LEARNING TO PROVE, NOT JUST TO ANSWER: REINFORCEMENT LEARNING FROM FORMAL VERIFICATION FOR NATURAL-LANGUAGE LOGICAL REASONING

Qili Zhang[1,2] **Qianren Mao**[1] **Hanze Cai**[2] **Kaiming Zhao**[2] **Yuening He**[2]
Xihan Lei[2] **Yashuo Luo**[2] **Hanwen Hao**[2] **Yutong Gu**[3] **Likang Xiao**[2]
Zhijun Chen[4] **Weifeng Jiang**[3] **Haoyi Zhou**[2] **Jianxin Li**[2]
[1]Zhongguancun Laboratory [2]Beihang University
[3]Nanyang Technological University [4]Hong Kong Polytechnic University
zhangql2023@mail.zgclab.edu.cn maoqr@zgclab.edu.cn

ABSTRACT

Large language models (LLMs) are increasingly deployed for natural-language logical reasoning, where the final answer is easy to check but the proof behind it is not. In natural-language logical reasoning, intermediate conclusion should follow from its premises, and the resulting derivation should support the final answer. Existing methods lack machine-checkable verification of intermediate conclusions and answer-supporting proof dependencies, so they may assign credit to invalid or answer-irrelevant steps. We propose **PROOF-R1**, a RL framework from formal verification that trains LLMs to construct verifiable proofs for natural-language logical reasoning. PROOF-R1 admits a generated conclusion into the verified proof state only when the corresponding reasoning action satisfie proof obligations through UNSAT-based machine-checkable formal verification. PROOF-R1 also recovers the answer-supporting dependency closure to trace the proof structure of the final answer and align outcome credit with the proof dependencies. Experiments demonstrate that PROOF-R1 improves answer accuracy across three logical reasoning benchmarks and four backbone models, and outperforms training-free agents and training-based methods in terms of reasoning-process verifiability.

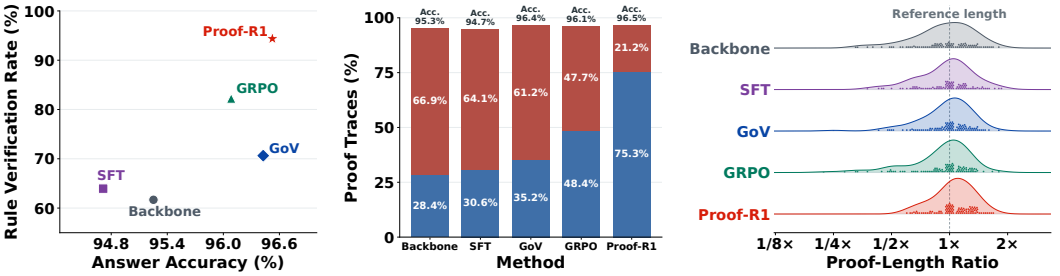


Figure 1: Performance of PROOF-R1 on ProverQA. **Left**: Accuracy and step verification. The x-axis shows answer accuracy, while the y-axis shows RVR reflecting the logical correctness of generated reasoning under schema, semantic, and rule checks. **Middle**: Performance of different methods. The total height of each bar represents the proportion of correct answers, while the blue segment indicates the portion whose proof traces pass formal verification. **Right**: Generated-to-reference proof-length ratios across methods on hard subset of ProverQA.

1 INTRODUCTION

Recent advances in large language models (LLMs) have demonstrated strong capabilities in multi-step reasoning and complex instruction following (Wei et al., 2022; Saparov et al., 2023; Dubey

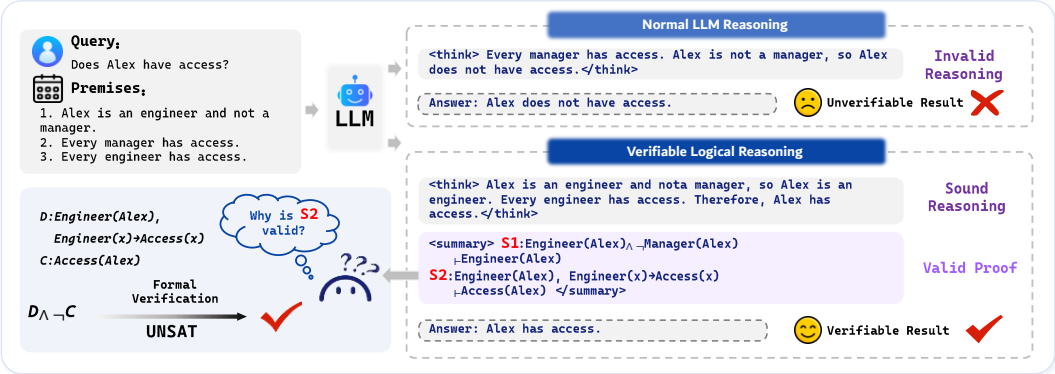


Figure 2: Illustrative contrast between normal LLM reasoning and verifiable logical reasoning. The normal LLM reasoning incorrectly infers that "Alex lacks access from not being a manager," committing the fallacy of denying the antecedent. The verifiable logical reasoning derives access from "Alex's engineer status" and exposes semantic proof obligation checked through formal verification.

