VERILOGIC: REINFORCEMENT LEARNING OVER VERIFIABLE LOGICAL REASONING TRACES

Anonymous authors
Paper under double-blind review

ABSTRACT

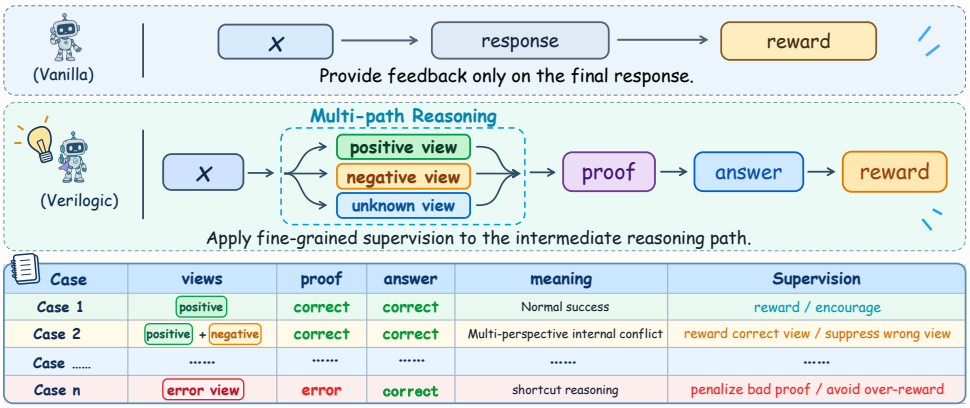
Logical reasoning derives conclusions from premises through explicit intermediate inference steps, providing a natural basis for process-level verification and supervision. However, existing learning methods often rely on response-level outcome signals, which overlook heterogeneous correctness within the reasoning trace: a correct answer may contain invalid reasoning, while valid intermediate steps may also appear in an incorrect response. To address this issue, we introduce Reinforcement Learning on **Veri**fiable **Logic**al Reasoning (**VeriLogic**), which turns internal reasoning traces into explicit and verifiable learning objects. Rather than treating a response as an indivisible learning unit, VeriLogic decomposes logical reasoning into explicit intermediate judgments, proofs, and final decisions, and organizes them into a unified "**Grounding - Optimization - Refinement**" pipeline: (1) ***Verification-Grounded Reasoning Environment Construction*** grounds logical reasoning into explicit intermediate states and formally checkable proof obligations, making the internal reasoning process accessible to verification and learning; (2) ***Logical Structured Policy Optimization*** learns from verifier feedback through complementary global and local rewards, jointly optimizing the overall reasoning outcome while providing targeted optimization signals to different intermediate reasoning paths; and (3) ***Verifier-Guided Iterative Refinement*** acts on verifier feedback by leveraging proof states and error messages to iteratively diagnose and repair failed reasoning. Experiments on six logical reasoning benchmarks show consistent gains over strong baselines, with average improvements of 6.39% and 5.00% on Qwen3 and Llama-3.1. These results highlight the value of making logical reasoning traces verifiable, learnable, and refinable beyond final-answer supervision.

1 INTRODUCTION

Logical reasoning (Smith, 2003; Enderton, 2001) is a fundamental capability of LLMs (Huang & Chang, 2023; Liu et al., 2025a), where a model derives conclusions from given premises through valid inference. Unlike many open-ended reasoning tasks (Cobbe et al., 2021; Hendrycks et al., 2021), logical reasoning is ***naturally structured***: intermediate judgments can be decomposed, inference paths can be traced, and their validity can often be explicitly checked. Yet this structure ***remains largely underused***. Existing methods mainly improve logical reasoning through prompting (Wei et al., 2022; Zhou et al., 2022) or inference-time augmentation (Wang et al., 2022; Yao et al., 2023), rather than internalizing it as a trainable model capability. Reinforcement Learning on Logic Reasoning offers a promising alternative, but a few current attempts (Xie et al., 2025; Liu et al., 2026) predominantly optimize final outputs with response-level rewards, overlooking the heterogeneous correctness that may arise across intermediate reasoning paths.

We exploit the potential capability of logical reasoning is to make its internal process both **traceable** and **verifiable**. Instead of treating reasoning as a single response, we break it into multiple explicit reasoning traces. Each trace follows a distinct reasoning path with intermediate judgments that can be examined step by step. More importantly, these traces can be optimized with verifier feedback according to their own correctness, rather than sharing a single reward from the final answer. This lets the model distinguish different reasoning outcomes—for example, correct reasoning with a wrong answer, wrong reasoning with a correct answer, or cases where both succeed or fail. As shown in Figure 1. Motivated by this view, we introduce Reinforcement Learning on **Veri**fiable

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.



Case	views	proof	answer	meaning	Supervision
Case 1	positive	correct	correct	Normal success	reward / encourage
Case 2	positive + negative	correct	correct	Multi-perspective internal conflict	reward correct view / suppress wrong view
Case	.....	.....	.....	.....	.....
Case n	error view	error	correct	shortcut reasoning	penalize bad proof / avoid over-reward

Figure 1: **Heterogeneous correctness in logical reasoning.** A final answer may conceal errors in intermediate reasoning views or proofs, while fine-grained supervision distinguishes and corrects these cases.

Logical Reasoning (**VeriLogic**), which turns internal reasoning traces into explicit and verifiable learning objects for fine-grained policy optimization.

Turning traceable and verifiable reasoning into an effective RL framework requires addressing three challenges: (**C1**) ***How to construct a verifiable reasoning space?*** Different reasoning paths must be explicitly represented, semantically aligned with the original problem, and formally checkable; (**C2**) ***How to optimize reasoning at the right granularity?*** Different reasoning paths within the same response may have different correctness states, so a single response-level reward cannot provide sufficiently precise optimization signals; (**C3**) ***How to make verification actionable after failure?*** Even after training, a model may still produce invalid or incomplete reasoning, requiring verifier feedback to guide iterative revision rather than serving only as a final correctness check.

To address these challenges, VeriLogic organizes logical reasoning into a unified "**Grounding - Optimization - Refinement**" pipeline. (1) ***Verifiable Reasoning Environment Construction*** exposes the internal structure of logical reasoning and makes each reasoning path formally checkable. It decomposes a problem into complementary reasoning views with explicit intermediate judgments and proof obligations, and grounds them into Lean4 for verification. (2) ***Logical Structured Reasoning Optimization*** optimizes different reasoning states at the granularity they require. It combines global supervision over the complete response with path-specific local feedback. The global reward evaluates answer correctness and output structure, while local rewards capture path correctness, proof validity, and consistency, dynamically providing finer-grained optimization signals to the corresponding reasoning paths. (3) ***Verifier-Guided Iterative Refinement*** turns verification failures into actionable signals for further reasoning. It feeds proof states, remaining goals, and error messages back to the model, enabling failed reasoning to be diagnosed, revised, and re-verified through an iterative loop. We evaluate VeriLogic on six logical reasoning benchmarks across three backbone models. VeriLogic consistently improves over the strongest corresponding baselines, with average gains of 1.83%, 6.39%, and 5.00% on Qwen3-8B, Qwen3-4B, and Llama-3.1-8B, respectively. Extensive ablations further confirm the the effectiveness of path-specific advantage routing and verifier-guided refinement. Overall, these results suggest that the key to improving logical reasoning is not only to reward correct answers, but to make the reasoning process itself explicit, verifiable, and optimizable.

