2 h3
  norm_cast at h1 h2 h3
  nlinarith
```

(Ma et al., 2023; Zhang et al., 2025b, 2024b,a; Li and Li, 2024) with each assessing only the final output or intermediate steps respectively. In contrast, implicit reward modeling bypasses the need for explicit reward function engineering by leveraging preference optimization techniques. Notable examples include direct preference optimization (DPO) (Rafailov et al., 2024) and other variants like KTO (Ethayarajh et al., 2024), SimPO (Meng et al., 2024), IPO (Azar et al., 2024), etc., all of which aim to refine LLM behavior without explicit reward function definition.

5 Conclusion and Discussion

Faced with the gap between existing answer selection methods and the upper bound from models' intrinsic abilities in the math problem-solving task, we propose FANS, a framework that introduces a formal language-based approach to perform better answer selection, offering a more rigorous and trustworthy alternative to heuristic methods like majority voting or best-of-N based on reward models. From the experiment results 1, we could see FANS indeed improve the accuracy for answer selection, achieving at most a relative improvement of 7.9% on MATH500 Number Theory and 20% on AMC23. This highlights the potential of utilizing formal language in enhancing answer selection for math problem-solving tasks. While it demonstrates notable improvements over baselines, there is still a gap between its current performance and the theoretical upper bound, i.e., pass @ N. Two primary directions for improvement stand out given the two-stage workflow of FANS.

Refining the NL-FL translation process is crucial to reducing false positives in theorem formulation. Errors in translation, such as misrepresenting an equation-solving problem as a tautology or mishandling extremum problems by misinterpreting constraints, often lead to vacuous proofs and incorrect answer selection. Enhancing the translator to preserve problem constraints accurately and explicitly consider optimization conditions can significantly improve formalization quality.

Enhancing the proving capabilities of the prover is necessary. Lean4's current package ecosystem is biased toward certain fields like algebra and number theory, limiting its applicability in others like geometry and combinatorics. Expanding its library support would enable broader coverage of mathematical domains. Additionally, while the Goedel-Prover (Lin et al., 2025) achieves state-of-the-art results on formal language benchmarks (Zheng et al., 2021; Ying et al., 2024), it still struggles with nearly 40% of proof problems. Strengthening its reasoning abilities through iterative fine-tuning and improved proof-search strategies could address this limitation.

Despite these challenges, the formal verification approach provides a trustworthy and interpretable alternative compared to conventional answer selection methods. By incorporating rigorously verifiable Lean4 deductions, FANS reduces dependence on extensive adaptation for different base models, ensuring consistent and scalable improvements in LLM-driven mathematical reasoning, shedding light on a more trustworthy way to better leverage the capability of existing models.

Limitations

One major limitation of using formal language for answer selection is its high false positive rate and the inherent incompleteness of provers, which prohibits successfully verifying all factually correct statements. Errors in formalization, such as incorrect theorem representations or inadequate constraint handling, further lead to misclassification. Additionally, the limited domain coverage of Lean 4 constrains its applicability to only a subset of mathematical fields.

Future work could focus on improving the robustness of NL-FL translation, ensuring accurate problem formulation, and enhancing the theorem-proving capabilities of provers to handle more complex and diverse mathematical problems. Expanding the formal proof ecosystem beyond algebra and number theory will be essential for broader applicability as well. We believe addressing these challenges will bring formal verification closer to a more reliable, scalable solution for mathematical reasoning in LLMs.

For potential risks, since this work focuses on the math reasoning task of LLMs, with no intersection with ethics or other societal questions, we believe it currently contains no such risk. Besides, AI assistants were only used for spelling and grammar checking during writing.

References

- Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi 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*, pages 4447–4455. PMLR.
- Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, and 1 others. 2024. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*.
- 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*.
- Projet Coq. 1996. The coq proof assistant-reference manual. *INRIA Rocquencourt and ENS Lyon, version*, 5.
- 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: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pages 378–388. Springer.
- Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.
- Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. 2024. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*.
- Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

- Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*.
- John Harrison. 2009. Hol light: An overview. In *International Conference on Theorem Proving in Higher Order Logics*, pages 60–66. Springer.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. [Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems](#). *Preprint*, arXiv:2402.14008.
- 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*.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.
- Albert Q Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. 2022. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283*.
