LEARNING TO GENERATE FORMALLY VERIFIABLE STEP-BY-STEP LOGIC REASONING VIA STRUCTURED FORMAL INTERMEDIARIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have recently demonstrated impressive performance on complex, multi-step reasoning tasks, especially when post-trained with outcome-rewarded reinforcement learning (Guo et al., 2025). However, it has been observed that outcome rewards often overlook flawed intermediate steps, leading to unreliable reasoning steps even when final answers are correct. To address this unreliable reasoning, we propose *PRoSFI* (*Process Reward over Structured Formal Intermediates*), a novel reward method that enhances reasoning reliability without compromising accuracy. Instead of generating formal proofs directly, which is rarely accomplishable for a modest-sized (7B) model, the model outputs structured intermediate steps aligned with its natural language reasoning. Each step is then verified by a formal prover. Only fully validated reasoning chains receive high rewards. The integration of formal verification guides the model towards generating step-by-step machine-checkable proofs, thereby yielding more credible final answers. *PRoSFI* offers a simple and effective approach to training trustworthy reasoning models.

1 INTRODUCTION

Recent advances in techniques such as Chain-of-Thought (CoT) prompting and Reinforcement Learning (RL) have significantly enhanced the reasoning capabilities of large language models (LLMs). These methods have improved both the length and quality of model-generated reasoning chains, giving rise to the paradigm of test-time scaling (Snell et al., 2025). Representative systems include OpenAI o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025), which set new records across multiple reasoning benchmarks.

Notably, DeepSeek-R1-Zero (Guo et al., 2025), which employs a rule-based reward mechanism (outcome correctness plus output format score) to guide learning via the Group Relative Policy Optimization (GRPO) algorithm, successfully achieves strong multi-step reasoning capabilities and exhibits emergent reflection behaviors. The simplicity and efficiency of this approach have inspired a series of follow-up works, such as R1-Zero replications TinyZero (Pan et al., 2025) and Logic-RL (Xie et al., 2025b), and various GRPO variants (Yu et al., 2025; Liu et al., 2025).

However, although these approaches greatly improve the final accuracy, they do not guarantee the correctness of the reasoning steps. In mathematical tasks, it has been observed that models may arrive at correct answers through invalid arguments (Uesato et al., 2022; Lightman et al., 2024). Recent formal audits make this gap concrete: Cao et al. (2026) report that 39.3% of intermediate steps are formally disproved even within reasoning chains that reach the correct answer. Helff et al. (2026) further show that verifiers checking only final-answer correctness actively incentivize shortcut behaviour in RLVR-trained models. Empirical studies, including our own observations, show that RL guided by outcome-only rewards can degrade the faithfulness of reasoning chains (Chen et al., 2025b). Furthermore, in domains like proof generation, where final results cannot be automatically validated, outcome-based rewards are inapplicable. To ensure trustworthy reasoning, process-level supervision, not just outcome supervision, is necessary.

Process-level supervision provides feedback at each chain-of-thought step, yielding more precise training signals and producing models with more interpretable, reliable reasoning steps (Uesato et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

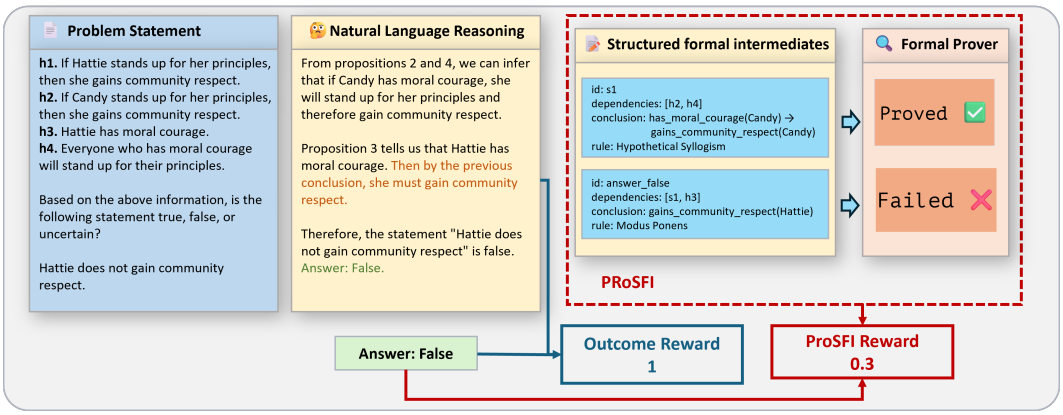


Figure 1: **Pipeline for *Process Reward over Structured Formal Intermediates* and comparison with Outcome Reward.** The process begins with a problem statement (left block) containing multiple propositions (h1–h4) related to individuals (Hattie and Candy). Then, a large language model (LLM) first generates natural language reasoning steps to solve the problem (middle block). The **Outcome Reward** approach then directly compares the generated answer with the ground truth, assigning a reward of 1 if they match. In contrast, *PRoSFI* introduces an additional step by generating structured formal intermediates that capture logical dependencies, inferred conclusions, and the application of formal inference rules such as *Hypothetical Syllogism* and *Modus Ponens*. These formal representations are then submitted to a **Formal Prover**, which verifies the logical soundness of the reasoning. In the illustrated example, although the generated answer ("Hattie does not gain community respect" is false) is correct, the formal proof fails verification. As a result, *PRoSFI* assigns a low reward despite the correct outcome.

2022). Nonetheless, collecting large-scale process-level feedback is challenging as human annotation at the granularity of individual proof steps is prohibitively expensive (Lightman et al., 2024). A natural solution is to leverage formal provers to verify intermediate reasoning steps. This approach requires access to formal codes, either directly generated by language models or translated from their natural language reasoning steps, which poses two key challenges.

First, models that excel in natural language reasoning often struggle to generate correct formal code. Training models to write formal proofs typically requires much larger model sizes and significant data distillation (Wang et al., 2025; Ren et al., 2025). Moreover, models optimized for formal proof generation often struggle with structure and format following (Xin et al., 2025). Consequently, it is difficult to have a model capable of generating both natural language explanations and aligned formal proofs simultaneously.

Second, existing models that excel at formal proof generation typically are not designed to take natural language inputs, and there is a lack of reliable methods for translating natural language into formal code, as well as for evaluating the faithfulness of such translations. Furthermore, generating formal code is inherently more challenging for large language models than producing natural language reasoning. Accurate formal code generation often relies on test-time scaling techniques like Best-of-N (BoN), which can significantly increase computational costs particularly during RL post-training.

To address these challenges, we propose a novel and effective reward method: *Process Reward over Structured Formal Intermediates* (*PRoSFI*, see Figure 1). Instead of generating full formal proofs directly, *PRoSFI* asks the model to output a series of reasoning blocks where each block contains a natural language explanation and a corresponding structured intermediate representation. This intermediate representation takes the form of a formatted array, usually in JSON or YAML, including an identifier, a logical rule, a derived conclusion (in formal language), and a list of dependencies (prior steps or premises). Then, formal propositions are extracted from each intermediate representation and passed to a formal prover for verification. The reasoning chain is considered trustworthy and receives the highest reward only if all steps are successfully verified.

Specifically, *PRoSFI* offers the following advantages. First, generating intermediate representations is easy. Existing models, even modest-sized (7B) language models, are typically well-equipped

for generating structured intermediate representations, and in this case, they only need to produce formalized propositions for each intermediate step—without having to construct full formal proofs. This avoids the core challenges of formal proof construction. Second, structured formal intermediates facilitate formal verification. Specifically, formal provers are tasked with verifying individual propositions derived from the intermediate representation, rather than constructing complete proofs for the original problems. This ensures step-level alignment with the intermediate reasoning. Overall, the structured formal intermediates serve as an effective bridge between language models and formal provers. They reduce the burden on both sides: language models are freed from producing formal syntax, avoiding distribution drift, while provers benefit from simplified sub-problems defined in a detailed and verifiable format.