et al., 2024; He et al., 2024; DeepSeek-AI, 2025; Yang et al., 2025). As shown in Figure 2, natural-language logical reasoning asks whether a target conclusion follows from premises expressed in both natural language and formal logic (Clark et al., 2020; Tafjord et al., 2021; Han et al., 2024). A verifiable solution is a finite derivation in which every intermediate conclusion follows from its declared premises alone under a valid rule, and the root conclusion establishes the target.

Post-training methods for natural-language logical reasoning commonly optimize answer-level objectives, with outcome-based reinforcement learning directly rewarding final-answer correctness (Shao et al., 2024; DeepSeek-AI, 2025; Xie et al., 2025). In parallel, symbolic solvers and proof assistants have been used to inspect model-generated reasoning, while formal verification feedback has been incorporated into supervised learning, preference learning, and reinforcement learning (Cao et al., 2025; Liu et al., 2025; Hubert et al., 2026; Fang et al., 2026).

Despite their successes, each approach has inherent limitations. Outcome rewards indicate whether a model reaches the correct answer but do not identify whether its intermediate deductions are valid (Shao et al., 2024; DeepSeek-AI, 2025; Xie et al., 2025). Existing methods lack machine-checkable verification of intermediate conclusions and answer-supporting proof dependencies, so they may assign credit to invalid or answer-irrelevant steps (Lightman et al., 2024; Wang et al., 2024; Xu et al., 2026; Liu et al., 2025). Some methods introduce formal verification to inspect generated steps or construct rewards (Liu et al., 2025; Feng et al., 2025; Chen et al., 2026; Fang et al., 2026). However, a verification verdict alone determines neither which generated conclusions serve as trusted facts for subsequent inferences nor which verified inferences actually support the final answer.

This raises our central research question: ***in natural-language logical reasoning, can formal verification do more than provide a reward for a proof trace and instead define the verifiable proof-construction process optimized by reinforcement learning?*** To address this question, we propose PROOF-R1, a RL framework from formal verification that trains LLMs to construct verifiable proofs for natural-language logical reasoning. The policy represents a proof trace as a sequence of structured reasoning actions, each specifying its dependencies, conclusion, and predicate-logic rule. For each candidate action, UNSAT-based *machine-checkable formal verification* (**MCFV**) determines whether its conclusion follows from the declared dependencies and can therefore serve as a trusted dependency for subsequent actions. Over the proof trace, PROOF-R1 constructs a candidate proof-certificate graph and recover the *answer-supporting dependency closure* (**ASDC**), thereby distinguishing answer-supporting actions from unused reasoning branches and making the proof basis of the final answer explicitly traceable. MCFV and ASDC yield *verification-aligned optimization*: formal verification no longer rewards, but determines which conclusions become trusted proof facts and which actions receive outcome credit during policy optimization. Our contributions are as follows:

- **Contribution 1** *(Novelty of* PROOF-R1*)*: To address missing machine-checkable verification for intermediate conclusions and proof dependencies, we propose PROOF-R1, a RL framework from formal verification that trains LLMs for natural-language logical reasoning.

- **Contribution 2** *(Performance of* PROOF-R1*)*: Across three logical reasoning benchmarks and four backbone models, PROOF-R1 improves answer accuracy and outperforms baselines in terms of reasoning-process verifiability.
- **Contribution 3** *(Mechanistic Insights into* PROOF-R1*)*: We analyze PROOF-R1 at both the behavioral and optimization levels: rollout statistics provide empirical support for the design of MCFV, training dynamics reveal how the policy progressively learns to suppress invalid inferences through ASDC, and parameter-gradient analysis explains how PROOF-R1 aligns optimization signals with verification evidence and proof dependencies.