The contributions of this work are summarized as follows:

- **A new RL paradigm for logical reasoning**. We deeply integrate reinforcement learning with logical reasoning by making the reasoning process both **traceable** and **verifiable**. Instead of optimizing only the final answer, VeriLogic exposes multi-path reasoning traces and directly optimizes their intermediate states with verifier-grounded feedback, forming a new RL framework for process-aware logical reasoning.
- **A unified "Grounding - Optimization - Refinement" RL framework**. To address the challenges of exposing reasoning structure, optimizing heterogeneous reasoning states, and recovering from reasoning failures, VeriLogic introduces three coordinated modules: ***Verifiable Reasoning Envi-***

ronment Construction to ground reasoning into explicit and checkable traces, ***Logical Structured Reasoning Optimization*** to combine global and local rewards for state-dependent optimization, and ***Verifier-Guided Iterative Refinement*** to diagnose and repair failed reasoning with verifier feedback.

2 Background and Related Work

LLM logical reasoning. Existing work improves LLM reasoning mainly through structured prompting, test-time scaling, and symbolic augmentation. Prompting methods such as Chain-of-Thought, Self-Consistency, Tree-of-Thoughts, Least-to-Most, Plan-and-Solve, and Graph-of-Thoughts encourage models to generate intermediate reasoning steps, aggregate multiple paths, plan ahead, or decompose complex problems before answering (Wei et al., 2022; Wang et al., 2022; Yao et al., 2023; Zhou et al., 2022; Wang et al., 2023; Besta et al., 2024). Recent reasoning models further scale this idea with longer reasoning chains, test-time compute, and reinforcement learning from outcome correctness (Jaech et al., 2024; Guo et al., 2025; Team et al., 2025; He et al., 2025; Guan et al., 2025). For logical reasoning, prior work introduces more explicit structures, including cumulative intermediate conclusions, determinacy-oriented reasoning, logic-augmented contexts, and semiotic-square-guided multi-perspective analysis (Zhang et al., 2023; Sun et al., 2024; Liu et al., 2025b; Zhang et al., 2026b). Another line translates natural-language premises into formal or symbolic representations and invokes external solvers, programs, or provers, such as Logic-LM, LINC, SymbCoT, PAL, CLOVER, and Aristotle (Pan et al., 2023; Olausson et al., 2023; Xu et al., 2024; Gao et al., 2023; Ryu et al., 2025; Xu et al., 2025). While these methods improve logical reasoning through prompting or inference-time tools, they usually do not train the model policy with formally verified feedback from its own generated reasoning.

Verifier-guided reasoning. Process-supervision methods evaluate intermediate reasoning steps rather than only final answers, using learned critics, human step labels, or automatically constructed process supervision (Uesato et al., 2022; Lightman et al., 2024; Wang et al., 2024; Luo et al., 2024; Zheng et al., 2025). However, such signals remain probabilistic and may overlook subtle logical errors. Symbolic verifiers provide stricter feedback by checking reasoning against formal semantics. LogicReward uses theorem-prover-based signals for step-wise logical supervision in natural-language reasoning (Xu et al., 2026), while recent Lean-based systems exploit proof-assistant feedback for formal theorem proving through informal-to-formal guidance, expert iteration, synthetic proof data, compiler feedback, subgoal decomposition, self-correction, copilot-style proof assistance, and reinforcement learning (Jiang et al., 2022; Xin et al., 2024; Lin et al., 2025a; Ying et al., 2024; Song et al., 2024; Ren et al., 2025; Wang et al., 2025; Lin et al., 2025b; 2026; Ji et al., 2025; Chen et al., 2025). Different from these works, our setting focuses on truth-value prediction over natural-language premise sets, where the model must distinguish `True`, `False`, and `Unknown` rather than merely produce a complete formal proof. VeriLogic therefore uses Lean4 feedback to verify the Positive and Negative Paths, while the Unknown Path receives no proof-verification reward. The resulting local credits are restricted to their aligned structured reasoning regions rather than broadcast uniformly across a response.

3 Methodology

VeriLogic turns internal logical reasoning into explicit and verifiable learning objects and organizes the overall process into a unified "**Grounding–Optimization–Refinement**" framework. ***Verifiable Reasoning Environment Construction*** exposes distinct reasoning paths and their proof targets, establishing a formally checkable reasoning space. ***Logical Structured Reasoning Optimization*** employs **Path-Specific Advantage Routing GRPO (PSAR-GRPO)** to combine the global outcome advantage with local advantages associated with different reasoning traces, allowing distinct correctness states within the reasoning process to receive corresponding fine-grained optimization signals. ***Verifier-Guided Iterative Refinement*** leverages proof states and error messages to diagnose and repair failed reasoning. Figure 2 illustrates the overall VeriLogic pipeline, while Algorithm 1 summarizes the complete procedure.

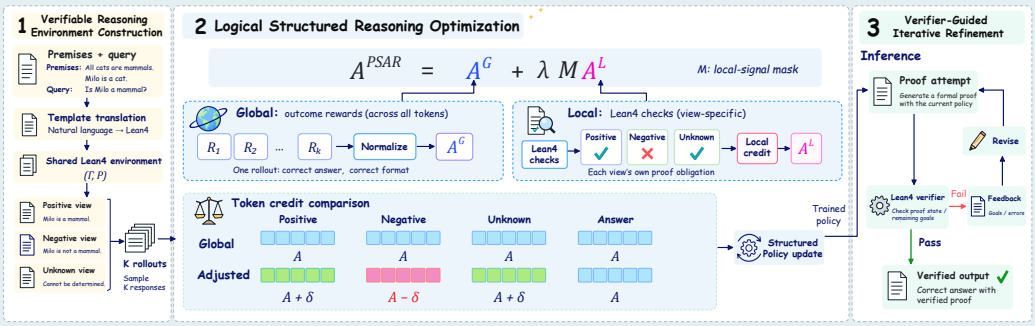


Figure 2: **Overview of VeriLogic.** VeriLogic follows a unified Grounding–Optimization–Refinement pipeline: it constructs explicit and verifiable reasoning paths, combines global outcome supervision with path-specific local advantages through PSAR-GRPO, and iteratively repairs unsuccessful proofs using Lean4 feedback at inference time.

3.1 Verifiable Reasoning Environment Construction

Explicit reasoning paths. We first formulate logical reasoning as a three-way task with explicit reasoning paths. Each input example is represented as a triple (C, q, y), where C is a natural-language context consisting of facts and rules, q is a queried proposition, and $y \in$ `True`, `False`, `Unknown` denotes the target label. Rather than directly predicting only the final label, we explicitly construct three reasoning paths corresponding to the three possible logical states—Positive, Negative, and Unknown—allowing the model to reason over each possibility separately.

Formal grounding and proof targets. To make the provable reasoning paths formally verifiable, we define a mapping τ that translates the natural-language problem into a Lean4 premise set and a target proposition:

$$\tau(C, q) = (\Gamma, P). \tag{1}$$

Here, Γ denotes the Lean4 premise set constructed from the context C, and P is the formal proposition semantically aligned with the query q. Based on this grounding, the Positive and Negative Paths are associated with the proof targets P and $\neg P$, respectively, and generate formal proofs aligned with their corresponding reasoning directions. The Unknown Path reasons about why the available information is insufficient to support either provable conclusion and therefore does not require a formal proof target.

This construction decomposes an otherwise implicit logical judgment into multiple reasoning paths with explicit semantics and proof targets, allowing different paths to be verified independently and providing the basis for subsequent fine-grained policy optimization over different reasoning states. The complete definition is provided in Appendix A.1.

3.2 Logical Structured Reasoning Optimization

Structured on-policy verification. Given the explicit and verifiable reasoning paths constructed above, we further use verifier feedback to optimize different reasoning states through structured policy learning. Specifically, we train the policy with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and integrate Lean4 verification directly into the on-policy rollout loop. For each problem, the current policy samples a group of responses, each containing multiple reasoning paths and their corresponding proofs, with different reasoning paths represented as separately addressable semantic regions. Lean4 checks the generated proofs, allowing reasoning paths and proofs with different correctness states within the same response to be evaluated separately, and the resulting verifier feedback is then used to update the current policy. The complete target mapping and output constraints are provided in Appendix A.