To evaluate the effectiveness of our method, we conduct controlled experiments in a logical reasoning setting. We use the ProverQA dataset (Qi et al., 2025), which is hard for current models and, as a synthetic dataset, provides high-quality formal representations of the problems. Through extensive experiments, we demonstrate that *PRoSFI* can significantly improve the reasoning soundness for complex logical problems. We summarize our contributions as follows:

- **Conceptually,** we show that formal process rewards do not require the model to produce complete formal proofs: constraining it to emit structured formal intermediates makes prover feedback usable in RL, where direct formal-proof generation yields almost no signal.
- **Technically,** we propose *PRoSFI*, a framework that effectively bridges language models and formal provers. It enables RL post-training by incorporating formal verification feedback into policy optimization.
- **Empirically,** we demonstrate that Qwen2.5-7B-Instruct, after *PRoSFI* RL post-training, can generate verifiable proofs on hard first-order logic tasks, achieving this without relying on massive LLMs or extensive data distillation, marking the first such result at this scale.

2 Related Work

Reinforcement Learning Post-Training has recently played a crucial role in training large language models, particularly for human preference alignment (RLHF) and enhancing complex reasoning abilities. Early work such as InstructGPT (Ouyang et al., 2022) leveraged Proximal Policy Optimization (PPO) (Schulman et al., 2017) to extract reward signals from human preference data, steering the model toward outputs more aligned with user expectations. However, the high computational cost of PPO on large-scale models motivated the development of more efficient alternatives.

One such alternative is Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which has been adopted in training DeepSeek-R1-Zero (Guo et al., 2025). DeepSeek-R1-Zero trains a pretrained model by using GRPO with only rule-based rewards to achieve significant gains on complex mathematical reasoning tasks. Since then, many replication projects have become public Pan et al. (2025).

Logical Reasoning Datasets serve as important benchmarks for evaluating the deductive and symbolic reasoning capabilities of LLMs. These datasets are typically constructed either by manual annotation or via automatic generation. Manually curated collections, such as Reclor (Yu et al., 2020) and FOLIO (Han et al., 2024), offer rich linguistic diversity and complex logical rules. Automatic dataset generation methods have been proposed. ProntoQA (Saparov & He, 2022) and ProofWriter (Tafjord et al., 2020) use predefined templates or rule-based systems to generate large-scale reasoning problems with controllable complexity.

Other synthetic logic-reasoning datasets also include LogicPro (Jiang et al., 2025), which converts LeetCode program problems into narrative reasoning problems and leverages Python execution to extract and verify detailed intermediate solution traces; LogicBench (Parmar et al., 2024), a natural language inference benchmark that covers propositional, first-order, and non-monotonic logics; and FLD (Morishita et al., 2023; 2024), which employs a set of first-order logic rules to generate synthetic multistep proofs.

Formal-Language-Augmented Reasoning Techniques have recently attracted great attention to improve language models' rigor and reliability. In logical reasoning tasks, systems such as LINC (Olausson et al., 2023) and Logic-LM (Pan et al., 2023) treat the LLM as a semantic parser, translating

natural language premises and conclusions into first-order logic expressions. These expressions are then checked by external solvers like Prover9 (McCune, 2005) or Z3 (De Moura & Bjørner, 2008). Logic-LM further incorporates a self-correction loop, using solver error feedback to iteratively refine the symbolic translations until they pass validation. Similarly, Logic Agent (Liu et al., 2024) systematically applies propositional logic rules to structure and verify the model's reasoning, enhancing both effectiveness and interpretability. Other related work on logical and program-aided reasoning includes (Gao et al., 2023; Lightman et al., 2024; Morishita et al., 2024; Parmar et al., 2024; Qi et al., 2025; Ryu et al., 2025; Xu et al., 2024; Yang et al., 2023; Zhou et al., 2024).

For automated theorem proving, OpenAI's GPT-f system (Polu & Sutskever, 2020) first demonstrated that LLMs can generate formal mathematical proofs, achieving a 31% success rate on the miniF2F benchmark (Zheng et al., 2022). More recent advances along this direction include DeepSeek-Prover-V1.5 (Xin et al., 2025), Kimina-Prover (Wang et al., 2025), DeepSeek-Prover-V2 (Ren et al., 2025), and Seed-Prover (Chen et al., 2025a). Delta-Prover (Zhou et al., 2025) further leverages the sketching and reflection capabilities of general-purpose LLMs, solving 95.9% of the MiniF2F problems through decomposition and iterative reflection.

In mathematical reasoning, recent work has begun to integrate provers into LLM reasoning: Don't Trust: Verify (DTV) framework (Zhou et al., 2024) focuses on quantitative solutions by instructing LLMs to formalize their natural language steps into Isabelle (Paulson, 1994) proof scripts, which are then automatically verified. On benchmarks such as GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), DTV outperforms naive majority voting by filtering out unsound solution paths, demonstrating that formal provers can supply valuable feedback for reasoning paths.

Process Reward Models and Verification. Early work (Uesato et al., 2022; Lightman et al., 2024; Wang et al., 2024) train process reward models (PRMs) to provide fine-grained feedback for reasoning steps, achieving higher accuracy on GSM8K and MATH benchmarks. Recent research has shifted towards using LLMs as verifiers to detect errors in the reasoning process. For example, Huang & Yang (2025) employed a general-purpose model with a verification-refinement pipeline to achieve a gold medal in IMO 2025. Closest to our setting, LogicReward (Xu et al., 2026) also derives step-level rewards from a theorem prover, formalizing steps into Isabelle/HOL with a soft-unification mechanism that tolerates inconsistent naming; *PRoSFI* instead constrains the model to emit lightweight structured intermediates, which keeps each verification obligation small enough for a 7B model to satisfy without producing a complete formal script. VPRMs (Pronesti et al., 2026) likewise replace neural judges with deterministic rule-based verifiers, but in a domain whose decision paths are fixed by guidelines rather than by first-order logic, while FaithRL (Wang et al., 2026) targets step-level faithfulness with a learned process reward model and therefore obtains no formal guarantee.

3 Preliminaries on Dataset, Base Model, Algorithms and Evaluation Metrics

Dataset Description ProverQA (Qi et al., 2025) is a synthetic first-order logic reasoning benchmark generated by the ProverGen framework, which combines the linguistic diversity of LLMs with the rigor of a symbolic prover (Prover9 (McCune, 2005)). More details are provided in Appendix B.

Model Description The Qwen2.5-7B-Instruct model, an instruction-tuned variant of the Qwen2.5 family developed by Alibaba's Qwen team, comprises 7 billion parameters and is released under the Apache License 2.0 (Qwen Team, 2024). It builds on the Qwen2.5 base LLM with further fine-tuning to follow instructions and engage in chat/dialogue. Specifically, it can produce reliable structured outputs (e.g. JSON), parses tables, spreadsheets, and other semi-structured formats robustly.

Table 1: Evaluation metrics and their definitions.

Metric	Definition
Answer Correct Rate	$\frac{\#\text{Final Answer Correct}}{\#\text{Total Examples}}$
Reward Hit Rate	$\frac{\#\text{Outputs with Max Reward}}{\#\text{Total Examples}}$
Soundness	$\frac{\#\text{Answer Correct \& Path Valid}}{\#\text{Total Examples}}$

Group Relative Policy Optimization (GRPO)
GRPO is a variant of Proximal Policy Optimization (PPO) tailored for training large language models (LLMs) without an explicit value (critic) model (Shao et al., 2024). The algorithm details can be found in Appendix C.

Metrics We introduce several metrics to evaluate the various facets of the model performance: *Answer Correct Rate*, *Reward Hit Rate* and *Soundness*.