2 RELATED WORK

Natural-language logical reasoning task is commonly formulated as determining the entailment relation between a set of textual premises and a query. In a verifiable solution, intermediate conclusions should follow from their premises and the resulting derivation should support the final answer (Schwichtenberg & Wainer, 2012; Tafjord et al., 2021). Benchmarks such as RuleTaker and ProofWriter evaluate multi-step deduction over textual facts and rules, with ProofWriter additionally providing explicit proof targets (Clark et al., 2020; Tafjord et al., 2021). PrOntoQA uses synthetic problems with recoverable proof structures to evaluate CoT reasoning and proof-planning abilities (Saparov & He, 2023). FOLIO and ProverQA broaden benchmark coverage to more complex and diverse first-order-logic problems through human annotation and prover-guided scalable generation, respectively (Han et al., 2024; Qi et al., 2025). Conventional approaches to these tasks include direct prediction(Clark et al., 2020) and supervised proof generation(Tafjord et al., 2021).

Formal verification-augmented LLM reasoning has emerged as a promising direction for improving both performance and reliability, as LLM capabilities advance and their deployment expands. Existing approaches differ in where formal verification enters the reasoning, verification, and learning process. At reasoning time, Logic-LM, LINC and LogicAgent use LLMs to formalize natural-language problems and delegate deduction to symbolic solvers (Pan et al., 2023; Olausson et al., 2023; Zhang et al., 2025). These reasoning-time pipelines improve problem solving, but do not provide a policy-learning objective for constructing the verified proof itself. A complementary line of work performs verification of generated reasoning. Graph of Verification (GoV) organizes LLM-based verification in a topologically ordered dependency graph with configurable granularity (Fang et al., 2026), while Safe and VeriCoT translate generated steps into Lean statements or first-order arguments for tool-based checks (Liu et al., 2025; Feng et al., 2025). For these methods, generation quality remains constrained by the backbone, and repeated verification calls can incur substantial token overhead. Formal feedback has entered learning through several distinct mechanisms. For natural-language reasoning, VeriCoT distills solver-validated traces into supervised and preference-training data.(Feng et al., 2025) LogicReward uses offline preference pairs generated by the policy and GPT-4o prior for supervised learning, while common theorem-proving reasoning datasets generally lack native pairwise preference annotations. (Xu et al., 2026). PRoSFI generates structured intermediates and aggregates their formal checks into a trajectory-level RL reward (Chen et al., 2026), whereas Logic-RL optimizes answer and format rewards without verifying intermediate deductions (Xie et al., 2025). These methods primarily use formal verification to construct training signals, rather than allowing it to determine during policy optimization which generated conclusions become trusted proof facts and which actions receive outcome credit.

3 METHODOLOGY

3.1 PROBLEM FORMULATION

A natural-language logical reasoning instance is $x = (\mathcal{P}, q, \mathcal{Y})$, where $\mathcal{P}$ contains premises, q is a query, and $\mathcal{Y} = \{y_1, \ldots, y_N\}$ is a finite set of candidate answer propositions. The premises in $\mathcal{P}$ are given both in natural language and as formal logical expressions. The task is to select a candidate entailed by $\mathcal{P}$ and construct a proof of that candidate. The reference answer $y^* \in \mathcal{Y}$ satisfies

$$\mathcal{P} \models y^*. \tag{1}$$

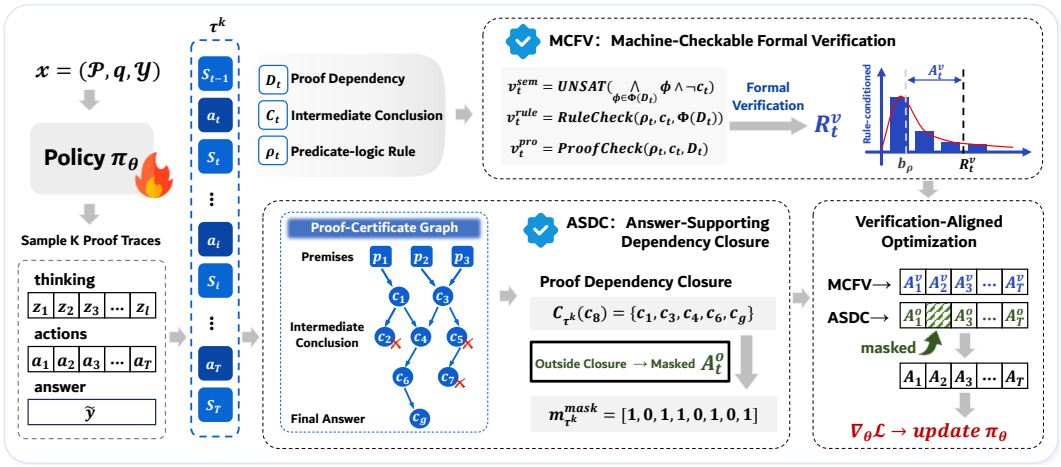


Figure 3: Framework of PROOF-R1. MCFV formally verifies structured reasoning actions and maintains verified proof states. ASDC maintains the candidate proof-certificate graph and constructs the answer-supporting dependency closure. Verification-aligned optimization integrates the supervision signals from MCFV and ASDC into the policy update.