Global outcome advantage. To provide task-level supervision over the complete reasoning outcome, we first construct a global reward at the response level. For each input, the model explicitly

generates three reasoning paths: Positive, Negative, and Unknown. The Positive Path consists of reasoning that supports proposition P together with its corresponding proof, while the Negative Path consists of reasoning that supports $\neg P$ and its corresponding proof. The Unknown Path reasons about why the available information is insufficient to support either provable conclusion and does not contain a formal proof. After completing the three paths, the model synthesizes their reasoning outcomes and autonomously produces the final three-way answer.

The global reward considers final-answer correctness and structured-output validity. For a prompt u with rollout group $\mathcal{G}_u$, we further normalize the response-level rewards within the group to obtain the global advantage:

$$R_i^{\mathrm{G}} = \lambda_a R_{\mathrm{ans}}(o_i, y) + \lambda_f R_{\mathrm{fmt}}(o_i), \qquad A_i^{\mathrm{G}} = \frac{R_i^{\mathrm{G}} - \mu_{u,G}}{\sigma_{u,G} + \epsilon}. \tag{2}$$

Here, R_{ans} measures whether the final three-way answer matches the target label, while R_{fmt} measures whether the complete response satisfies the predefined structured-output constraints. $\mu_{u,G}$ and $\sigma_{u,G}$ denote the mean and standard deviation of the global rewards within the rollout group, respectively. The resulting global advantage is broadcast to all valid tokens in the response, providing a unified task-level optimization signal for the complete reasoning process. Since this shared signal captures only the overall outcome and cannot distinguish correctness differences among the three reasoning paths, we further introduce path-level local advantages for fine-grained optimization.

Path-specific local advantage. While the global advantage supervises the overall outcome of a response, it cannot distinguish correctness differences among the reasoning paths within the same response. We therefore introduce **Path-Specific Advantage Routing GRPO (PSAR-GRPO)**, which augments the global advantage with an independently constructed local advantage for each reasoning path. Let

$$\mathcal{R} = \{\mathrm{pos}, \mathrm{neg}, \mathrm{unk}\}$$

denote the Positive, Negative, and Unknown Paths, respectively. For path r in response i, we define the path-level local reward as

$$R_{i,r}^{\mathrm{L}} = \lambda_p R_{\mathrm{path}}(o_i^{(r)}, y) + \mathbb{I}[r \in \{\mathrm{pos}, \mathrm{neg}\}]\lambda_v R_{\mathrm{ver}}(o_i^{(r)}) + \lambda_c R_{\mathrm{cons}}(o_i^{(r)}, o_i), \tag{3}$$

where R_{path} measures the correctness of the reasoning state expressed by the current path, R_{ver} evaluates the validity of the corresponding proof in the Positive and Negative Paths according to Lean4 verification, and R_{cons} measures consistency among the path reasoning, proof outcome, and final decision. Since the Unknown Path contains no formal proof, its local reward does not include the verification term.

For a prompt u with rollout group $\mathcal{G}_u$, we further normalize the local rewards independently within each path type to obtain path-specific relative advantages:

$$A_{i,r}^{\mathrm{L}} = \begin{cases} \dfrac{R_{i,r}^{\mathrm{L}} - \mu_{u,r}}{\sigma_{u,r} + \epsilon}, & \sigma_{u,r} > 0, \\ 0, & \sigma_{u,r} = 0, \end{cases} \tag{4}$$

where $\mu_{u,r}$ and $\sigma_{u,r}$ denote the mean and standard deviation of the local rewards for path r within the rollout group, respectively. Through this path-wise relative normalization, the Positive, Negative, and Unknown Paths can receive independent local optimization signals according to their own reasoning correctness, proof validity, and consistency, without being directly determined by the correctness states of other paths in the same response. We then combine these path-specific local advantages with the global advantage and route them to their corresponding reasoning regions for policy optimization.

Policy objective. Finally, we combine the global advantage with the path-specific local advantages to construct the token-level advantage used for policy optimization. Let $M_{i,t,r} = 1$ if and only if token t belongs to the content span of path r in response i. We define

$$A_{i,t}^{\mathrm{PSAR}} = A_i^{\mathrm{G}} + \lambda \sum_{r \in \mathcal{R}} M_{i,t,r} A_{i,r}^{\mathrm{L}}, \tag{5}$$

where A_i^{G} is broadcast over the complete response, whereas the path-specific advantage $A_{i,r}^{\mathrm{L}}$ acts only on its corresponding reasoning region through the path mask. The complete response therefore remains supervised by the global task outcome, while individual reasoning paths receive distinct local corrections according to their own correctness states.

We then use the resulting token-level advantages in the clipped GRPO objective with KL regularization:

$$\mathcal{L}_{\text{PSAR-GRPO}}(\theta) = -\frac{1}{K}\sum_{i=1}^{K}\frac{1}{|\mathcal{T}_i|}\sum_{t\in\mathcal{T}_i}\min\Big(\rho_{i,t}(\theta)A_{i,t}^{\mathrm{PSAR}}, \\ \mathrm{clip}(\rho_{i,t}(\theta), 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}})\, A_{i,t}^{\mathrm{PSAR}}\Big) + \beta D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}}), \tag{6}$$

where

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid x, o_{i,<t})}{\pi_{\theta_{\mathrm{old}}}(o_{i,t} \mid x, o_{i,<t})}. \tag{7}$$

Here, $\mathcal{T}_i$ denotes the set of valid tokens in response i. Overall, PSAR-GRPO unifies response-level global outcome supervision and path-level local feedback within a single policy objective, while routing fine-grained learning signals to their corresponding reasoning paths.

3.3 Verifier-Guided Iterative Refinement

Iterative proof refinement. Even after structured policy optimization, the model may still produce invalid or incomplete formal proofs at inference time. To make verifier feedback useful not only for judging proof correctness but also for repairing failed reasoning, we introduce a verifier-guided iterative refinement mechanism at test time. For a proof rejected by Lean4, the model uses the current proof together with the returned error messages and proof state as an updated reasoning context. It then revises the proof accordingly and resubmits the updated result to Lean4 for verification.

Adaptive decision and fallback. Let S_t denote the reasoning context at refinement step t. For the Positive and Negative Paths that require formal verification, the model selects

$$a_t \sim \pi_\theta(\cdot \mid S_t), \qquad a_t \in \{\textsc{Prove}(P), \textsc{Prove}(\neg P), \textsc{NoProof}\}. \tag{8}$$

For proof actions, let $G(a_t) = P$ when $a_t = \textsc{Prove}(P)$ and $G(a_t) = \neg P$ when $a_t = \textsc{Prove}(\neg P)$. Lean4 verifies the current proof attempt and returns

$$F_t = \mathcal{V}(\Gamma, G(a_t), a_t) \in \{\textsc{Success}, \textsc{Progress}, \textsc{Error}\}, \qquad a_t \neq \textsc{NoProof}. \tag{9}$$

Here, Success indicates that the proof is accepted, Progress denotes an incomplete proof with a proof state that can support further refinement, and Error indicates a formal error. For an unsuccessful proof attempt, the model updates its reasoning context using F_t and continues refinement, allowing up to three additional repairs after the initial attempt. If the model selects NoProof, or no valid proof is obtained within the given budget, the system falls back to the final three-way prediction generated from the current three reasoning paths.