Answer Correct Rate measures the percentage of examples where the model produces the correct final answer, regardless of the validity of the reasoning steps. *Reward Hit Rate* reflects how often the model receives the maximum reward ($R = 1$) under our reinforcement learning framework, that is, when the final answer is correct and the structured reasoning trace is verified with respect to its declared dependencies. *Soundness* assesses both answer correctness and reasoning path validity, which is defined as the proportion of examples where the final answer is correct and the reasoning path is logically valid.

We note that *Reward Hit*, i.e., all structured formal intermediates are verified, does not necessarily guarantee the *Soundness* of the entire reasoning path. This is because the extracted formal propositions may not fully cover the entire reasoning path. In practice, rigorously assessing the soundness of a reasoning path would require extensive human evaluation, which is often infeasible at scale. As an alternative, we adopt state-of-the-art LLM-based evaluators, such as GPT-OSS-120B, to approximate *Soundness*. In the experimental section, we will analyze potential biases introduced by these LLM-as-a-Judge metrics. This caution is warranted: in a large-scale audit of 21 judge models, Norman et al. (2026) find that exact-match agreement overstates judge quality by 33–41 percentage points relative to chance-corrected agreement. With the dataset, model, algorithm, and evaluation metrics defined, we now explore how to train LLMs to generate step-by-step reasoning that is formally verifiable.

4 METHOD: *Process Reward over Structured Formal Intermediates*

To incorporate step-level verification into the reinforcement learning process, a natural approach is to (1) ask for the model to emit natural language reasoning step and formal proof script at the same time (e.g., in Lean 4 (Moura & Ullrich, 2021)), and (2) use a theorem prover to verify the correctness of each step. The verification outcomes can then be used as reward signals to distinguish valid reasoning chains from unsound ones.

Direct generation of stepwise formal proofs fails for modest models. A direct approach to formal process supervision is to generate natural-language reasoning alongside Lean 4 proofs and use verification outcomes as rewards. However, our 7B models rarely produced compilable proofs aligned with their reasoning, limiting useful supervision (Section 5.2). PRoSFI instead uses structured formal intermediates to express individual verification obligations without requiring complete proof scripts. Verification takes place after generation, decoupling prover calls from autoregressive decoding.

4.1 THE DESIGN OF *Process Reward over Structured Formal Intermediates*

To overcome the challenges encountered in directly generating formal proofs, we propose a novel approach: *Process Reward over Structured Formal Intermediates* (*PRoSFI*), which introduces a structured JSON/YAML based intermediate representation as a bridge between natural language reasoning paths and formal verification. This format serves as a lightweight, verifiable scaffold that enables formal reward during reinforcement learning. Crucially, Qwen2.5-7B-Instruct, the base model, is well-suited for this task due to its strong formatting capabilities, particularly for structured JSON or YAML outputs (Qwen Team, 2024).

Structured Formal Intermediates In *PRoSFI*, the model is prompted to produce a sequence of reasoning steps in structured format, where each step explicitly encodes the structure of logical reasoning, which includes the following fields: `id`: a unique identifier for the step, used for reference; `dependencies`: a list of premises this step depends on, which may include given conditions or previous steps; `conclusion`: The new conclusion drawn in this step, expressed in a formal language; `rule`: The logical rule applied to derive the conclusion.

To ensure each step remains atomic, we impose a constraint that the number of dependencies per step must be fewer than 5 (tunable). Each step can then be converted into a sub-problem for verification. Moreover, we also require that the final step's conclusion match the target proposition, and that its identifier be among the provided answer candidates.

This structured format allows natural language reasoning to be decomposed into discrete, formalizable sub-problems. For the first-order logic (FOL) tasks considered in this paper, which are relatively simple and well-structured, each proposition can typically be verified using lightweight tactics in Lean (e.g., `aesop, simp`), or through external automated solvers such as Z3 (De Moura & Bjørner, 2008) or Prover9 (McCune, 2005). For more complex settings, we optionally delegate verification to powerful formal models such as DeepSeek-Prover-V1.5 (Xin et al., 2025). These configurations can be adapted with different formal formats or verification tools for different tasks and domains.

Reward Design If all steps pass formal verification, the entire reasoning path is assigned a high reward $R = 1$; otherwise, the failure of any individual step flags a potential flaw in the reasoning process, and the path receives a reduced reward. These values are heuristic: they are chosen only to impose the ordinal preference *verified correct* $>$ *correct but unverified* $>$ *well-formatted but incorrect* $>$ *invalid*. Here 0.3 retains a weaker positive signal for a correct final answer whose verification fails, and 0.1 matches the small format-compliance reward used by the `Outcome-CoT` baseline without letting formatting compete with answer correctness. Because GRPO forms group-relative advantages, the ordering and separation of these rewards matter more than their absolute scale, and we do not claim these particular constants are optimal. Mathematically, the reward function is as follows:

$$R = \begin{cases} 1.0, & \text{Answer correct and all steps verified,} \\ 0.3, & \text{Answer correct, but some steps not verified,} \\ 0.1, & \text{Output format correct, but answer incorrect,} \\ 0.0, & \text{Output format incorrect or other failure cases.} \end{cases}$$

This reward design of ***PRoSFI*** exploits the rigor of formal provers while avoiding the difficulties of generating full formal proofs, resulting in more stable, fine-grained, and structure-aligned supervision signals, which are crucial for effective reinforcement learning. Moreover, ***PRoSFI*** avoids the efficiency bottlenecks typically associated with integrating formal provers into the rollout phase of RL training. The model generates the full natural language reasoning and the structured formal intermediates in a single, uninterrupted forward pass. Formal verification is performed after generation, allowing it to be fully decoupled from the autoregressive process. This decoupling not only improves training throughput by enabling batch-level parallelism but also allows verification tools to run asynchronously, maximizing efficiency. We next present empirical results to demonstrate that ***PRoSFI*** can effectively improve the soundness of LLM reasoning paths.

5 Experimental Results

Training Setup As described in Section 3, our experiments are conducted on the ProverQA dataset, a synthetic logical reasoning benchmark designed with controlled difficulty levels and well-defined ground-truth labels (True/False/Uncertain). This makes it particularly well-suited for evaluating models in a controlled reinforcement learning (RL) setting. To address the difficulty imbalance within the ProverQA dataset, we excluded instances labeled as Uncertain. The remaining data were then partitioned into training and test sets for each difficulty level (easy, medium, and hard), yielding a total of 769 instances for the training set.

We adopt Qwen2.5-7B-Instruct as our base model. This model offers a good choice for strong performance given the model size under our hardware constraints. Moreover, it has demonstrated superior responsiveness to RL-based post-training compared to other open-source alternatives such as the LLaMA series Shao et al. (2025); Zeng et al. (2025).

We use the GRPO algorithm with a learning rate of 3×10^{-7} and batch size of 64. For each training iteration, GRPO samples 8 responses, with the maximum sequence length set to 5120 tokens. The KL coefficient in the GRPO loss function is set to 0.001. To enhance training stability, we implement the clip-higher strategy from the Decoupled Advantage Policy Optimization (DAPO) algorithm (Yu et al., 2025), whose details are in Appendix C.

All experiments are conducted on a compute node equipped with 8× Hopper GPUs, running on Debian GNU/Linux 12 with CUDA 12.4, PyTorch 2.6.0, vLLM 0.8.3, and Verl 0.4.0. Each training run consists of 960 iterations, which takes approximately 14 hours to complete.

Testing Setup Besides the test splits of difficulty levels in the ProverQA dataset, we introduce a more challenging out-of-distribution (OOD) test set, named ProverQA-Extra. This set consists of 128 synthesized examples generated from the ProverQA repository using Qwen2.5-32B-Instruct as the background story generation model. To reduce randomness, we sampled 16 responses for each test data instance and took the average of their results.

5.1 Outcome Reward Leads to High Answer Correct Rate But Low Soundness

We employ GRPO with a reward scheme similar to that of DeepSeek-R1-Zero to replicate its training methodology on ProverQA. This serves a baseline for the evaluation of reasoning path soundness.