Let $\mathcal{R}$ denote the trusted predicate-logic inference system. Π is a finite dependency-structured derivation of $\tilde{y}$ from $\mathcal{P}$ under $\mathcal{R}$. A successful proof trace therefore satisfies:

$$\tilde{y} = y^*, \qquad \mathcal{P} \vdash_{\mathcal{R}}^{\Pi} \tilde{y} \tag{2}$$

Equation 2 defines the proof-certificate derivability judgment used in our framework.

3.2 STRUCTURED PROOF GENERATION

As shown in Figure 3, PROOF-R1 links formal verification with proof-state transitions ($S_{t-1} \rightarrow S_t$) and proof dependencies (D_t) to train the policy to construct answer-supporting proofs. For each problem x, the policy π_θ samples K rollouts during training. The k-th rollout is

$$\tau^k = (z^k, a_1^k, \ldots, a_{T_k}^k, \tilde{y}^k), \tag{3}$$

where z^k is free-form analysis, T_k is the number of structured reasoning actions, and $\tilde{y}^k$ is the selected candidate.

A checkable proof step should specify both its conclusion and the inference claimed to justify it. We represent an ordinary step at the granularity of a single predicate-logic rule application:

$$a_t = (D_t, c_t, \rho_t), \tag{4}$$

where D_t contains dependency identifiers, c_t is the intermediate conclusion, and ρ_t is the declared predicate-logic rule. Each action a_t has a unique identifier i_t. The terminal action a_g matches the selected candidate, with conclusion $c_g = \tilde{y}$. In subsequent descriptions of individual traces, we omit the rollout index k to simplify the notation and make the equations easier to follow.

3.3 MACHINE-CHECKABLE FORMAL VERIFICATION (MCFV)

A generated conclusion should pass verification before it can support a subsequent verified inference. As the verifier checks actions in generation order, it maintains two maps from action identifiers to accepted conclusions: a semantically verified state $\mathcal{S}_t^{\text{sem}}$ and a stricter rule-verified state $\mathcal{S}_t^{\text{rule}}$. Both maps are initially empty, while the original premises $\mathcal{P}$ remain available throughout verification. Their pair $\mathcal{S}_t = (\mathcal{S}_t^{\text{sem}}, \mathcal{S}_t^{\text{rule}})$ is the aggregate proof state shown in Figure 3.

Semantic verification checks whether the declared dependencies entail the proposed conclusion. Before checking a_t, the verifier uses $\Phi_t^{\text{sem}}(\cdot)$ to resolve dependency identifiers into formulas against the original premises and the semantic state constructed from preceding actions:

$$\Phi_t^{\text{sem}}(D_t) = \left\{ \left(\mathcal{P} \cup \mathcal{S}_{t-1}^{\text{sem}}\right)(d) \;\middle|\; d \in D_t \right\}. \tag{5}$$

Here, premises and accepted conclusions are indexed by their identifiers, so $(\mathcal{P} \cup \mathcal{S})(d)$ denotes the formula named by dependency d; an identifier absent from this union is rejected as an untrusted dependency. The semantic proof obligation is discharged when a satisfiability modulo theories (SMT) solver returns UNSAT.

$$\mathrm{VC}_t^{\mathrm{sem}} = \bigwedge_{\phi \in \Phi_t^{\mathrm{sem}}(D_t)} \phi \wedge \neg c_t, \qquad v_t^{\mathrm{sem}} = \mathrm{UNSAT}\left(\mathrm{VC}_t^{\mathrm{sem}}\right). \tag{6}$$

A positive verdict establishes $\Phi_t^{\mathrm{sem}}(D_t) \models c_t$ and admits the conclusion to the semantic state:

$$\mathcal{S}_t^{\mathrm{sem}} = \begin{cases} \mathcal{S}_{t-1}^{\mathrm{sem}} \cup \{i_t \mapsto c_t\}, & v_t^{\mathrm{schema}} \wedge v_t^{\mathrm{sem}}, \\ \mathcal{S}_{t-1}^{\mathrm{sem}}, & \text{otherwise.} \end{cases} \tag{7}$$