Algorithm 1 Overall Procedure of VeriLogic

Require: Dataset $\mathcal{D}$, policy π_θ, verifier $\mathcal{V}$, group size K, refinement budget $N_{\max}$
Ensure: Trained policy π_θ and prediction $\hat{y}$

```
D_F ← {(Γ, P, y) : (Γ, P) = τ(C, q), (C, q, y) ∈ D}    ▷ Grounding
repeat
    Sample K responses with Positive, Negative, and Unknown Paths from π_θ
    Verify path proofs with V and compute A_i^G, A_{i,r}^L, and A_{i,t}^PSAR
    Update π_θ using Eq. equation 6    ▷ Optimization
until training budget is exhausted
(Γ, P) ← τ(C_test, q_test); generate three reasoning paths and initialize S_1
for t = 1, ..., N_max do
    Refine unresolved Positive/Negative proofs using Lean4 feedback and update S_{t+1}    ▷ Refinement
end for
return ŷ from the refined reasoning paths
```

Table 1: **Main results under the Lean4-formalized setting.** Accuracy is reported for each dataset, and Avg. denotes the average over six benchmarks. ↑ indicates higher is better. Δ reports VeriLogic minus the strongest corresponding baseline in each column. Averages and differences are computed before rounding.

Model	Method	MultiLogic	ProofWriter	FOLIO	ProverQA	LogiQA	e-SNLI	Avg. ↑
Qwen3-8B	Base	92.67	85.33	88.00	89.33	91.67	93.00	90.00
	GRPO	94.67	88.00	90.50	91.00	95.67	96.33	92.70
	LogicReward	95.67	92.67	92.50	94.33	96.00	97.00	94.70
	LPT	96.00	89.33	90.50	90.67	96.33	98.00	93.47
	VeriLogic (ours)	**96.67**	**96.67**	**94.50**	**95.33**	**96.67**	**99.33**	**96.53**
	Δ	+0.67	+4.00	+2.00	+1.00	+0.34	+1.33	+1.83
Llama-3.1 8B	Base	55.33	45.67	57.00	48.33	58.67	74.00	56.50
	GRPO	59.67	49.33	61.00	52.67	63.33	80.33	61.06
	LogicReward	56.33	52.33	64.00	60.67	60.00	81.00	62.39
	LPT	60.33	51.67	60.50	57.67	61.33	77.67	61.53
	VeriLogic (ours)	**61.00**	**60.67**	**68.00**	**62.67**	**64.67**	**87.33**	**67.39**
	Δ	+0.67	+8.33	+4.00	+2.00	+1.33	+6.33	+5.00
Qwen3-4B	Base	81.67	69.67	73.50	69.33	83.67	82.33	76.70
	GRPO	85.67	74.33	77.50	74.33	88.00	86.33	81.03
	LPT	88.00	79.67	80.00	78.00	90.00	88.67	84.06
	VeriLogic (ours)	**92.33**	**94.00**	**88.00**	**82.67**	**93.67**	**92.00**	**90.44**
	Δ	+4.33	+14.33	+8.00	+4.67	+3.67	+3.33	+6.39

3.4 End-to-End Procedure

Algorithm 1 summarizes the complete VeriLogic procedure following the unified "Grounding–Optimization–Refinement" framework. Natural-language problems are first transformed into verifiable instances with explicit reasoning paths and proof targets. The policy is then optimized over on-policy rollouts using global and path-specific advantages, while verifier feedback is further used at inference time to iteratively refine unsuccessful proofs.

4 Experiments

4.1 Experimental Settings

Dataset. We evaluate VeriLogic on six logical reasoning benchmarks under a unified `True/False/Unknown` setting, covering diverse forms of rule-based and natural-language logical inference. Dataset preprocessing, split details, and statistics are provided in Appendix B.

Baselines. We compare VeriLogic with Base (the original backbone), GRPO, LogicReward (Xu et al., 2026), and Logical Phase Transitions (LPT) (Zhang et al., 2026a), when available for the corresponding backbone (Appendix B.3).

Settings. We train Qwen3-8B and Qwen3-4B (Yang et al., 2025), and Llama-3.1-8B (Grattafiori et al., 2024) using LoRA fine-tuning (Hu et al., 2021), and report three-way answer accuracy. At test time, VeriLogic performs up to three additional feedback-guided repairs for unsuccessful proof attempts. Optimization details, decoding parameters, and dataset statistics are provided in Appendix B.

4.2 Main Results

As shown in Table 1, VeriLogic achieves the highest average accuracy across all three backbones, outperforming the strongest corresponding baselines by 1.83, 6.39, and 5.00 percentage points on Qwen3-8B, Qwen3-4B, and Llama-3.1-8B, respectively. These results demonstrate consistent gains across different model families and scales. The Non-CoT evaluation in Appendix C also yields average improvements, although the gains vary across datasets.

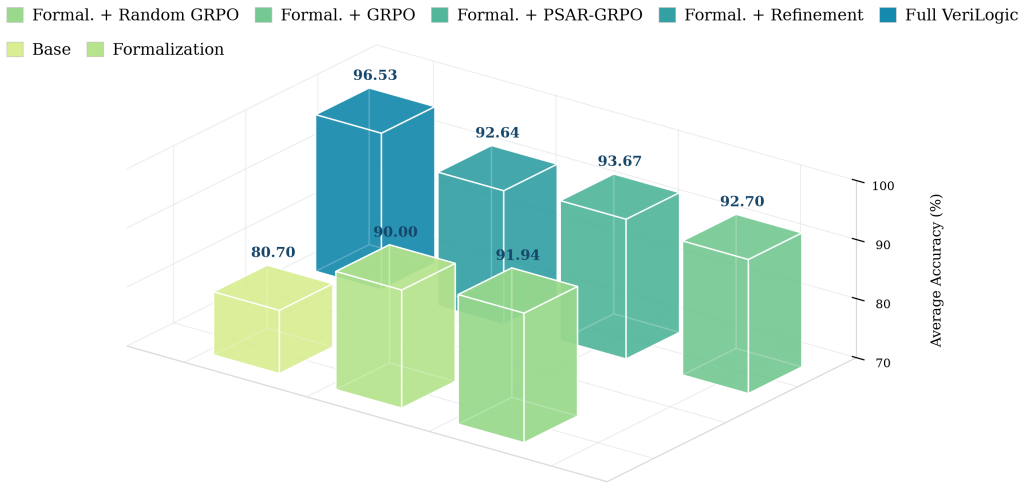


Figure 3: **Component ablation of VeriLogic on Qwen3-8B.** Macro-averaged accuracy over six logical reasoning datasets under formalization, GRPO, PSAR-GRPO, verifier-guided refinement, and full VeriLogic configurations, together with matched controls.

4.3 Ablation Study

Ablation setup. We isolate the contributions of verifiable formalization, PSAR-GRPO, and verifier-guided iterative refinement on Qwen3-8B. The ablation follows the dependency structure of the framework, with test-time refinement evaluated only on formalized inputs. All settings use the same 1,700-example evaluation set as the main experiments: 300 examples per dataset, except FOLIO with 200. We report macro-averaged accuracy over the six datasets.

Component analysis. Figure 3 summarizes the results on Qwen3-8B. **Formalization** provides the largest single-component gain, increasing average accuracy from 80.70% to 90.00%. **Formalization + GRPO** reaches 92.70% with the same formalized inputs and global response-level reward, whereas the matched random-reward control reaches 91.94%. **PSAR-GRPO** further improves performance to 93.67%, showing that path-specific local advantages over the Positive, Negative, and Unknown Paths provide finer-grained optimization beyond the global outcome advantage. **Verifier-guided refinement** reaches 92.64% with Lean4 feedback. The complete **VeriLogic** system achieves 96.53%, a 2.86-point gain over the strongest incomplete configuration, further demonstrating the complementarity of the three components.