Reward Design Following the DeepSeek-R1-Zero pipeline, the model must emit its chain of thought wrapped in `<think> ... </think>` tags before the final answer, referred to as *format correct*. The outcome-based reward R is given in Section 4.1. This reward encourages adherence to the output format and correctness of the final answer.

Prompt Design The system prompt, adapted from DeepSeek-R1-Zero, is provided in Appendix D. The user prompt starts with the original dataset description, then presents the problem in formal language with clearly indexed premises (see Figure 1). We refer to this setup as *Outcome Reward with CoT* or *Outcome-CoT* for short. The full user prompt is presented in Appendix D.

Results: GRPO with *Outcome-CoT* significantly boosts the Answer Correct Rate but achieves rather low Soundness. Specifically, in our evaluation on the ProverQA-Hard subset, GRPO with *Outcome-CoT* substantially improves the Answer Correct Rate, rising from 46.0% with Qwen2.5-7B-Instruct to 91.31%. This confirms the successful replication of the R1-Zero training procedure. However, the GPT-judged Soundness score remains low, improving only from 9.0% to 21.97%. This indicates that although the final answers are often correct, the generated reasoning paths frequently contain flawed or invalid inference steps.

Table 2: Results on ProverQA Hard and Extra. GPT-judged Soundness values include the difference of each method versus the Outcome-CoT baseline.

Method	Metric (%)	Hard	Extra
Outcome-CoT	Answer Correct Rate	91.31	86.96
	GPT-judged Soundness	21.97	11.87
Direct Lean Generation	Answer Correct Rate	92.29	83.15
	GPT-judged Soundness	33.11 ↑ 11.1	10.45 ↓ −1.4
ProSFI	Answer Correct Rate	92.97	91.94
	GPT-judged Soundness	76.07 ↑ 54.1	56.10 ↑ 44.2

5.2 Direct Lean Generation Fails to Provide Reliable Stepwise Verification

As a direct baseline for integrating formal verification into reinforcement learning, we evaluated an approach in which the model is required to emit complete Lean 4 proof scripts (rather than the structured intermediates introduced in *PRoSFI*). This baseline was designed to isolate the effect of output modality: reward schedule and training protocol were kept the same as in the structured-intermediate condition, and the only difference was that the model produced full formal proofs immediately following each natural-language reasoning trace.

Empirically, this direct-generation strategy failed to provide a useful stepwise verification signal for GRPO. Compile-and-verify rates were low: the Reward Hit Rate on ProverQA-Hard was 13.77% and on ProverQA-Extra was 4.35%. Instances that did compile were vanishingly rare, and when compilation succeeded the resulting proofs typically reduced to a single opaque tactic (e.g. a one-line `simp` invocation) that bore little structural correspondence to the model's natural-language chain of thought. Consequently, the direct-Lean baseline produced almost no reliable, fine-grained supervision: both final-answer accuracy and measured reasoning soundness decreased relative to the Outcome-CoT baseline on the Extra set, as shown in Table 2.

5.3 *PRoSFI* Learning with ProverQA

Results: Structured formal rewards improve final accuracy with significantly more credible reasoning path. After GRPO training with stepwise formal rewards, we achieve comparable final-answer accuracy to the GRPO training with only final-answer reward on ProverQA, as shown in Table 2. This confirms that our stepwise rewards can effectively be combined into R1-Zero reward with high reasoning accuracy. We can also see that stepwise formal rewards can significantly improve the credibility of reasoning paths in comparison with final-answer only reward in Section 5.1. Moreover, the results on ProverQA-Extra also demonstrates *PRoSFI* achieves strong out-of-distribution (OOD) performance on logical reasoning questions of previously unseen difficulty.

5.4 Test-Time Scaling Behaviors of *PRoSFI*

A critical advantage of *PRoSFI* is its strong *test-time scaling* behavior, made possible through integration with the *Don't Trust; Verify* (DTV) method (Zhou et al., 2024). DTV is designed to enhance both accuracy and reasoning reliability by sampling multiple reasoning paths and employing a lightweight verifier to assess the logical validity of each output. If any candidates are successfully verified, one of them is selected; otherwise, the method defaults to majority voting. *PRoSFI* is particularly well-suited for integration with DTV due to its structured intermediate representations, which enable reliable formal verification. In contrast, approaches that rely solely on natural language reasoning lack the structured outputs required for effective collaboration with DTV, limiting their ability to benefit from test-time verification.

In experiments, we apply the DTV method to the outputs generated by *PRoSFI*. As shown in Figure 2, increasing the number of sampled reasoning paths consistently improves performance on the Soundness metric, demonstrating *PRoSFI*'s robust and scalable behavior at test time. This trend holds across both the `ProverQA-Hard` and `ProverQA-Extra` subsets. In contrast, methods such as `Outcome-CoT` fail to benefit from additional sampling, as majority voting alone is insufficient to discern the soundness of the reasoning paths and cannot effectively filter out invalid reasoning.

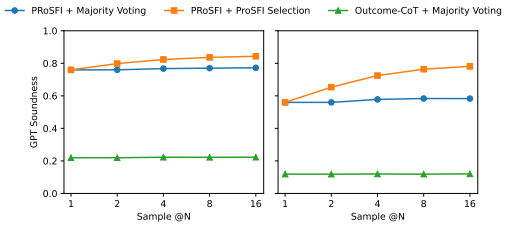


Figure 2: **Left:** GPT-judged Soundness on `ProverQA -hard` subset. **Right:** GPT-judged Soundness on `ProverQA-extra` subset. As the number of sampled paths increases, *PRoSFI* with *PRoSFI* selection consistently improves, while `Outcome-CoT` cannot benefit due to translation limits.

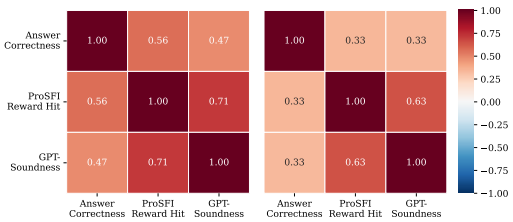


Figure 3: **Left:** Metric correlation heatmap on `ProverQA-Hard`. **Right:** Metric correlation heatmap on `ProverQA-Extra`. Compared to Answer Correctness, *PRoSFI* Reward Hit better reflects reasoning-path soundness.

Table 3: Results on Knights and Knaves dataset.

Method	Metric (%)	2ppl	3ppl	4ppl	5ppl	6ppl	7ppl	8ppl
Outcome-CoT	Answer Correct Rate	99.63	99.81	99.81	98.13	98.69	95.13	94.25
	GPT-judged Soundness	64.81	68.56	69.13	61.75	56.06	53.88	47.94
PRoSFI	Answer Correct Rate	99.75	99.31	99.75	99.63	99.25	98.56	97.38
	GPT-judged Soundness	86.56 ↑ 21.8	90.31 ↑ 21.8	90.50 ↑ 21.4	90.44 ↑ 28.7	91.00 ↑ 35.0	86.31 ↑ 32.4	80.94 ↑ 33.0

5.5 *PRoSFI* Reward Aligns Well with Soundness

To evaluate the effectiveness of the reward design used in *PRoSFI*, we conduct a correlation analysis among three metrics: Answer Correctness, *PRoSFI* Reward Hit, and GPT-Soundness of generated reasoning paths. This analysis is performed on the outputs of *PRoSFI*, and the results are in Figure 3.

The results reveal that the *PRoSFI* Reward Hit signal exhibits a high correlation with GPT-Soundness, much higher than the Answer Correctness. This suggests that our reward function is more aligned with step-by-step reasoning quality, thereby achieving our design goal. In contrast, answer correctness alone may fail to capture subtle flaws or inconsistencies in intermediate reasoning steps. These findings highlight that *PRoSFI*'s reward design provides meaningful supervision that better promotes logically coherent and sound outputs.