Rule verification uses $\Phi_t^{\mathrm{rule}}(\cdot)$ to resolve dependency from the stricter rule-verified state:

$$\Phi_t^{\mathrm{rule}}(D_t) = \left\{ \left(\mathcal{P} \cup \mathcal{S}_{t-1}^{\mathrm{rule}}\right)(d) \;\middle|\; d \in D_t \right\}, \qquad v_t^{\mathrm{rule}} = \mathrm{RuleCheck}\left(\rho_t, \Phi_t^{\mathrm{rule}}(D_t), c_t\right). \tag{8}$$

RULECHECK validates the rule's principal premises, substitutions, and conclusion (Appendix F). A conclusion enters the rule-verified state only when its semantic and rule obligations are satisfied:

$$\mathcal{S}_t^{\mathrm{rule}} = \begin{cases} \mathcal{S}_{t-1}^{\mathrm{rule}} \cup \{i_t \mapsto c_t\}, & v_t^{\mathrm{schema}} \wedge v_t^{\mathrm{sem}} \wedge v_t^{\mathrm{rule}}, \\ \mathcal{S}_{t-1}^{\mathrm{rule}}, & \text{otherwise.} \end{cases} \tag{9}$$

PROOFCHECK checks whether an ordinary action makes nontrivial proof progress (Appendix H):

$$v_t^{\mathrm{pro}} = \mathrm{ProofCheck}(\rho_t, c_t, D_t). \tag{10}$$

The verifier retains the four Boolean verdicts:

$$v_t = (v_t^{\mathrm{schema}}, v_t^{\mathrm{sem}}, v_t^{\mathrm{rule}}, v_t^{\mathrm{pro}}) \in \{\mathrm{true}, \mathrm{false}\}^4. \tag{11}$$

During trace verification, each state is therefore both the output at step t and the verifier memory used to resolve dependencies at step $t+1$; downstream optimization consumes the resulting verdicts. The structured verdict v_t^k of a_t^k is mapped to a numerical verification signal:

$$R_{k,t}^v = \begin{cases} 0.0, & \neg v_t^{\mathrm{schema},k}, \\ 0.1, & v_t^{\mathrm{schema},k} \wedge \neg v_t^{\mathrm{sem},k}, \\ 0.3, & v_t^{\mathrm{schema},k} \wedge v_t^{\mathrm{sem},k} \wedge \left(\neg v_t^{\mathrm{rule},k} \vee \neg v_t^{\mathrm{pro},k}\right), \\ 1.0, & v_t^{\mathrm{schema},k} \wedge v_t^{\mathrm{sem},k} \wedge v_t^{\mathrm{rule},k} \wedge v_t^{\mathrm{pro},k}. \end{cases} \tag{12}$$

We center each verification signal using a rule-conditioned centering value estimated from the recent verification outcomes of actions that invoke the same predicate-logic rule. Let $\mathrm{EMA}_\rho^{(u-1)}$ denote the exponential moving average for rule ρ before update u. The clipped rule-conditioned centering value and action-verification advantage are:

$$b_\rho^{(u-1)} = \mathrm{clip}(\mathrm{EMA}_\rho^{(u-1)}, b_{\min}, b_{\max}), \qquad A_{k,t}^v = R_{k,t}^v - b_{\rho_t^k}^{(u-1)}. \tag{13}$$

For each rule observed in the current batch, let $\mathcal{B}_\rho^{(u)}$ contain the actions assigned to that rule. After computing the current advantages, we update:

$$\overline{R}_\rho^{v,(u)} = \frac{1}{|\mathcal{B}_\rho^{(u)}|} \sum_{a_t^k \in \mathcal{B}_\rho^{(u)}} R_{k,t}^v, \qquad \mathrm{EMA}_\rho^{(u)} = \beta\, \mathrm{EMA}_\rho^{(u-1)} + (1-\beta)\overline{R}_\rho^{v,(u)}. \tag{14}$$

3.4 ANSWER-SUPPORTING DEPENDENCY CLOSURE (ASDC)

A verified conclusion need not participate in the proof of the selected answer. To identify the declared answer derivation, we construct a verifier-annotated candidate proof-certificate graph $G_\tau = (V_\tau, E_\tau)$. Writing the original premises as $\mathcal{P} = \{p_j\}_{j=1}^M$, its nodes and edges are

$$V_\tau = \mathcal{P} \cup \{c_t\}_{t=1}^T, \qquad E_\tau = \{(u, c_t) \mid u \in \mathcal{P} \cup \{c_j\}_{j<t},\ \mathrm{id}(u) \in D_t\}. \tag{15}$$