4.4 Analysis

We further analyze complexity robustness, reward validity, path-specific credit assignment, and iterative verifier feedback.

Q1: Complexity–accuracy robustness. Following the LoCM definition of Zhang et al. (Zhang et al., 2026a), we partition test instances into eight complexity quantile bins and report the average accuracy in each bin. Figure 4 shows how performance changes as problem complexity increases. We conduct this analysis on the test set provided by LoCM, rather than the test set used in our main experiments.

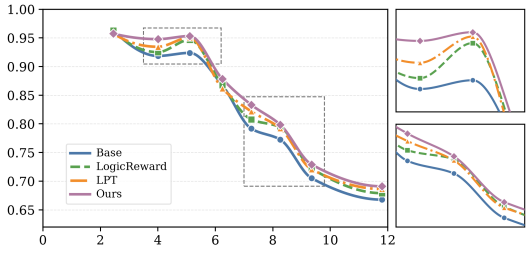


Figure 4: **LoCM complexity curves.**

All methods exhibit noticeable accuracy drops as LoCM increases, indicating that more complex logical operator compositions and longer reasoning chains remain challenging. VeriLogic achieves

the highest or tied-highest accuracy in every complexity bin and maintains comparatively more stable performance in the higher-complexity regions where degradation is more pronounced.

Q2: Task-grounded global reward. Under the same formalized inputs, initialization, sampling setup, and optimization budget, **Formalization + GRPO** uses the task-grounded global response-level reward, while **Formalization + Random GRPO** replaces it with independently sampled rewards $\tilde{R}_i \sim U[0,1]$. As shown in Figure 3, the two settings achieve 92.70% and 91.94% average accuracy, respectively, a difference of 0.76 percentage points. This comparison shows higher accuracy with task-grounded global rewards in the reported setting, suggesting that the improvement comes not only from policy optimization itself, but also from reward signals that are aligned with the target task. Although the margin is modest, it consistently favors task-grounded supervision under the matched training setup.

Q3: Path-specific advantage routing. We compare the correct path-specific advantage routing in VeriLogic with several alternative assignments. Figure 3 places the routing result within the overall component ablation, while Table 2 further compares different routing strategies. All settings share the same structured outputs, reward definitions, and training budget. **Correct routing** applies the local advantages of the Positive, Negative, and Unknown Paths to their corresponding reasoning paths. **Swapped routing** exchanges the local advantages of the Positive and Negative Paths, **full-response broadcast** applies each path-level local advantage to the entire response, and **random-region routing** assigns each local advantage to a random token span with the same length as its intended path.

Routing	Avg. (%)	Δ
Base (GRPO)	**92.70**	–
Swapped	72.75	−19.95
Full-response	90.76	−1.94
Random-region	86.86	−5.84
Correct routing (ours)	**96.53**	+3.83

Table 2: **Different routing strategies.**

As shown in Table 2, correct routing achieves 96.53% average accuracy, compared with 90.76% for full-response broadcast, 86.86% for random-region routing, and 72.75% for swapped routing. These results indicate that local advantages must not only provide finer-grained supervision, but also be correctly aligned with their corresponding reasoning paths.

Q4: Iterative verifier feedback. As shown in Figure 5, the formalization pass rate increases from 58.0% to 99.6% across verifier-guided repair rounds, while the proof pass rate rises from 42.3% to 76.4%. The two curves measure formalization validity and successful proof construction, respectively. Lean4 error messages and proof states are fed back to guide subsequent repairs, yielding consistent gains at both stages. Formalization mainly requires correcting declarations, types, and target statements, and therefore saturates quickly. Proof construction remains more challenging, as the model must further resolve remaining goals and synthesize a complete proof. The gap between the two curves shows that verifier feedback supports both formalization repair and proof refinement while exposing distinct failure modes.

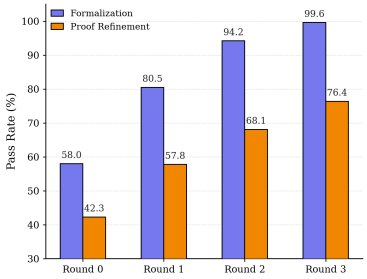


Figure 5: **Verifier feedback by repair round.**

5 Conclusion

This study introduces **VeriLogic**, a reinforcement learning framework that connects natural-language logical reasoning with Lean4 verification. VeriLogic uses **Verifiable Reasoning Environment Construction** to turn internal reasoning into explicit and verifiable reasoning paths, **Logical Structured Reasoning Optimization** to combine global supervision over the complete response with path-specific local feedback, and **Verifier-Guided Iterative Refinement** to iteratively repair unsuccessful proofs using Lean4 proof states and error feedback. Experiments on six logical reasoning benchmarks and three backbone models show consistent improvements in reasoning accuracy, demonstrating the effectiveness of explicit verifiable reasoning, fine-grained path-level optimization, and verifier-guided refinement.

Reproducibility Statement

All information necessary to reproduce our experiments can be found in Appendices A–D and the supplementary material.

AI Use Statement

Generative AI tools were used to convert natural-language logical problems into Lean4 formalizations and to generate candidate reasoning and proof steps for dataset construction. Training and test instances were retained after Lean4 formalization checks, with an additional proof check for `True` and `False` instances. AI tools were also used to assist with writing and language polishing; detailed usage is described in the paper.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Andrea Brunello, Cristian Curaba, Luca Geatti, Michele Mignani, Angelo Montanari, and Nicola Saccomanno. Fixing folio and malls: Verified annotations and an llm-assisted framework to focus human relabeling. *arXiv preprint arXiv:2606.02837*, 2026.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: Natural language inference with natural language explanations. *Advances in neural information processing systems*, 31, 2018.

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*, 2025.

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.

Herbert B Enderton. *A mathematical introduction to logic*. Elsevier, 2001.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

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

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, 2024.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, et al. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*, 2025.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In *Findings of the association for computational linguistics: ACL 2023*, pp. 1049–1065, 2023.

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.

Xingguang Ji, Yahui Liu, Qi Wang, Jingyuan Zhang, Yang Yue, Rui Shi, Chenxi Sun, Fuzheng Zhang, Guorui Zhou, and Kun Gai. Leanabell-prover-v2: Verifier-integrated reasoning for formal theorem proving via reinforcement learning. *arXiv preprint arXiv:2507.08649*, 2025.

Albert Q Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283*, 2022.

Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng Lyu, Kevin Blin, Fernando Gonzalez Adauto, Max Kleiman-Weiner, Mrinmaya Sachan, et al. Cladder: Assessing causal reasoning in language models. *Advances in Neural Information Processing Systems*, 36: 31038–31065, 2023.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. Lean-star: Learning to interleave thinking and proving. In *International Conference on Learning Representations*, volume 2025, pp. 66041–66062, 2025a.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, et al. Goedel-prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*, 2025b.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, et al. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. In *International Conference on Learning Representations*, volume 2026, pp. 11793–11818, 2026.

Hanmeng Liu, Zhizhang Fu, Mengru Ding, Ruoxi Ning, Chaoli Zhang, Xiaozhang Liu, and Yue Zhang. Logical reasoning in large language models: A survey. *arXiv preprint arXiv:2502.09100*, 2025a.

Jiacai Liu, Chaojie Wang, Chris Liu, Liang Zeng, Rui Yan, Yiwen Sun, and Yang Liu. Dapo: Improving multi-step reasoning abilities of large language models with direct advantage-based policy optimization. *Advances in Neural Information Processing Systems*, 38:71445–71477, 2026.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

Tongxuan Liu, Wenjiang Xu, Weizhe Huang, Yuting Zeng, Jiaxing Wang, Xingyu Wang, Hailong Yang, and Jing Li. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10168–10185, 2025b.