6 *PRoSFI* Generalizes to Knights and Knaves Data

To demonstrate the generalization ability of our proposed method, we extend experiments to the Knights and Knaves dataset Xie et al. (2025a), whose details can be found in Appendix B.3.

For the Outcome-CoT baseline, we adopt the same prompt template as Logic-RL, while the prompt for *PRoSFI* is adapted slightly to suit this task. Following the out-of-distribution (OOD) evaluation protocol established in Logic-RL, our model is trained exclusively on instances with difficulty levels from 3ppl to 7ppl. Evaluation is then conducted across the full test range, from 2ppl to 8ppl, with the 8ppl split serving as the primary benchmark for OOD performance.

The training configuration closely follows that of ProverQA, including a batch size of 64, 8 sampled responses per prompt, a maximum sequence length of 5120, and the same KL coefficient and clip-higher strategy from DAPO, except that we set the learning rate to 4×10^{-7}, consistent with Logic-RL, and extend the training schedule to 1440 steps to align with our experimental design.

From the results shown in Table 3, we can see that on the Knights and Knaves dataset, ProSFI consistently improves GPT soundness over Outcome-CoT across all difficulty levels, while maintaining comparable answer accuracy, demonstrating strong logical consistency and reliable generalization, particularly on the hardest 8ppl split. Therefore, our approach is generalizable to other logical reasoning tasks beyond ProverQA.

Limitations

PRoSFI is evaluated primarily on first-order logic tasks; broader applications may require richer datasets, stronger models, and specialized formal tools. Binary verification provides limited feedback on partial progress and individual steps' contributions, motivating finer-grained rewards and credit assignment. Moreover, verification establishes validity only within the formal representation and does not guarantee alignment with the natural-language reasoning. Our GPT-judged soundness metric is also approximate, and the 20-example human audit provides only a sanity check rather than large-scale validation.

AI USE STATEMENT

In this work, we used generative AI tools to help writing: grammar and spelling correction, rewording for clarity. We also used AI-based coding assistants for routine implementation support, such as data-processing and plotting scripts. We did not use generative AI tools to generate research ideas, to design the method, to derive or state any of the claims, to produce or alter any experimental result; the literature survey and all scientific content are our own.

We have reviewed all AI-generated content. Every AI-edited sentence was read and approved by at least one author, and AI-assisted code was inspected and tested by the authors before any reported number was produced from it. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or the collection of new data from people. All experiments are run on publicly released datasets (ProverQA and Knights and Knaves) and publicly released open-weight base models; we use them in accordance with their licenses and intended research use, and we introduce no new data collection of our own.

The aim of the method is to make model reasoning *more* checkable rather than merely more persuasive, which we regard as a net positive for safety: the formal intermediates produced by *PRoSFI* are machine-verifiable, so a reader is not asked to trust a fluent natural-language argument on its face. We note two limitations that bear on responsible use. First, verification is only as strong as the formalization: a reasoning chain whose formal intermediates pass the prover is sound *with respect to the formalized premises*, and a mis-formalized premise can still yield a verified-but-wrong conclusion. Verification should therefore not be read as a guarantee of factual correctness. Second, our evaluation is confined to first-order logic tasks, so claims about reliability should not be extrapolated to open-domain reasoning or to high-stakes deployment without further study. We discuss both points in the Limitations section.

We are not aware of conflicts of interest or sponsorship that would bias the reported results.

REPRODUCIBILITY STATEMENT

We have aimed to make every component of this work reproducible from the paper and its appendix. The datasets, base models, RL algorithms, and evaluation metrics are specified in Section 3, with the full construction and filtering of the ProverQA splits (including the synthesized *Extra Split*) in Section B and of the Knights and Knaves splits in Section B.3. The method itself, including the structure of the formal intermediates and how prover feedback is turned into a reward, is given in Section 4, and the training procedure is written out step by step in Section C. The exact reward function used for step-by-step verification, including how partial and failed proofs are scored, is stated in the appendix.

Because prompt wording materially affects RL training on these tasks, we reproduce the complete prompts verbatim rather than paraphrasing them: the baseline prompts in Section D, and the system and user prompts for *PRoSFI*, together with a representative post-training model response, in the following appendix section. Training hyperparameters — batch size, number of sampled responses per prompt, maximum sequence length, learning rate, KL coefficient, clipping strategy, and training schedule — are reported in the main text for both datasets. The appendix also documents the LLM-based soundness metric and our credibility analysis of it, so that the evaluation protocol can be reimplemented and audited independently.

Source code for training, verification, and evaluation will be released; an anonymized copy is provided as supplementary material with this submission.

REFERENCES

Chuxue Cao, Jinluan Yang, Haoran Li, Kunhao Pan, Zijian Zhao, Zhengyu Chen, Yuchen Tian, Lijun Wu, Conghui He, Yike Guo, and Sirui Han. Pushing the boundaries of natural reasoning: Inter-

leaved bonus from formal-logic verification in language models. *arXiv preprint arXiv:2601.22642*, 2026.

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, et al. Seed-Prover: Deep and broad reasoning for automated theorem proving. *arXiv preprint arXiv:2507.23726*, 2025a.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025b.

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

Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340. Springer, 2008.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 23–29 Jul 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, et al. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22017–22031, Miami, Florida, USA, 2024.

Lukas Helff, Quentin Delfosse, David Steinmann, Ruben Härle, Hikaru Shindo, Patrick Schramowski, Wolfgang Stammer, Kristian Kersting, and Felix Friedrich. LLMs gaming verifiers: RLVR can lead to reward hacking. *arXiv preprint arXiv:2604.15149*, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS 2021), Datasets and Benchmarks Track*, 2021.

Yichen Huang and Lin F Yang. Winning gold at IMO 2025 with a model-agnostic verification-and-refinement pipeline. *arXiv preprint arXiv:2507.15855*, 2025.

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

Jin Jiang, Yuchen Yan, Yang Liu, Jianing Wang, Shuai Peng, Xunliang Cai, Yixin Cao, Mengdi Zhang, and Liangcai Gao. LogicPro: Improving complex logical reasoning via program-guided learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26200–26218, Vienna, Austria, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *Proceedings of the International Conference on Learning Representations (ICLR 2024)*, 2024.

Hanmeng Liu, Zhiyang Teng, Chaoli Zhang, and Yue Zhang. Logic Agent: Enhancing validity with logic rule invocation. *arXiv preprint arXiv:2404.18130*, 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Second Conference on Language Modeling (COLM 2025)*, 2025.

William McCune. Release of Prover9. In *Mile high conference on quasigroups, loops and nonassociative systems, Denver, Colorado*, 2005.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. Learning deductive reasoning from synthetic corpus based on formal logic. In *International Conference on Machine Learning*, pp. 25254–25274. PMLR, 2023.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. Enhancing reasoning capabilities of LLMs via principled synthetic logic corpus. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, pp. 73572–73604, 2024.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, pp. 625–635. Springer, 2021.

Justin D. Norman, Michael U. Rivera, and D. Alex Hughes. Reliability without validity: A systematic, large-scale evaluation of LLM-as-a-Judge models across agreement, consistency, and bias. *arXiv preprint arXiv:2606.19544*, 2026.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176, Singapore, December 2023. Association for Computational Linguistics.

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

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. TinyZero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, Singapore, December 2023. Association for Computational Linguistics.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. LogicBench: Towards systematic evaluation of logical reasoning ability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13679–13707, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

Lawrence C Paulson. *Isabelle: A generic theorem prover*. Springer, 1994.

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

Massimiliano Pronesti, Anya Belz, and Yufang Hou. Beyond outcome verification: Verifiable process reward models for structured reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32187–32202, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1611.

Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. Large language models meet symbolic provers for logical reasoning evaluation. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, 2025.

Qwen Team. Qwen2.5: A party of foundation models, September 2024.

ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. In *Proceedings of the International Conference on Learning Representations (ICLR 2025)*, 2025.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in RLVR. *arXiv preprint arXiv:2506.10947*, 2025.

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

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, 2025.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*, 2020.

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

Chao Wang, Hexuan Deng, Min Zhang, Ruiyu Fang, Shuangyong Song, Shuo Nie, Xuebo Liu, Xuelong Li, and Yu Li. Stop rewarding hallucinated steps: Faithfulness-aware step-level reinforcement learning for small reasoning models. *arXiv preprint arXiv:2602.05897*, 2026.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, et al. Kimina-Prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025.

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

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 2742–2785, Mumbai, India, December 2025a.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-RL: Unleashing LLM reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025b.

Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, et al. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte-Carlo tree search. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, 2025.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13326–13365, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

Jundong Xu, Hao Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Yang Wang, Mong-Li Lee, and Wynne Hsu. LogicReward: Incentivizing LLM reasoning via step-wise logical supervision. In *The Fourteenth International Conference on Learning Representations (ICLR 2026)*, 2026.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J. Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Datasets and Benchmarks Track*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. ReClor: A reading comprehension dataset requiring logical reasoning. *arXiv preprint arXiv:2002.04326*, 2020.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. SimpleRL-Zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Second Conference on Language Modeling (COLM 2025)*, 2025.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: A cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations (ICLR 2022)*, 2022.

Jin Peng Zhou, Charles E Staats, Wenda Li, Christian Szegedy, Kilian Q Weinberger, and Yuhuai Wu. Don't Trust: Verify – grounding LLM quantitative reasoning with autoformalization. In *The Twelfth International Conference on Learning Representations*, 2024.

Yichi Zhou, Jianqiu Zhao, Yongxin Zhang, Bohan Wang, Siran Wang, Luoxin Chen, Jiahui Wang, Haowei Chen, Allan Jie, Xinbo Zhang, et al. Solving formal math problems by decomposition and iterative reflection. *arXiv preprint arXiv:2507.15225*, 2025.

Appendices

A USE OF LARGE LANGUAGE MODELS (LLMS)

While preparing this manuscript, we made use of Large Language Models (LLMs) as writing aids, specifically for tasks such as grammar correction, wording refinement, and rephrasing to enhance clarity and readability. In line with ethical guidelines, we, the authors, confirm that we take full responsibility for the content of this work. Any text produced with the assistance of LLMs has been carefully reviewed, revised, and approved by us. All scientific claims, findings, and conclusions are entirely our own. We are responsible for any possible mistakes, inaccuracies, or ethical issues that may arise in this submission.

B DETAILS OF DATASETS

B.1 PROVERQA DATASET

The ProverQA dataset contains 1,500 problems evenly split into easy, medium, and hard subsets, each defined by controlled complexity. For each difficulty level, we first filtered out the instances labeled as "Uncertain" and reserved 64 instances examples for testing. To assess generalization to even harder problems, we further synthesized 128 additional extra hard examples using the same framework, referred to as the *Extra Split*.

Every instance of ProverQA comprises:

- A background story created for each problem to introduce linguistic context and variation,
- A logic skeleton of FOL premises (facts + rules with 7 operators: $\wedge, \vee, \neg, \rightarrow, \oplus, \forall, \exists$) and distractors (non-essential facts or semantically related but irrelevant rules),
- A goal to classify as *True, False, or Uncertain*,
- Intermediate reasoning chains provided in both symbolic form and natural language.

This design meets key criteria: scalability, natural & diverse language, explicit symbolic representations, and faithful reasoning chains. At the same time, it is challenging enough, for example, GPT4o achieves accuracy 46.2 with standard prompts, and 50 with CoT prompts for hard samples (Qi et al., 2025). Hence it is good to work with ProverQA for logical reasoning studies.

An Example from ProverQA


```
{
  "id": 0,
  "options": [
    "A) True",
    "B) False",
    "C) Uncertain"
  ],
  "answer": "B",
  "question": "Based on the above information, is the following statement true, false, or
    ↪ uncertain? Brecken has never experienced heartbreak.",
  "reasoning": "fact1: Brecken has experienced heartbreak.\nrule: Either Brecken has
    ↪ experienced heartbreak or he has never experienced heartbreak, but not both.\
    ↪ nconclusion: Brecken has experienced heartbreak.\n\nTherefore, it is false that
    ↪ Brecken has never experienced heartbreak. The correct option is: B.",
  "context": "Brecken has experienced heartbreak. Either Brecken has experienced
    ↪ heartbreak or he has never experienced heartbreak, but not both.",
  "nl2fol": {
    "Brecken has experienced heartbreak.": "has_experienced_heartbreak(Brecken)",
    "Either Brecken has experienced heartbreak or he has never experienced heartbreak, but
    ↪  not both.": "has_experienced_heartbreak(Brecken) ⊕
    ↪ has_never_experienced_heartbreak(Brecken)"
  },
  "conclusion_fol": "has_never_experienced_heartbreak(Brecken)"
}
```

B.2 An Example from ProverQA

To illustrate the structure of the ProverQA dataset, we present a representative example as follows. Each entry is organized as a JSON object containing the question, multiple-choice options, the correct answer, and the reasoning chain used to derive the conclusion. In addition, the dataset provides a context field capturing the relevant facts and rules, as well as a mapping from natural language (NL) to first-order logic (FOL) formulas (nl2fol). Finally, the conclusion_fol field specifies the logical statement under evaluation.

This example highlights how ProverQA combines natural language reasoning with its corresponding logical formalization, enabling the evaluation of models not only on final predictions but also on their formal reasoning steps.

B.3 Knight and Knaves Dataset

The Knights and Knaves (K&K) puzzles(Xie et al., 2025b) are an algorithmically generated reasoning dataset where characters are either knights, who always tell the truth, or knaves, who always lie. The goal is to determine each character's role from their statements. The dataset is procedurally generated, providing unlimited variation and unseen test cases for evaluating generalization. Difficulty is precisely controlled by varying the number of characters (2–8) and the complexity of logical operations, supporting curriculum learning and out-of-distribution testing. Each puzzle has a single verifiable solution, ensuring evaluations rely on strict deductive reasoning.

K&K dataset is particularly useful due to its formal, synthetic design. Every puzzle is governed by explicit rules and admits a unique, deterministic solution, eliminating ambiguity and allowing researchers to clearly separate true reasoning ability from memorization.

Here is an example adapted from the Xie et al. (2025b).

An Example from K&K Puzzle Problem

```
Problem: A very special island is inhabited only by knights and knaves. Knights always
    ↪ tell the truth, and knaves always lie. You meet 2 inhabitants: Zoey, and Oliver.
    ↪ Zoey remarked, "Oliver is not a knight". Oliver stated, "Oliver is a knight if and
    ↪  only if Zoey is a knave". So who is a knight and who is a knave?

Solution: (1) Zoey is a knave (2) Oliver is a knight
```

C Algorithm Details

For completeness, we revisit the GRPO algorithm in detail. The procedure consists of the following steps:

1. Samples a group of output sequences $\{o_1, \ldots, o_G\}$ for each prompt q from the policy $\pi_{\theta_{\text{old}}}$.
2. Scores each sequence o_i with a reward model $r_\phi(q, o_i)$, yielding rewards $\{r_i\}_{i=1}^G$.
3. Computes relative advantages by normalizing these rewards within the group:

$$\hat{A}_i = \frac{r_i - \frac{1}{G}\sum_{j=1}^G r_j}{\sqrt{\frac{1}{G}\sum_{j=1}^G \left(r_j - \frac{1}{G}\sum_{k=1}^G r_k\right)^2}}.$$

4. Optimizes the policy by maximizing a PPO-style clipped surrogate objective, with a KL penalty against a frozen reference model π_{ref}.