Here $\mathrm{id}(p_j)$ is the identifier of an original premise and $\mathrm{id}(c_j) = i_j$ is the identifier of the action that generated c_j. The answer-supporting dependency closure rooted at the final conclusion c_g is the closed ancestor set:

$$\mathcal{C}_\tau(c_g) := \mathrm{Anc}_{G_\tau}(c_g) \cup \{c_g\} = \{u \in V_\tau : u \rightsquigarrow_{G_\tau} c_g\} \cup \{c_g\}, \tag{16}$$

where $\mathrm{Anc}_{G_\tau}(c_g)$ denotes the strict ancestors of c_g, and $\rightsquigarrow_{G_\tau}$ denotes directed reachability in G_τ. We compute the closure by traversing dependencies backward from c_g until all ancestor conclusions and premises have been included. The proof-certificate graph for the selected answer is the subgraph induced by this closure:

$$\Pi_\tau = G_\tau[\mathcal{C}_\tau(c_g)]\,. \tag{17}$$

Thus, Π_τ contains exactly the premise and generated-conclusion nodes declared to support the final answer, together with their dependency relations.

A correct answer does not make every action in its response part of the answer's proof. The closure mask maps conclusion membership back to the action that generated each conclusion:

$$m_{k,t} = \mathbb{I}[c_t^k \in \mathcal{C}_{\tau^k}(c_g^k)], \qquad \mathbf{m}^k = (m_{k,1}, \ldots, m_{k,T_k}). \tag{18}$$

For rollout k, the outcome signal is $R_k^o = \mathbb{I}[\tilde{y}^k = y^*]$. We normalize this signal over eligible rollouts for the same problem:

$$\widehat{A}_k^o = \frac{R_k^o - \mu_x}{\sigma_x + \epsilon}. \tag{19}$$

Here μ_x and σ_x are the group mean and standard deviation. The mask restricts outcome credit to actions that support the answer:

$$A_{k,t}^o = m_{k,t}\widehat{A}_k^o. \tag{20}$$

Consequently, an action receives no outcome advantage when its generated conclusion lies outside the closure, even when the final answer is correct.

3.5 Verification-aligned optimization

Verification-aligned optimization combines the action-verification advantage from MCFV with the closure-masked outcome advantage from ASDC, and aligns the resulting action-level advantages with the exact token spans of structured actions:

$$A_{k,t} = \lambda_v A_{k,t}^v + \lambda_o A_{k,t}^o, \qquad \mathbf{A}^{\mathrm{tok},k} = M^k \mathbf{A}^k. \tag{21}$$

Here $\mathbf{A}^k$ collects the action-level advantages, and M^k is the normalized action-to-token alignment matrix that maps each action advantage to its exact token span. We then optimize the policy with a KL-regularized PPO-style clipped objective (Schulman et al., 2017; Shao et al., 2024). And we provide the construction of M^k, full objective, and sampled KL estimator (Appendix A).

4 Experiments

4.1 Experimental Setup

Datasets and models. We train and evaluate PROOF-R1 on ProverQA (Qi et al., 2025). We evaluate two backbones without a native thinking mode, Qwen2.5-7B-Instruct (Yang et al., 2024) and Llama-3.1-8B (Dubey et al., 2024), and two backbones with native thinking support, Qwen3-8B (Yang et al., 2025) and GLM-Z1-9B-0414 (THUDM, 2025). We further assess cross-benchmark generalization on FOLIO (Han et al., 2024) and ProofWriter (Tafjord et al., 2021). Notably, the evaluated FOLIO subset contains valid derivations that require a strictly broader inference system, $\mathcal{R}_{\mathrm{FOLIO}} \supset \mathcal{R}_{\mathrm{train}}$, with universal and existential quantifier rule families (Appendix G).

Baselines. Comparisons of training-free agents include LogicAgent (Zhang et al., 2025) and GoV (Fang et al., 2026). LogicAgent uses an LLM to reason over natural-language problems and delegates deduction to symbolic solvers. GoV organizes LLM-based verification in a topologically ordered dependency graph with configurable granularity. The training-based methods comprise SFT, GRPO (Shao et al., 2024), and PRoSFI (Chen et al., 2026). PRoSFI is an RL method for natural-language logical reasoning.

Table 1: Results on ProverQA, reported as means followed by standard deviations in gray. ❄ and 🔥 denote training-free and training-based methods, respectively. **Bold** and underline denote the best and second-best means within each backbone model. ProverQA-Hard denotes the hard split of ProverQA, whereas ProverQA-Overall denotes the evaluation set across all difficulty levels.