- 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*.
- Kimi Team. 2025. Kimi k1.5: Scaling reinforcement learning with llms.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- 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.
- Wendi Li and Yixuan Li. 2024. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*.
- 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*.
- Haohan Lin, Zhiqing Sun, Yiming Yang, and Sean Welleck. 2024. Lean-star: Learning to interleave thinking and proving. *arXiv preprint arXiv:2407.10040*.
- Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. 2025. [Goedel-prover: A frontier model for open-source automated theorem proving](#). *Preprint*, arXiv:2502.07640.
- Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. 2023. Let’s reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*.
- Norman Megill and David A Wheeler. 2019. *Metamath: a computer language for mathematical proofs*. Lulu.com.
- Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.
- Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In *Automated Deduction—CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, pages 625–635. Springer.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. sl: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.
- Open-Source-O1. 2024. [Open-o1](#). Accessed: 2024-12-28.
- OpenAI. 2024. Learning to reason with llms. <https://openai.com/index/learning-to-reason-with-llms/>. Accessed: 2024-11-24.
- 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.
- Lawrence C Paulson. 1994. *Isabelle: A generic theorem prover*. Springer.
- Qwen Team. 2025. [Qwq-32b: Embracing the power of reinforcement learning](#).

- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.
- ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, and 1 others. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*.
- Marco Dos Santos, Haiming Wang, Hugues de Saxcé, Ran Wang, Mantas Baksys, Mert Unsal, Junqi Liu, Zhengying Liu, and Jia Li. 2025. [Kimina lean server: Technical report](#). *Preprint*, arXiv:2504.21230.
- Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Mehran Kazemi, Najoung Kim, and He He. 2023. Testing the general deductive reasoning capacity of large language models using ood examples. *Advances in Neural Information Processing Systems*, 36:3083–3105.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseek-math: 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*.
- Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, and 1 others. 2024a. Secrets of rlhf in large language models part ii: Reward modeling. *arXiv preprint arXiv:2401.06080*.
- Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, and 1 others. 2025a. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*.
- Ruida Wang, Rui Pan, Yuxin Li, Jipeng Zhang, Yizhen Jia, Shizhe Diao, Renjie Pi, Junjie Hu, and Tong Zhang. 2025b. Ma-lot: Multi-agent lean-based long chain-of-thought reasoning enhances formal theorem proving. *arXiv preprint arXiv:2503.03205*.
- Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong Zhang. 2024b. Theoremllama: Transforming general-purpose llms into lean4 experts. *arXiv preprint arXiv:2407.03203*.
- Ruida Wang, Wangchunshu Zhou, and Mrinmaya Sachan. 2023. Let’s synthesize step by step: Iterative dataset synthesis with large language models by extrapolating errors from small models. *arXiv preprint arXiv:2310.13671*.
- 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 its reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- 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, and 3 others. 2020. [Trans-formers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024a. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*.
- Zijian Wu, Suozhi Huang, Zhejian Zhou, Huaiyuan Ying, Jiayu Wang, Dahua Lin, and Kai Chen. 2024b. Internlm2. 5-step-prover: Advancing automated theorem proving via expert iteration on large-scale lean problems. *arXiv preprint arXiv:2410.15700*.
- Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024a. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*.
- Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, and 1 others. 2024b. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152*.
- Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, and 1 others. 2025. A minimalist approach to llm reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.
- Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J

Prenger, and Animashree Anandkumar. 2024c. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems*, 36.

Jiarui Yao, Yifan Hao, Hanning Zhang, Hanze Dong, Wei Xiong, Nan Jiang, and Tong Zhang. 2025. Optimizing chain-of-thought reasoners via gradient variance minimization in rejection sampling and rl. *arXiv preprint arXiv:2505.02391*.

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, 2023. URL <https://arxiv.org/pdf/2305.10601.pdf>.

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*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhengguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

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. *arXiv preprint arXiv:2406.03816*.

Hanning Zhang, Jiarui Yao, Chenlu Ye, Wei Xiong, and Tong Zhang. 2025a. Online-dpo-rl: Unlocking effective reasoning without the ppo overhead.

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

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

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

Jin Peng Zhou, Charles Staats, Wenda Li, Christian Szegedy, Kilian Q Weinberger, and Yuhuai Wu. 2024. Don't trust: Verify-grounding llm quantitative reasoning with autoformalization. *arXiv preprint arXiv:2403.18120*.

A Translator Training Example

This section gives examples of the training and inference method for using the transfer learning method to train the Long CoT capability of translator based on data without special Long CoT annotation.

The prompt for training is as follows:

```
[
{
  "role": "system",
  "content": "You are a helpful
    ↪ mathematical assistant
    ↪ specialized in formal language
    ↪ Lean4. Please translate the
    ↪ following natural language
    ↪ statement into Lean4 theorem
    ↪ statement. You are directly
    ↪ asked to provide the answer to
    ↪ the problem **WITHOUT** using
    ↪ the Long Chain of Thought."
},
{
  "role": "user",
  "content": "\"\"@ Natural language
    ↪ theorem statement: Let $a,b,c$
    ↪ be positive real numbers .Prove
    ↪ that $ \\frac{b+c}{\\sqrt{a^2 +
    ↪ 8bc}} + \\frac{c+a}{\\sqrt{b^2 +
    ↪ 8ca}} + \\frac{a+b}{\\sqrt{c^2 +
    ↪ 8ab}} \\geq 2. $

    @ Lean4 theorem statement:\""
},
{
  "role": "assistant",
  "content": "\"\"<Thought> The user ask
    ↪ not to solve with long CoT, so I
    ↪ will directly write the
    ↪ answer.</Thought>

  ```lean4
 theorem lean_workbook_0 (a b c : R)
 ↪ (ha : 0 < a) (hb : 0 < b) (hc :
 ↪ 0 < c) : (b + c) / Real.sqrt (a
 ↪ ^ 2 + 8 * b * c) + (c + a) /
 ↪ Real.sqrt (b ^ 2 + 8 * c * a) +
 ↪ (a + b) / Real.sqrt (c ^ 2 + 8 *
 ↪ a * b) >= 2 := by
  ```\""
}
]
```

We use system prompt and user input as the training input and the assistant part as the expected output for our model training. We can see that in the system prompt, we indicate the model not to use Long CoT and in training, we just put a dummy placeholder for the Long CoT.

The inference output is as follows:

===Model Input===

```
<|begin_of_sentence|>Follow these
  ↳ instructions carefully:
1. Provide a logically correct and
  ↳ rigorous translate of natural
  ↳ language theorem statement to Lean4
  ↳ statement.
2. In the <Thought> section, include
  ↳ your detailed step-by-step
  ↳ reasoning.
3. In the <Output> section, provide only
  ↳ the final Lean4 proof or final
  ↳ result.### Instruction:You will
  ↳ receive several Lean4 problems. For
  ↳ each:
- **Use** a step-by-step solution
  ↳ internally in <Thought>.
- **Do not** reveal your chain of
  ↳ thought outside the <Thought> block.
- **Ensure** the final Lean4 code or
  ↳ final result is placed only in
  ↳ <Output>.
@ Natural language theorem statement:
mathd_numbertheory_3
What is the units digit of the sum of the
  ↳ squares of the first nine positive
  ↳ integers? Show that it is 5.

@ Lean4 theorem statement:
### Response:
<Thought>
```

Alright, I should do the following:

1. Analysis the problem and understand
 - ↳ the requirements.
2. Use natural language to analysis
 - ↳ the potential Lean4 condition
 - ↳ and requirements in the natural
 - ↳ language theorem statement.
3. Write the translated Lean4
 - ↳ statement code.

===Model Output===

he `mathd_numbertheory_3` has the
 ↳ following conditions:

- The problem asks for the units digit
 - ↳ of the sum of the squares of the
 - ↳ first nine positive integers.
- The expected answer is 5.

So, the theorem should state that the
 ↳ units digit of the sum of the
 ↳ squares of the first nine positive
 ↳ integers is 5.

</Thought>

<Output>