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^G}\left[\frac{1}{G}\sum_{i=1}^G\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta\mathbb{D}_{KL}(\pi_\theta\|\pi_{\text{ref}})\right)\right]$$

$$\mathbb{D}_{KL}(\pi_\theta \| \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - \log \frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - 1,$$

Moreover in the experiments, we implement the clip-higher strategy from the Decoupled Advantage Policy Optimization (DAPO) algorithm (Yu et al., 2025) to stablize the training of GRPO. This strategy increases the upper clip range of the importance sampling ratio in the policy gradient loss to encourage exploration. Consistent with the DAPO methodology, we set the asymmetric clipping range with $\varepsilon_{\text{low}} = 0.2$ and $\varepsilon_{\text{high}} = 0.28$.

D Prompts for The Outcome-CoT Approach

System Prompt for Outcome-CoT

```
System:

Given a problem statement as contexts, the task is to answer a logical reasoning question.

The assistant first thinks about the reasoning process in the mind and then provides the
    ↪ user with the answer.
The reasoning process is enclosed within <think> </think> tags, followed by the answer, i.
    ↪ e., <think> reasoning process here </think> answer here.

The last A), B), or C) in the answer section (i.e., the content outside <think> </think>)
will be considered your answer. If the format is incorrect, you may not receive a score.
```

The User Prompt for the Outcome-CoT approach used in Section 5.1 is as follows.

The prompt begins with a context section containing background knowledge expressed in natural language, including facts (e.g., Paola appreciates beauty) and rules (e.g., Anyone who can solve problems and communicate effectively has practical skills). These statements establish the logical environment within which reasoning must take place.

Following the context, the prompt specifies a question that requires truth evaluation of a target statement—in this case, whether Paola is artistically inclined. The prompt also provides a set of options (True, False, or Uncertain), mirroring the multiple-choice format used throughout the dataset. Finally, the template ends with the phrase "The correct option is:", leaving space for the model to generate its answer.

User Prompt for Outcome-CoT

```
User:

Context:
Anyone who can solve problems and communicate effectively has practical skills.
If Paola appreciates beauty, then she values precision and enjoys manual work.
Paola appreciates beauty.
Paola is either skilled in mathematics or has practical skills, but not both.
If someone enjoys manual work and values attention to detail, then they can craft with
    ↪ their hands.
Paola is either skilled in mathematics or artistically inclined, but surprisingly, she is
    ↪ actually both.
Paola values attention to detail.
Anyone who crafts with their hands has practical skills.

Question:
Based on the above information, is the following statement true, false, or uncertain?
Paola is artistically inclined.

Options:
A) True
B) False
C) Uncertain

The correct option is:
```

E PROMPTS AND EXAMPLE RESPONSES FOR *PRoSFI*

E.1 SYSTEM PROMPT FOR *PRoSFI*

To demonstrate how *PRoSFI* is guided in the reasoning process, we provide the system prompt used in Section 4 (see Listing X). The prompt instructs the model to behave as a mathematical reasoner that operates on premises expressed in Lean4 syntax (e.g., $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\oplus$, $\forall$, $\exists$).

The structure of the prompt has several key elements:

- **Formatted reasoning**: All intermediate thinking is enclosed within <think>...</think>.
- **Formatted summarized output**: Inside <summary>...</summary>, the model must return a JSON array of atomic inference steps.

Each **JSON object** specifies:

- id: a unique identifier for the step,
- dependencies: the premises or earlier steps it relies on,
- conclusion: a single new FOL formula expressed in Lean4,
- rule: the inference rule applied.

Constraints – Each inference step must be as fine-grained as possible (fewer than 5 dependencies), and the JSON must be valid and parsable. The final object corresponds to the model's chosen answer and must align with one of the provided options.

This design enforces faithful, step-by-step reasoning, making the model's decision process transparent and verifiable.

System Prompt for PRoSFI

````
System:

You are a math reasoner. For each problem, you will receive a set of premises annotated in
    ↪ Lean4 (use →, ⊕, ∧, ∨, ¬, ∀, ∃). Answer the multiple-choice reasoning question
    ↪ as follows:

1. Internal reasoning goes inside <think>···</think>.
2. Then, inside <summary>···</summary>, emit a JSON array of atomic inferences, one per
    ↪ object:

  ```json
  {
    "id": "s",
    "dependencies": ["h1","s"], // original premises or earlier steps
    "conclusion": "...",           // exactly one new FOL formula (Lean4 syntax)
    "rule": "..."                  // the single inference rule applied
  }
  ```

  - The JSON array must be valid and parsable without errors.
  - Each object must represent exactly one atomic inference step.
  - Each step should be as atomic as possible, using less than 5 dependencies.

The last object in the array corresponds to your answer. Its id and conclusion must be
    ↪ identical to one of the options.
````

E.2 USER PROMPT FOR PROSFI

To make the interaction format concrete, we show the **user prompt** used for *PRoSFI* in Section 4. The prompt is divided into several components:

- **Context**: A sequence of natural language premises paired with their corresponding formal statements in Lean4 syntax. For example, the first two premises assert conditional rules about Hattie and Candy, while the 3rd and 4th connect moral courage to principled action.

- **Question**: A query formulated in natural language, asking whether a target statement (here, "Hattie does not gain community respect") is true or false given the context.
- **Options**: The possible answers, each expressed both in natural language and as a formal Lean4 statement.
- **Answer slot**: The prompt ends with "The correct option is:", leaving space for the model to provide its choice.

This structure illustrates how *PRoSFI* is guided to align natural language reasoning with formal logical inference, ensuring that both the problem and the candidate answers are clearly specified in parallel in both natural language and formal statement.

User Prompt for PRoSFI

```
User:

Context:
1. If Hattie stands up for her principles, then she gains community respect. Formal
    ↪ statement: `h₁ : stands_up_for_principles Hattie → gains_community_respect Hattie
    ↪ `.
2. If Candy stands up for her principles, then she gains community respect. Formal
    ↪ statement: `h₂ : stands_up_for_principles Candy → gains_community_respect Candy`.
3. Hattie has moral courage. Formal statement: `h₃ : has_moral_courage Hattie`.
4. Everyone who has moral courage will stand up for their principles. Formal statement: `
    ↪ h₄ : ∀x, (has_moral_courage x → stands_up_for_principles x)`.

Question: Based on the above information, is the following statement true or false? Hattie
    ↪  does not gain community respect.

Options:
A) True. Formal statement: `h_goal_true: ¬gains_community_respect Hattie`.
B) False. Formal statement: `h_goal_false: ¬(¬gains_community_respect Hattie)`.

The correct option is:
```

E.3 Model Response after RL with PRoSFI

We next illustrate how the model responds after RL with PRoSFI. The response is divided into two main parts:

- **Reasoning path**: Enclosed within <think> … </think>, the model provides a step-by-step explanation in natural language. In this case, it applies Universal Instantiation and Modus Ponens to derive that Hattie stands up for her principles, and subsequently that she gains community respect. This reasoning shows why the negated claim (Hattie does not gain community respect) is false.
- **Structured summary**: Enclosed within <summary> … </summary>, the model outputs a JSON array of atomic inference steps. Each object specifies the inference id, the set of dependencies, the derived conclusion (in Lean4 syntax), and the rule applied. The last object corresponds to the final answer and matches one of the candidate options.