Model	Method	ProverQA-Overall					ProverQA-Hard				
		Avg@3 ↑	Pass@3 ↑	FVR ↑	RVR ↑	RGD ↓	Avg@3 ↑	Pass@3 ↑	FVR ↑	RVR ↑	RGD ↓
Qwen2.5-7B-Instruct	❄ Backbone	44.79 0.25	68.43 1.78	67.64 0.68	18.91 1.09	0.88 0.017	27.14 0.58	51.76 3.54	56.59 1.08	9.84 0.43	1.36 0.022
	❄ LogicAgent	59.28 0.56	81.20 1.49	54.39 1.02	30.44 0.55	0.93 0.009	40.03 0.44	65.83 3.37	38.11 1.42	15.72 1.67	1.51 0.008
	❄ GoV	47.23 0.82	70.63 1.75	65.28 0.54	23.38 0.45	0.87 0.010	26.80 1.60	51.76 3.54	54.50 1.09	10.28 1.07	1.34 0.010
	🔥 SFT	47.04 0.73	70.78 1.75	66.39 0.55	19.94 0.86	0.80 0.018	30.82 1.17	53.77 3.54	57.59 0.88	9.49 0.83	1.15 0.028
	🔥 GRPO	47.28 1.33	76.36 1.63	68.09 0.84	12.50 0.66	1.02 0.013	23.45 1.17	49.75 3.56	48.85 1.72	6.03 0.51	1.63 0.027
	🔥 PRoSFI	59.57 0.68	84.21 1.04	62.59 0.55	15.22 0.97	0.81 0.009	38.61 0.77	65.58 2.76	47.84 1.18	8.34 0.80	1.18 0.013
	🔥 **PROOF-R1**	**62.80** 0.56	**87.08** 1.29	**70.17** 0.28	**41.19** 0.44	**0.70** 0.018	**44.89** 0.60	**76.88** 2.99	**58.74** 0.26	**19.38** 0.75	**1.05** 0.033
Qwen3-8B	❄ Backbone	95.25 0.53	98.83 0.21	81.82 0.36	61.68 0.11	0.33 0.011	91.12 1.31	96.50 0.50	81.12 0.91	56.40 1.03	0.40 0.026
	❄ LogicAgent	**96.87** 0.42	99.12 0.36	93.88 0.19	79.70 0.55	0.33 0.009	91.29 1.57	97.49 1.11	90.39 0.52	67.14 1.71	0.47 0.025
	❄ GoV	96.43 0.05	98.53 0.46	89.05 0.61	70.63 0.63	0.29 0.006	**92.96** 0.77	97.49 1.10	86.04 0.57	61.69 0.28	0.38 0.017
	🔥 SFT	94.71 0.37	99.12 0.36	83.00 0.06	63.94 1.01	0.33 0.008	89.11 0.67	97.99 0.99	78.99 1.44	55.97 1.93	0.41 0.019
	🔥 GRPO	96.08 0.44	**99.27** 0.33	94.22 0.32	82.23 0.55	0.31 0.007	90.28 1.47	97.99 0.99	91.06 0.32	73.26 0.70	0.39 0.026
	🔥 PRoSFI	96.23 0.13	98.83 0.33	95.13 0.26	89.28 0.38	0.29 0.006	92.13 0.50	97.99 0.99	92.88 0.53	84.92 0.69	0.37 0.011
	🔥 **PROOF-R1**	96.52 0.26	**99.27** 0.33	**96.56** 0.15	**94.36** 0.08	**0.28** 0.005	92.63 1.10	**99.00** 0.71	**93.55** 0.32	**90.22** 0.26	**0.36** 0.002

Evaluation metrics. We report Avg@3 and Pass@3 for answer correctness. We assess proof quality using *Formal Verification Rate* (FVR) (Xu et al., 2026), *Rule Verification Rate* (RVR) (Zhou et al., 2025), and *Reasoning Granularity Deviation* (RGD) (Gu et al., 2026)(Appendix B).[1]