```
```lean4
theorem mathd_numbertheory_3 : (Sum i in
 ↳ Finset.range 10, i ^ 2) % 10 = 5 :=
```&
</Output>
```

We can see that by using the system prompt to clearly instruct the model to use Long CoT in inference, the model apply its internal bearing Long CoT capability, together with the Lean4 knowledge and statement translation ability learned in transfer learning. The model can perform accurate translation from NL to FL using Long CoT inference.

B Experiments Details

B.1 Implementation Settings

For the answer generation, we use vLLM (Kwon et al., 2023) as the inference framework, with the temperature set to 0.6 to avoid too much repetition, and set the max tokens allowed to be generated as 4096. During the first stage of translation, we set the temperature of generation to 0, together with the number of max new tokens set to 2048, and use Transformers (Wolf et al., 2020) as the inference framework directly. For the prover, we use vLLM for inference as well and set the temperature to 1.0, with the max number of new tokens set to 2048.

For all experiment results, we evaluate our method FANS on four different base models, Mistral-7B (Jiang et al., 2023), DeepSeek-Math-7B-Instruct (Shao et al., 2024), Qwen-2.5-Math-1.5B-Instruct (Yang et al., 2024b), Qwen2-Math-7B-Instruct (Yang et al., 2024a), and the results could be found in table 5. We did not include Qwen-2.5-Math-7B-Instruct as one of our base models

Formal Language Translation Prompt without Natural Language Alignment

"{question} Show that it is: {answer}."

Figure 3: Formal language translation prompt without natural language alignment.

because during inference we found that it could easily generate nonsense outputs not related to the math problems until the inference budget is reached. This phenomenon persists no matter what inference framework we use. So as an alternative, we chose the 7B model from a version before. Overall, the performance of the latter two models is better than the former two. And FANS could achieve uniform performance gain among all different base models, with a larger margin on weaker base models.

The performance gain on stronger base models is smaller due to the fact that harder problems are more challenging to be translated into appropriate natural language statements with consistent meaning and math formulation, and they are hard to be proved automatically by the prover in stage two. In an ideal situation, if the success rates for NL-FL translation and prover are p and q respectively, and the accuracies for majority vote and pass @ n are r_1 and r_2 respectively, then the theoretical performance gain of FANS should be roughly $(r_2 - r_1) \cdot pq$. This leaves a huge space for future work to improve the current pipeline from both the point of views of translator and prover. With better translator achieving lower false positive rates, and / or better prover which could systematically prove harder formal language statements, FANS will be more effective and efficient benefiting from both aspects.

B.2 Effects of Natural Language Alignment

Without natural language statements alignment, we directly append the generated answers to the end of the questions and utilize the concatenated ones as instructions for the formal language translator. In this scenario, the prompt used for formal language translation is table 3.

However, since the translator model is mainly trained on the dataset in the aligned format, we conduct an ablation study on whether natural language statement alignment is helpful or not. We

use meta-llama/Llama-3.3-70B-Instruct¹ as the alignment model to transform the original natural language statements into the corresponding aligned format. During translation, we keep the temperature as zero, and set the max length of new tokens to 1024. To accelerate the inference, we utilize the package vLLM (Kwon et al., 2023) as our translation framework. The alignment prompt is shown as in Figure 4.

For a given math question answering problem in its original format, the alignment format will be in “Given ... (premises), show that ... (goal)”, or “If ... (premises), prove that ... (goal)” like standard formats. Table 5 demonstrates a concrete example of aligning the original natural language statement into the standard format. From the results in table 6, we may notice that whether to use natural language alignment does not affect the overall performance of FANS, so we omit this alignment step in the other experiments.

The prompt used for formal language translation without Long-CoT is shown in Figure 6, which takes advantage of the ability of LLMs’ in-context learning provided few-shot examples.

Concatenating the formal language statement and the proof header, together with the natural language statement as the auxiliary comment, we trigger the prover model and generate potential proofs for the given formal language statements.

Below we present more concrete examples from FANS, verifying that with the assistance of formal language, we could select the correct answer out of all possible candidates successfully.

¹<https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct>

Base Model	Method	Precalculus	Prealgebra	InterAlgebra	Algebra	Number Theory	MATH500	AMC23
Mistral-7B	Pass @ 1	0.1071	0.4954	0.1289	0.3579	0.2681	0.2785	0.0781
	Pass @ 8	0.3214	0.8049	0.3402	0.6532	0.4677	0.5180	0.3250
	Majority Vote @ 8	0.1786	0.5732	0.1546	0.4274	0.3387	0.3880	0.1250
	FANS @ 8	0.1786	0.6341	0.1753	0.4597	0.3548	0.3640	0.1500
Deepseek-Math-7B-Instruct	Pass @ 1	0.1696	0.6387	0.2023	0.5998	0.3609	0.4098	0.1687
	Pass @ 8	0.4107	0.8780	0.4742	0.8548	0.7258	0.6860	0.4500
	Majority Vote @ 8	0.1786	0.7195	0.3402	0.7177	0.5484	0.5200	0.2250
	FANS @ 8	0.1786	0.7195	0.3608	0.7339	0.5484	0.5240	0.2250
	ORM @ 8	0.3750	0.8049	0.4021	0.8226	0.6129	0.6260	0.3000
	ORM + FANS @ 8	0.3750	0.8171	0.4227	0.8226	0.6613	0.6380	0.3250
Qwen-2.5-Math-1.5B-Instruct	Pass @ 1	0.5558	0.7774	0.5296	0.8871	0.8165	0.7055	0.4938
	Pass @ 8	0.7500	0.8902	0.7526	0.9839	0.9839	0.8720	0.8000
	Majority Vote @ 8	0.6250	0.8415	0.5979	0.9435	0.8871	0.7740	0.6000
	FANS @ 8	0.6250	0.8659	0.5979	0.9435	0.8871	0.7760	0.6250
	ORM @ 8	0.6607	0.8780	0.6392	0.9677	0.9194	0.8080	0.7000
	ORM + FANS @ 8	0.6607	0.8780	0.6495	0.9839	0.9355	0.8180	0.7000
Qwen-2-Math-7B-Instruct	Pass @ 1	0.5446	0.8110	0.5206	0.8891	0.8145	0.7182	0.4781
	Pass @ 8	0.7321	0.9302	0.7320	0.9758	0.9677	0.8700	0.8500
	Majority Vote @ 8	0.6071	0.8780	0.5773	0.9355	0.9194	0.7840	0.5500
	FANS @ 8	0.5893	0.8659	0.5773	0.9355	0.9194	0.7760	0.5750
	ORM @ 8	0.6964	0.8902	0.6289	0.9597	0.9032	0.8180	0.7500
	ORM + FANS @ 8	0.6964	0.8902	0.6392	0.9597	0.9194	0.8220	0.7500

Table 5: Full experiments results.

Method	Precalculus	Prealgebra	InterAlgebra	Algebra	Number Theory	Geometry	Counting & Probability	MATH500
FANS @ 8	0.2143	0.5976	0.1443	0.4839	0.3710	0.2195	0.2632	0.354
FANS @ 8 aligned NL	0.1786	0.5244	0.1753	0.5081	0.3548	0.3171	0.2368	0.354

Table 6: Comparision between FANS with aligned natural language statement and original ones.

Natural Language Statements Alignment Prompt

Example list:

=====