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

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, 2023.

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, 2023.

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, 2024.

Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. Multi-logieval: Towards evaluating multi-step logical reasoning ability of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20856–20879, 2024.

Chengwen Qi, Ren Ma, Bowen Li, Binyuan Hui, Jinwang Wu, Yuanjun Laili, Conghui He, et al. Large language models meet symbolic provers for logical reasoning evaluation. In *International Conference on Learning Representations*, volume 2025, pp. 54186–54205, 2025.

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 *International Conference on Learning Representations*, volume 2025, pp. 24935–24964, 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.

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

Peter Smith. *An introduction to formal logic*. Cambridge University Press, 2003.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Lean copilot: Large language models as copilots for theorem proving in lean. *arXiv preprint arXiv:2404.12534*, 2024.

Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. Determlr: Augmenting llm-based logical reasoning from indeterminacy to determinacy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9828–9862, 2024.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the association for computational linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

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.

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.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 2609–2634, 2023.

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, 2024.

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

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

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*, 2025.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*, 2024.

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, 2024.

Jundong Xu, Hao Fei, Meng Luo, Qian Liu, Liangming Pan, William Yang Wang, Preslav Nakov, Mong-Li Lee, and Wynne Hsu. Aristotle: Mastering logical reasoning with a logic-complete decompose-search-resolve framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3052–3075, 2025.

Jundong Xu, Hao Scofield Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Wang, Mong-Li Lee, and Wynne Hsu. Logicreward: Incentivizing llm reasoning via step-wise logical supervision. In *International Conference on Learning Representations*, volume 2026, pp. 43490–43511, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

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

Xinglang Zhang, Yunyao Zhang, ZeLiang Chen, Junqing Yu, Wei Yang, and Zikai Song. Logical phase transitions: Understanding collapse in llm logical reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18836–18860, 2026a.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*, 2023.

Yunyao Zhang, Xinglang Zhang, Junxi Sheng, Wenbing Li, Junqing Yu, Yi-Ping Phoebe Chen, Wei Yang, and Zikai Song. Semantic-aware logical reasoning via a semiotic framework. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18349–18374, 2026b.

Hongyu Zhao, Kangrui Wang, Mo Yu, and Hongyuan Mei. Explicit planning helps language models in logical reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 11155–11173, 2023.

Congmin Zheng, Jiachen Zhu, Zhuoying Ou, Yuxiang Chen, Kangning Zhang, Rong Shan, Zeyu Zheng, Mengyue Yang, Jianghao Lin, Yong Yu, et al. A survey of process reward models: From outcome signals to process supervisions for large language models. *arXiv preprint arXiv:2510.08049*, 2025.

Zi'ou Zheng, Christopher Malon, Martin Renqiang Min, and Xiaodan Zhu. Exploring the role of reasoning structures for constructing proofs in multi-step natural language reasoning with large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15299–15312, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Appendix

Appendix Overview

A. Methodological Derivations

B. Experimental Details

C. Additional Analysis

D. Case Study and Prompts

A Methodological Derivations

This appendix supplements the Grounding–Optimization–Refinement pipeline of VeriLogic. It clarifies the three reasoning paths, the separation of global and local feedback, and the interpretation of verifier-guided refinement, using the notation of the main text.

A.1 Formal Grounding and Three-Path Semantics

The mapping $\tau(C, q) = (\Gamma, P)$ in Eq. equation 1 represents the natural-language premises and query in Lean4. The template specifies entities, predicates, facts, and rules; auxiliary declarations supply the types and definitions required for elaboration. The Positive and Negative Paths share this premise context and use the respective proof targets P and $\neg P$. The Unknown Path reasons about insufficient support for either conclusion and has no formal proof target.

Under the consistent-premise convention, the intended label semantics are

$$y = \begin{cases} \texttt{True}, & \Gamma \vdash P, \\ \texttt{False}, & \Gamma \vdash \neg P, \\ \texttt{Unknown}, & \Gamma \nvdash P \ \wedge \ \Gamma \nvdash \neg P. \end{cases} \tag{10}$$

These semantics distinguish a logical state from the outcome of a particular proof attempt. Lean4 acceptance certifies the submitted proposition under Γ; a rejected or incomplete proof does not establish non-derivability. In particular, two unsuccessful proof attempts do not certify `Unknown`. Verification also depends on the fidelity of τ: checking a formal proof alone does not establish that the premises faithfully translate the natural-language input.

Structured-output contract. A response contains exactly one Positive, Negative, and Unknown Path, in that order, followed by one final answer $\hat{y}_i \in \{\texttt{True}, \texttt{False}, \texttt{Unknown}\}$. Each path contains its reasoning and an explicit binary verdict $z_{i,r}$ indicating whether it endorses its associated label, with $y_{\text{pos}} = \texttt{True}$, $y_{\text{neg}} = \texttt{False}$, and $y_{\text{unk}} = \texttt{Unknown}$. Positive and Negative Paths each contain a separately extractable proof field, which may be empty when that path does not claim derivability; a claim of derivability requires a nonempty proof field. The Unknown Path has no proof field. Path boundaries must be unambiguous and nonoverlapping. Missing, duplicated, out-of-order, or unparseable required fields make the format invalid. These structural checks do not require the verdicts to agree with one another or with the final answer: such disagreements are evaluated by the local consistency reward.

A.2 Global and Path-Level Feedback

The global reward in Eq. equation 2 evaluates the final answer and the validity of the response structure. Its components are

$$R_{\text{ans}}(o_i, y) = \mathbb{I}[\hat{y}_i = y], \qquad R_{\text{fmt}}(o_i) = \mathbb{I}[o_i \in \mathcal{O}_{\text{valid}}], \tag{11}$$

where $\mathcal{O}_{\text{valid}}$ is defined by the structured-output contract in Appendix A.1. Format validity concerns the response structure; proof correctness is evaluated separately.

The local reward in Eq. equation 3 distinguishes three sources of feedback. R_{path} evaluates the logical state expressed by a path against the target label, R_{ver} evaluates its submitted proof, and

R_{cons} evaluates agreement among the path's reasoning, proof outcome, and final decision. For $r \in \{\text{pos}, \text{neg}\}$, write

$$V_{i,r} = \mathbb{I}[\mathcal{V}(\Gamma, G_r, \pi_{i,r}) = \textsc{Success}], \qquad G_{\text{pos}} = P, \quad G_{\text{neg}} = \neg P, \tag{12}$$

where $\pi_{i,r}$ is the proof submitted in that path. A missing, incomplete, or rejected proof supplies no successful verification evidence. The Unknown Path receives path-correctness and consistency feedback but no verification reward. Thus, correctness of the final answer does not by itself determine the local rewards, and failure to prove either polarity is not treated as a verified Unknown judgment.

Binary scoring specification. To make the reward terms explicit, define the gold path indicator $z_r^\star = \mathbb{I}[y = y_r]$ and the answer-induced indicator $\hat{z}_{i,r} = \mathbb{I}[\hat{y}_i = y_r]$. A binary realization of the local reward components is

$$\begin{aligned} R_{\text{path}}(o_i^{(r)}, y) &= \mathbb{I}[z_{i,r} = z_r^\star], \\ R_{\text{ver}}(o_i^{(r)}) &= V_{i,r}, \qquad r \in \{\text{pos}, \text{neg}\}, \\ R_{\text{cons}}(o_i^{(r)}, o_i) &= \mathbb{I}[z_{i,r} = \hat{z}_{i,r}]\, C_{i,r}, \\ C_{i,r} &= \begin{cases} \mathbb{I}[z_{i,r} = V_{i,r}], & r \in \{\text{pos}, \text{neg}\}, \\ 1, & r = \text{unk}. \end{cases} \end{aligned} \tag{13}$$

Thus, endorsing the gold path or rejecting a non-gold path earns path-correctness credit. A positive derivability claim is proof-consistent only with an accepted proof; an accepted proof contradicts a negative derivability verdict. A negative verdict with no accepted proof is merely non-conflicting, not certified non-derivability: its correctness is assessed separately against the gold label. The Unknown verdict is checked against the final answer for consistency and against the gold label for correctness, without any proof reward. These checks operate on explicit verdicts and proof outcomes; they do not independently verify every natural-language reasoning step.