We can see the model's response is both interpretable in natural language and machine-verifiable through its structured reasoning steps.

```
Assistant:

<think>
1. From h₄ (∀x, has_moral_courage x → stands_up_for_principles x) and h₃ (has_moral_courage
    ↪ Hattie), by Universal Instantiation on Hattie and Modus Ponens, derive
    ↪ stands_up_for_principles Hattie.
2. From h₁ (stands_up_for_principles Hattie → gains_community_respect Hattie) and the above,
    ↪ by Modus Ponens, derive gains_community_respect Hattie.
3. Therefore ¬ gains_community_respect Hattie is false.
</think>
<summary>[
  {
    "id": "s1",
    "dependencies": ["h₃", "h₄"],
```

```
    "conclusion": "stands_up_for_principles Hattie",
    "rule": "Universal Instantiation + Modus Ponens"
  },
  {
    "id": "h_goal_false",
    "dependencies": ["h1", "s1"],
    "conclusion": "¬(¬gains_community_respect Hattie)",
    "rule": "Modus Ponens"
  }
]
</summary>
```

F TYPICAL ERRORS FOR OUTCOME-COT

In this section, we select two representative cases to analyze two typical types of errors that occur during the reasoning process of the `Outcome-CoT` model, namely semantic confusion and commonsense shortcuts.

F.1 SEMANTIC CONFUSION

```
User:

Context:
...
3. Either Miracle is community-oriented or self-centered, but not both.
...
6. If Miracle is a leader, then she is either selfless or thinks of herself, but not both.
...
Assistant:
...
Since she is community-oriented, she cannot be self-centered (statement 3). Therefore, she
    ↪ must be selfless. Thus, the statement 'Miracle is not self-centered' is true.
...
```

In this example, the model infers that "`Miracle is not self-centered`" from the premise that "`Miracle is community-oriented`", and then equates "`not self-centered`" with "`selfless`". In doing so, it bypasses a rigorous logical analysis of the premises and directly concludes that "`Miracle is selfless`". However, determining whether Miracle is truly `selfless` requires a more comprehensive evaluation based on Proposition 6, whether Miracle is a leader, as well as the mutually exclusive conditions specified by the two choices. Thus, the model's conclusion reflects a logical error arising from semantic confusion.

F.2 COMMONSENSE SHORTCUT

```
User:

Context:
...
6. If Alvaro faces adversity, then he either stays motivated or pursues his goals, but not
    ↪ both.
...
Assistant:
...
Since Alvaro is facing adversity and is either staying motivated or pursuing his goals, it is
    ↪ clear that Alvaro is overcoming challenges in some form. Therefore, the statement "
    ↪ Alvaro does not overcome challenges" is false.
...
```

In this example, after confirming that Alvaro is facing adversity and satisfies either "`staying motivated`" or "`pursuing goals`", the model proceeds to conclude that "`Alvaro overcame the challenge`". Although this inference may seem reasonable from a commonsense perspective, it lacks a rigorous logical basis. The context of the question does not provide information on whether Alvaro actually overcame the challenge. Therefore, the model's conclusion is a flawed logical generalization based on a commonsense shortcut, which fails to adhere to the principles of deductive reasoning.

G Details and Credibility Analysis of the GPT Soundness Metric

System prompt

```
System:

The following is a problem statement and a reasoning chain to solve it.

Problem:
...

Reasoning:
...

Task:
Please evaluate whether the reasoning process contains any irreparable logical errors.
- If so, return a JSON where "is_error" is True, "error_instance" contains the original
    ↪ text of the first instance of the error, "error_type" specifies the type of the
    ↪ error, and "error_reason" explains the reason for the error.
- Otherwise, return a JSON where "is_error" is False.
Wrap your JSON within ```json ```.
```

We use a simple prompt to ask the GPT-OSS-120B model to evaluate the correctness of the reasoning path and ask it to return a JSON; when the reasoning is correct, it only needs to return a bool value, but when it is incorrect, it needs to provide the location and reason for the error.

We manually checked 10 outputs each from `Outcome-CoT` and *PRoSFI* to evaluate the correctness of GPT's assessment of the reasoning process.

For the outputs of `Outcome-CoT`, GPT's evaluations were all correct. Among them, 8 outputs had errors of invalid deduction, one output was completely correct, and one output had an error of misinterpretation of premise.

For the outputs of *PRoSFI*, GPT's evaluations were almost entirely correct but slightly too strict. Among them, 5 outputs had errors of invalid deduction, while 3 outputs were completely correct. The remaining two outputs had reasoning that was generally correct but contained some minor errors: the first output used unnecessary premises in one step, and the second output wrote the wrong name for the reasoning rule used.

H Reward Function for *PRoSFI* Step-by-Step Verification

We first extract a structured JSON representation from the natural language reasoning paths generated by the model. We then determine whether the format is correct based on its validity and the structural specifications defined in the main text. By analyzing the `id` and `conclusion` fields of the last object in the JSON, we can determine whether the final answer is correct.

To verify the logical correctness of the entire reasoning process, we use a top-down syntax parser to parse the first-order logic (FOL) expressions generated by the model. These are then converted into formats supported by target formal tools such as Lean4, Prover9, or Z3. For each reasoning conclusion, we express it as a proposition in Lean4 and verify it using a sequence of tactics or a large language model capable of formal theorem proving. Alternatively, the reasoning steps can be converted into input formats accepted by Prover9 or Z3, and the validity of each step can be verified using an automated solver.

The algorithm is presented in Algorithm 1, which follows this routine:

1. Extract a summary and FOL statements from the model's response.
2. If no valid statements are extracted, return a score of 0.
3. If the ID of the final reasoning step doesn't match the ground truth, return a score reflecting formatting issues.
4. If all FOL problems are solved correctly, return the full score.
5. Otherwise, return a partial (optional) score.

Algorithm 1 Reward Function for *PRoSFI* Step-by-Step Verification

```
Input: response, ground_truth
Output: score
summary ← extract_summary(response)
statements ← extract_fol_problems(summary, ground_truth)
if statements is None then
    return 0.0
end if
if summary.last_id ≠ ground_truth then
    return format_score
end if
if fol_all_solved(statements) then
    return full_score
else
    return option_score
end if
```

I Correlation Analysis of the Results on Outcome-CoT

For the output of the `Outcome-CoT` method, we analyze the relationship between two available metrics: the correctness of the final answer and the reasoning soundness automatically evaluated by GPT (GPT Soundness). The confusion matrix is shown in Figure 4.

It can be observed that a large number of reasoning paths, although yielding the correct answer, are evaluated as unsound in their reasoning process. This result indicates that a correct answer alone is not sufficient to reflect the logical validity of the reasoning process. It further supports the qualitative observation in Section 5.1 — namely, that traditional CoT methods struggle to ensure the validity of reasoning paths.

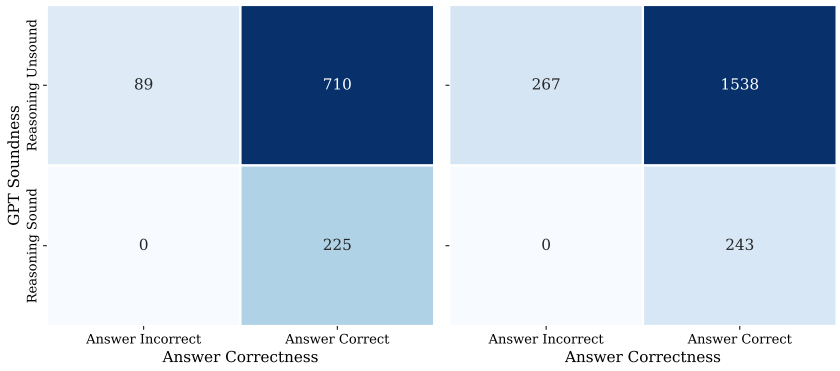


Figure 4: **Left:** Confusion matrix of answer correctness vs. reasoning soundness for `Outcome-CoT` on the `ProverQA-hard` subset. **Right:** Corresponding results on the `ProverQA-extra` subset. The lack of alignment between correct answers and sound reasoning indicates that answer accuracy alone is not a sufficient criterion for judging the quality of reasoning.

In comparison, Figure 3 presents the heatmaps of variable correlations on the two subsets. The results show that the designed reward signal has a stronger correlation with GPT Soundness than with the final answer correctness. This suggests that our formal verification mechanism effectively identifies and enhances logical rigor, making it a more reliable signal for training and selection.