```md

Let  $a, b, c > 0$ . Prove that:

$$\frac{a}{b+c} \sqrt{\frac{bc}{(b+a)(c+a)}} + \frac{b}{c+a} \sqrt{\frac{ca}{(c+b)(a+b)}} + \frac{c}{a+b} \sqrt{\frac{ab}{(a+c)(b+c)}} + \frac{2abc}{(a+b)(b+c)(c+a)} \leq 1$$

```

=====

```md

Let  $a, b, c \geq 0$  satisfy  $a + b + c = 3$ . Prove that

$$(a^2 + 2bc)(b^2 + 2ca)(c^2 + 2ab) \leq \frac{81}{8}(ab + bc + ca)$$

```

=====

```md

If  $x_1 + x_2 = -a$  and  $x_2 + x_3 = -b$ , then  $x_1 - x_3 = b - a$

```

=====

```md

If  $a, b, c \in [-k, k]$  where  $a, b, c, k$  are real numbers and  $a + b + c = 0$ , show that  $a^3 + b^3 + c^3 \leq \frac{3k^3}{4}$

```

=====

```md

If  $x, y, z > 0$  then prove:

$$x^2 + y^2 + z^2 - xy - yz - zx + \frac{3xyz}{x+y+z} \geq \sqrt[3]{x^2 y^2 z^2}$$

```

=====

Help me transform the following math problem with the given answer to a proof problem, and the language format of the question should be identical to the example list above, also in the “md” structure. Do not modify the given answer even if it is not correct, just transform the format.

Question

```md

Figure 4: The prompt used for natural language statements alignment.