For example, if $y = \texttt{True}$, $\hat{y}_i = \texttt{True}$, and $z_{i,\text{pos}} = 1$ but its proof fails, the Positive Path receives $R_{\text{path}} = 1$, $R_{\text{ver}} = 0$, and $R_{\text{cons}} = 0$. This separates a correct judgment from an unsupported proof claim. For an unparseable path, its local reward components and token mask are set to zero; an unparseable final answer receives zero answer and consistency rewards. Format violations always receive $R_{\text{fmt}} = 0$.

A.3 Path-Specific Normalization and Routing

For a rollout group $\mathcal{G}_u$ of size $K > 1$, the statistics for each path $r \in \{\text{pos}, \text{neg}, \text{unk}\}$ are

$$\mu_{u,r} = \frac{1}{K} \sum_{j \in \mathcal{G}_u} R_{j,r}^{\text{L}}, \qquad \sigma_{u,r} = \sqrt{\frac{1}{K-1} \sum_{j \in \mathcal{G}_u} (R_{j,r}^{\text{L}} - \mu_{u,r})^2}. \tag{14}$$

Global statistics are computed in the same way from R_j^{G}. Normalization compares rewards across responses within the same path type, rather than pooling different paths. As in Eq. equation 4, a path with zero reward variance receives zero local advantage.

The mask $M_{i,t,r}$ in Eq. equation 5 selects the content tokens of path r, including its associated proof when present. The final-answer tokens and other tokens outside the path spans receive only A_i^{G}. Consequently, local feedback changes the update within its corresponding path while global supervision remains active across the complete response. If $\lambda = 0$, or all local advantages vanish, the update reduces to the global-only objective with the same response-level reward.

The resulting token advantages are inserted directly into the clipped, KL-regularized objective in Eq. equation 6. Rewards, normalization statistics, and masks are held fixed during each policy update. This routing controls where local credit is applied; it does not imply independent model parameters or independent reasoning paths.

A.4 Verifier-Guided Refinement and Final Prediction

Following Section 3.3, inference allows one initial proof attempt and up to three additional repairs. For an unsuccessful Positive or Negative proof, the next context includes the current attempt together

with Lean4's error messages, remaining goals, and available hypotheses:

$$S_{t+1} = \mathrm{Concat}(S_t, a_t, F_t). \tag{15}$$

The statuses SUCCESS, PROGRESS, and ERROR retain the meanings given in Eq. equation 9. Only SUCCESS supplies an accepted proof; the other statuses provide information for revision.

An accepted proof supports the corresponding Positive or Negative conclusion. If the model selects NOPROOF, or the repair budget is exhausted without a valid proof, the final three-way prediction is generated from the current reasoning paths, as in Algorithm 1. Neither condition automatically implies `Unknown`. Final-answer accuracy and successful formal verification therefore measure distinct aspects of the output.

B EXPERIMENTAL DETAILS

This section summarizes the experimental settings used in our main experiments. It covers dataset preprocessing and splits, dataset statistics, optimization and generation settings, and baseline sources.

B.1 DATASETS, PREPROCESSING, AND SPLITS

We evaluate on six logical reasoning datasets: MultiLogic, which covers diverse logical forms and rule combinations (Patel et al., 2024); ProofWriter, which evaluates rule-based natural language reasoning with facts, rules, and proof structures (Tafjord et al., 2021); FOLIO, which provides natural language reasoning examples with first-order logic annotations (Han et al., 2024); ProverQA, a large-scale first-order logic reasoning dataset constructed with the assistance of symbolic provers (Qi et al., 2025); LogiQA, a reading-comprehension-style logical reasoning benchmark derived from examination scenarios (Liu et al., 2020); and e-SNLI, a natural-language reasoning dataset with entailment, contradiction, and neutral labels (Camburu et al., 2018). For FOLIO, we additionally refer to the verified annotation resources released by Brunello et al. (2026), which address annotation issues in the original dataset. These datasets are also connected to a broader line of logical-reasoning benchmarks and analyses that test proof planning, rule application, multi-step proof construction, negation, and systematic logical patterns (Saparov & He, 2022; Zhao et al., 2023; Zheng et al., 2024; Parmar et al., 2024; Jin et al., 2023).

Label normalization. All datasets are normalized into a unified three-way reasoning format:

$$\text{premises} + \text{question} \rightarrow \{\text{TRUE}, \text{FALSE}, \text{UNKNOWN}\}.$$

For datasets whose original label space does not explicitly include UNKNOWN, we use a stronger model to assist label normalization. This step separates refutable FALSE cases from underdetermined UNKNOWN cases, reducing the risk of mixing unverifiable or underspecified examples into the FALSE class.

Splits and retention. All six datasets are used for both training and evaluation, with randomly sampled training and evaluation subsets constructed from each dataset. Dataset statistics after preprocessing are shown in Table 3.

Training and evaluation use the same retention criteria: all instances must pass Lean4 formalization checks; `True` and `False` instances additionally require an accepted proof of P and $\neg P$, respectively, while `Unknown` instances are retained after formalization.

B.2 TRAINING CONFIGURATION

Optimization. The full VeriLogic model is optimized with PSAR-GRPO using LoRA fine-tuning on 2 H100 GPUs. Table 3 summarizes the dataset statistics and main training hyperparameters.

Table 3: Dataset statistics and training details of VeriLogic.

(a) Dataset sizes after preprocessing.

Dataset	e-SNLI	FOLIO	LogiQA	MultiLogic	ProofWriter	ProverQA	Total
Training examples	458	351	527	380	1,727	2,533	5,976
Test examples	300	200	300	300	300	300	1,700

(b) Training details.

Item	Setting
Backbone models	Qwen3-8B and Qwen3-4B; Llama-3.1-8B
Training algorithm	PSAR-GRPO with LoRA fine-tuning
LoRA configuration	Rank 16 / alpha 32 for all three backbones
LoRA target modules	All linear modules
Precision	bfloat16
Group size	8 sampled completions per prompt
Effective batch size	32
Optimization	3 epochs, learning rate 1×10^{-5}, $\beta = 0.04$
Generation	Temperature 0.7, maximum sequence length 8192, maximum completion length 4096

B.3 Baseline Details

Base denotes the original backbone model without fine-tuning. LogicReward uses publicly released weights when available. LPT is reproduced under our experimental setting. Missing baseline entries are omitted when no comparable checkpoint or setting is available for the corresponding backbone.

C Additional Analysis

This section evaluates whether the improvements from Lean4-guided training remain visible when explicit reasoning traces are removed from the model output. In the Non-CoT setting, the model directly generates the final three-way answer without generating intermediate reasoning traces.

Table 4: **Main results under the Lean4-formalized Non-CoT setting.** Accuracy is reported for each dataset, and Avg. is the average over the six reported benchmarks. ↑ indicates that higher values are better. Δ reports VeriLogic minus the strongest baseline for each column.