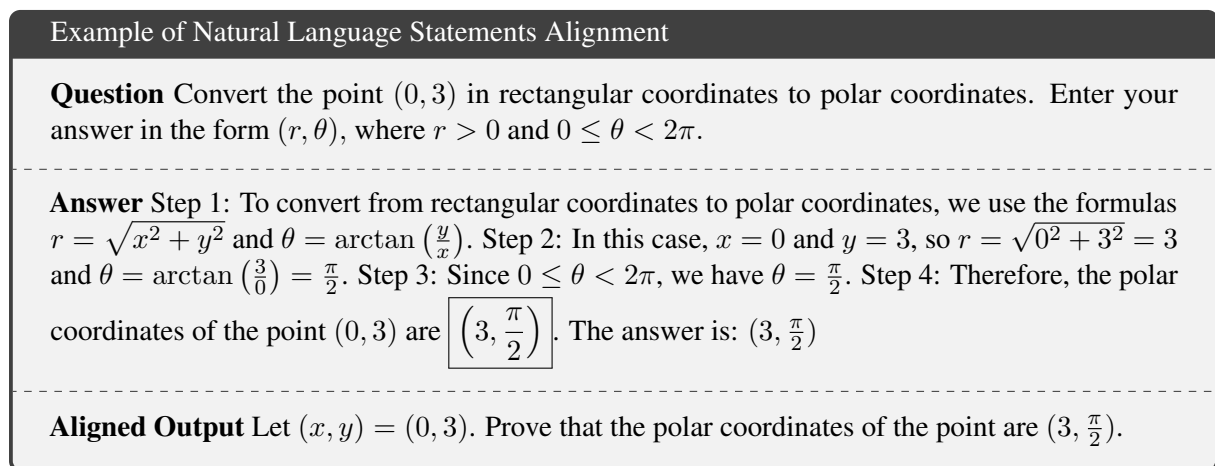


Figure 5: Example of the aligned natural language statement generated by the alignment model, given the original question and answer.

### Formal Language Translation Prompt without Long-CoT

Convert the following natural language mathematical statement into a corresponding Lean4 statement.

### Natural language statement:

Let  $a, b, c > 0$ . Prove that:

$$\frac{a}{b+c} \sqrt{\frac{bc}{(b+a)(c+a)}} + \frac{b}{c+a} \sqrt{\frac{ca}{(c+b)(a+b)}} + \frac{c}{a+b} \sqrt{\frac{ab}{(a+c)(b+c)}} + \frac{2abc}{(a+b)(b+c)(c+a)} \leq 1$$

### Lean4 statement:

```
``lean4
```

```
theorem lean_workbook_plus_49559 (a b c : ℝ) (ha : 0 < a) (hb : 0 < b) (hc : 0 < c) : (a / (b + c) * Real.sqrt (b * c / (b + a) * (c + a)) + b / (c + a) * Real.sqrt (c * a / (c + b) * (a + b)) + c / (a + b) * Real.sqrt (a * b / (a + c) * (b + c))) + 2 * a * b * c / (a + b) / (b + c) / (c + a) ≤ 1 := by
```

```
``&
```

=====

Convert the following natural language mathematical statement into a corresponding Lean4 statement.

### Natural language statement:

Let  $a, b, c \geq 0$  satisfy  $a + b + c = 3$ . Prove that

$$(a^2 + 2bc)(b^2 + 2ca)(c^2 + 2ab) \leq \frac{81}{8}(ab + bc + ca)$$

### Lean4 statement:

```
``lean4
```

```
theorem lean_workbook_plus_82625 (a b c : ℝ) (ha : a ≥ 0 ∧ b ≥ 0 ∧ c ≥ 0) (hab : a + b + c = 3) : (a^2 + 2 * b * c) * (b^2 + 2 * c * a) * (c^2 + 2 * a * b) ≤ 81 / 8 * (a * b + b * c + c * a) := by
```

```
``&
```

=====

Convert the following natural language mathematical statement into a corresponding Lean4 statement.

### Natural language statement:

{natural language statement to be translated}

### Lean4 statement:

```
``lean4
```

Figure 6: Formal language translation prompt without Long-CoT instruction.