Model	Method	MultiLogic	ProofWriter	FOLIO	ProverQA	LogiQA	e-SNLI	Avg. ↑
Qwen3-8B	Base	65.33	64.67	63.50	59.67	72.00	89.00	69.03
	GRPO	68.33	68.00	67.00	63.00	75.67	92.67	72.44
	LogicReward	**69.67**	65.67	67.00	63.00	**78.00**	**93.67**	72.83
	LPT	68.67	67.33	**67.50**	62.67	76.00	92.67	72.47
	VeriLogic (ours)	68.67	**73.67**	67.00	**63.33**	76.00	93.00	**73.61**
	Δ	-1.00	+5.67	-0.50	+0.33	-2.00	-0.67	+0.78
Llama-3.1 8B	Base	29.67	46.33	39.50	40.67	33.00	51.67	40.14
	GRPO	32.33	50.67	43.00	44.67	36.33	55.00	43.67
	LogicReward	39.67	57.33	**50.50**	**48.33**	37.00	66.67	49.92
	LPT	40.33	54.00	44.50	45.33	33.00	63.67	46.81
	VeriLogic (ours)	**40.67**	**58.00**	49.50	46.67	**38.67**	**68.00**	**50.25**
	Δ	+0.33	+0.67	-1.00	-1.67	+1.67	+1.33	+0.33
Qwen3-4B	Base	62.33	59.67	60.50	53.33	72.33	84.67	65.47
	GRPO	66.67	63.67	64.50	57.33	**77.00**	**88.00**	69.53
	LPT	**67.33**	63.67	65.50	58.67	75.33	87.67	69.69
	VeriLogic (ours)	66.00	**67.00**	**67.00**	**59.67**	75.33	87.67	**70.44**
	Δ	-1.33	+3.33	+1.50	+1.00	-1.67	-0.33	+0.75

As shown in Table 4, VeriLogic achieves the best average accuracy for all three backbones in the Non-CoT setting, improving over the strongest corresponding baselines by 0.78, 0.33, and 0.75 points for Qwen3-8B, Llama-3.1-8B, and Qwen3-4B, respectively. These results show small average gains under answer-only evaluation, with larger improvements in the CoT setting where the model can express its structured reasoning.

D Case Study and Prompts

This appendix presents a compact formalization case and a simplified prompt template aligned with the Positive, Negative, and Unknown Paths of VeriLogic. The output fields follow Appendix A.1.

D.1 Compact End-to-End Lean4 Verification Case

This case study connects natural-language premises, the two Lean4 proof targets, and the three-path judgment leading to a final answer.

Compact End-to-End Lean4 Verification Case

Natural-language problem. Elian appreciates culture. Anyone who appreciates culture travels extensively. Anyone who travels extensively becomes experienced. If Elian is experienced, then he cannot both lack refinement and be sophisticated. Elian is sophisticated. Elian either lacks refinement or has good taste. Elian cannot have both good taste and bad taste.
Question. Is the statement "Elian has bad taste" true, false, or unknown?
Polarity targets.

$$P := badTaste(Elian), \qquad G^{+} = P, \qquad G^{-} = \neg P.$$

Lean4 instance and proof of the negative target.

```
axiom Entity : Type
axiom Elian : Entity

axiom appreciatesCulture : Entity -> Prop
axiom travelsExtensively : Entity -> Prop
axiom becomesExperienced : Entity -> Prop
axiom lacksRefinement : Entity -> Prop
axiom isSophisticated : Entity -> Prop
axiom goodTaste : Entity -> Prop
axiom badTaste : Entity -> Prop

axiom elianAppreciatesCulture :
  appreciatesCulture Elian

axiom cultureToTravel :
  (x : Entity) -> appreciatesCulture x -> travelsExtensively x

axiom travelToExperienced :
  (x : Entity) -> travelsExtensively x -> becomesExperienced x

axiom experiencedExclusive :
  becomesExperienced Elian ->
  ¬ (lacksRefinement Elian ∧ isSophisticated Elian)

axiom elianSophisticated :
  isSophisticated Elian

axiom refinementOrGoodTaste :
  lacksRefinement Elian ∨ goodTaste Elian

axiom tasteExclusive :
  ¬ (goodTaste Elian ∧ badTaste Elian)

theorem elian_not_bad_taste : ¬ badTaste Elian := by
  have hExp : becomesExperienced Elian :=
    travelToExperienced Elian
      (cultureToTravel Elian elianAppreciatesCulture)

  have hNoLack : ¬ lacksRefinement Elian := by
    intro hLack
    exact experiencedExclusive hExp
      ⟨hLack, elianSophisticated⟩

  have hGood : goodTaste Elian := by
```

```
    cases refinementOrGoodTaste with
    | inl hLack =>
        exact False.elim (hNoLack hLack)
    | inr hGood =>
        exact hGood

  intro hBad
  exact tasteExclusive ⟨hGood, hBad⟩
```

Three-path interpretation. The Positive Path is `unsupported`: the premises entail the negation of the query. The Negative Path is `supported`, with the proof above establishing $\neg P$. The Unknown Path is `unsupported`, since a definite negative conclusion is available. The empty Positive proof field does not itself establish non-derivability; the negative conclusion is supported by the constructed proof.

Final answer. The proof derives the negative target:

$$\Gamma \vdash \neg badTaste(Elian).$$

Thus the original statement $badTaste(Elian)$ is `False`:

$$\boxed{y = \texttt{False}}.$$

D.2 GRPO Prompt Template

The following simplified template specifies the three-path response structure used in the main text; it is not a verbatim training prompt. Each path states whether its associated label is supported. Proofs are nested within the Positive and Negative Paths, while the Unknown Path contains no proof field. The final answer is generated after considering all three paths.

Simplified Three-Path GRPO Prompt Template

System instruction.

```
You reason over a Lean4 premise context and a query P.
Assess three paths before producing a final answer:
- Positive: whether the premises support P (True).
- Negative: whether the premises support Not P (False).
- Unknown: whether neither conclusion is supported (Unknown).

For each path, write a short explanation and one verdict:
  supported | unsupported
The verdict endorses or rejects that path's associated label.
A failed proof attempt alone does not establish Unknown.

Proof requirements:
- Place each proof inside its own Positive or Negative Path.
- A supported Positive/Negative verdict requires a nonempty
  proof field targeting P/Not P, respectively.
- An unsupported Positive/Negative verdict may have an empty
  proof field. Never put a proof field in the Unknown Path.
- Use only the supplied premises and complete Lean4 proofs.
  Do not introduce additional axioms or use sorry or admit.
- Do not treat an unsuccessful attempt as verified evidence.

Output exactly one positive, negative, and unknown block,
in that order, followed by one answer. Replace placeholders
and choose one verdict and one final label; do not print the
alternative lists literally. Generate the final label by
considering the three paths and the available proof evidence.

<think>
<positive>
<verdict>supported | unsupported</verdict>
[Brief reasoning about P]
<proof>[Lean4 proof of P, or empty]</proof>
</positive>
<negative>
<verdict>supported | unsupported</verdict>
[Brief reasoning about Not P]
<proof>[Lean4 proof of Not P, or empty]</proof>
</negative>
<unknown>
<verdict>supported | unsupported</verdict>
[Brief reasoning about whether neither conclusion is supported]
```

```
</unknown>
</think>
<answer>{"label":"True" | "False" | "Unknown"}</answer>
```

User prompt.

```
Premise context:
<LEAN4_DECLARATIONS_AND_PREMISES>

Positive theorem statement: <POSITIVE_THEOREM_NAME> : <P>
Negative theorem statement: <NEGATIVE_THEOREM_NAME> : Not (<P>)

Assess the Positive, Negative, and Unknown Paths using this
shared context. Return the required structured response and
one final three-way answer. Complete a theorem with := by
and its proof when claiming that the corresponding goal is
supported.
```