### Formal Language Translation Prompt with Long-CoT

Follow these instructions carefully:

1. Provide a logically correct and rigorous translate of natural language theorem statement to Lean4 statement.
2. In the <Thought> section, include your detailed step-by-step reasoning.
3. In the <Output> section, provide only the final Lean4 proof or final result.

### Instruction: You will receive several Lean4 problems. For each:

- **Use** a step-by-step solution internally in <Thought>.
- **Do not** reveal your chain of thought outside the <Thought> block.
- **Ensure** the final Lean4 code or final result is placed **only** in <Output>.

@ Natural language theorem statement:

algebra\_sqineq\_2unitcircatblt1

Show that for any real numbers  $a$  and  $b$  such that  $a^2 + b^2 = 2$ ,  $ab \leq 1$ .

@ Lean4 theorem statement:

```
theorem algebra_sqineq_2unitcircatblt1 (a b : ℝ) (h₀ : a ^ 2 + b ^ 2 = 2) : a * b ≤ 1 :=
```

=====

@ Natural language theorem statement:

mathd\_numbertheory\_629

Suppose  $t$  is a positive integer such that  $\text{lcm}[12, t]^3 = (12t)^2$ . What is the smallest possible value for  $t$ ? Show that it is 18.

@ Lean4 theorem statement:

```
theorem mathd_numbertheory_629 : IsLeast { t : ℕ | 0 < t ∧ Nat.lcm 12 t ^ 3 = (12 * t) ^ 2 }
18 :=
```

=====

@ Natural language theorem statement:

{title} {NL statement}

@ Lean4 theorem statement:

### Response:

<Thought>

Alright, I should do the following:

1. Analyze the problem and understand the requirements.
2. Use natural language to analyze the potential Lean4 condition and requirements in the natural language theorem statement.
3. Write the translated Lean4 statement code.

The `{title}` has the following conditions

Figure 7: Formal Language Translation Prompt with Long-CoT

### Header of Formal Proof Prompt

Complete the following Lean 4 code with explanatory comments preceding each line of code:

```
```lean4
```

```
import Mathlib
```

```
import Aesop
```

```
set_option maxHeartbeats 0
```

```
open BigOperators Real Nat Topology Rat
```

Figure 8: Header of Formal Proof Prompt

More Concrete Examples for FANS Pipeline

Natural Language Math Problem

Find the value of $\log_2 3 \cdot \log_3 4 \cdot \log_4 5 \cdot \log_5 6 \cdot \log_6 7 \cdot \log_7 8$.

Answer 3

Translated Formal Language Statement

```
theorem math_problem_0 : Real.logb 2 3 * Real.logb 3 4 * Real.logb 4 5 * Real.logb 5 6 *  
  Real.logb 6 7 * Real.logb 7 8 = 3 := by
```

Formal Language Proof

```
theorem math_problem_0 : Real.logb 2 3 * Real.logb 3 4 * Real.logb 4 5 * Real.logb 5 6 *  
  Real.logb 6 7 * Real.logb 7 8 = 3 := by  
  simp [Real.logb, mul_assoc, mul_comm, mul_left_comm, show (3 : ℝ) = 3 by norm_num,  
    show (4 : ℝ) = 2 ^ 2 by norm_num, show (5 : ℝ) = 5 by norm_num, show (6 : ℝ) = 2 * 3 by  
    norm_num,  
    show (7 : ℝ) = 7 by norm_num, show (8 : ℝ) = 2 ^ 3 by norm_num]  
  field_simp  
  ring_nf
```

Natural Language Math Problem

A steel sphere with a 3-inch radius is made by removing metal from the corners of a cube that has the shortest possible side lengths. How many cubic inches are in the volume of the cube?

Answer 216

Translated Formal Language Statement

```
theorem math_problem_0 (r : ℝ) (h₀ : r = 3) : let cube_side := 2 * r; cube_side ^ 3 = 216 :=  
  by
```

Formal Language Proof

```
theorem math_problem_0 (r : ℝ) (h₀ : r = 3) : let cube_side := 2 * r; cube_side ^ 3 = 216 :=  
  by  
    subst h₀  
    simp only [pow_three]  
    norm_num
```

Figure 9: More concrete examples that demonstrate formal language could indeed translate the natural language statement correctly and then prove it successfully.