4.2 RESULTS

Table 1 presents the main results of PROOF-R1 on ProverQA. On Qwen2.5-7B-Instruct, PROOF-R1 improves Avg@3, Pass@3, FVR, and RVR over the strongest Overall-split baseline for each metric by 3.23, 2.87, 2.08, and 10.75 points, respectively, while reducing RGD from 0.80 to 0.70. On the more challenging Hard split, it improves Avg@3, Pass@3, FVR, and RVR by 4.86, 11.05, 1.15, and 3.66 points, respectively, while reducing RGD from 1.15 to 1.05. On Qwen3-8B, answer accuracy is near saturation. The training-free agents LogicAgent and GoV attain slightly higher Avg@3 than PROOF-R1 on the Overall and Hard splits. Both methods obtain these marginal gains through multiple model calls for multi-stage reasoning, verification, and answer selection, which substantially increases inference-time token consumption (Appendix L). On the Overall-split, PROOF-R1 achieves an FVR of 96.56 and an RVR of 94.36, improving over the strongest baselines by 1.43 and 5.08 points, and obtains the lowest RGD of 0.28. The results indicate that *our method improves the formal-verification quality of proof steps while preserving answer accuracy*.

4.3 ABLATIONS

Table 2 examines the roles of MCFV and ASDC in PROOF-R1. Across both backbone models on ProverQA-Hard, the full method achieves the best result on every metric, outperforming both ablated variants. The w/o MCFV variant obtains a lower RGD than w/o ASDC on both backbones (1.08 vs. 1.34 and 0.37 vs. 0.40), indicating that ASDC guides the policy toward proof traces whose granularity more closely matches the reference proofs.

Table 2: Ablation results on ProverQA-Hard.

Model	Variant	Avg@3 ↑	Pass@3 ↑	FVR ↑	RVR ↑	RGD ↓
Qwen2.5-7B-Instruct	w/o MCFV	37.86	68.34	40.16	11.65	1.08
	w/o ASDC	41.37	71.86	44.18	16.69	1.34
	PROOF-R1	**44.89**	**76.88**	**58.74**	**19.38**	**1.05**
Qwen3-8B	w/o MCFV	90.28	98.49	82.52	57.73	0.37
	w/o ASDC	89.61	97.99	**93.55**	85.71	0.40
	PROOF-R1	**92.63**	**99.00**	**93.55**	**90.22**	**0.36**

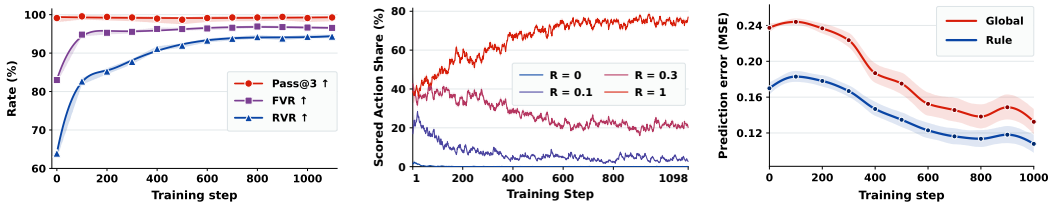
[1]Because FOLIO provides no reference proofs, whereas ProofWriter provides reference traces generated under a Horn inference system distinct from $\mathcal{R}_{\text{train}}$, RGD is not used as a common metric in Tables 4 and 6.

Table 3: Performance of PROOF-R1 trained with backbones from additional model families on ProverQA-Overall.

Model	Method	Avg@3 ↑	Pass@3 ↑	FVR ↑	RVR ↑	RGD ↓
Llama-3.1 8B	Backbone	46.30	76.54	50.11	9.20	0.94
	SFT	44.65	75.19	46.23	9.61	0.92
	PROOF-R1	**53.95**	**82.84**	**54.72**	**23.02**	**0.91**
GLM-Z1 9B-0414	Backbone	71.76	91.92	82.64	52.47	0.70
	SFT	74.40	96.48	83.03	57.23	0.65
	PROOF-R1	**89.33**	**99.27**	**95.01**	**87.26**	**0.42**

Table 4: Cross-dataset generalization of PROOF-R1 trained on ProverQA, tested on FOLIO and ProofWriter.

Dataset	Method	Avg@3 ↑	Pass@3 ↑	FVR ↑	RVR ↑
FOLIO	Backbone	88.89	**98.04**	64.46	38.87
	SFT	85.62	**98.04**	63.22	41.03
	PROOF-R1	**90.85**	**98.04**	**82.58**	**59.70**
Proof Writer	Backbone	80.89	90.67	77.07	51.78
	SFT	82.22	92.67	64.72	40.97
	PROOF-R1	**94.89**	**98.00**	**97.57**	**85.46**



(a) Performance during training **(b)** Verification-signal proportions **(c)** Centering-value error

Figure 4: Training diagnostics of PROOF-R1. Shaded regions indicate 95% confidence intervals. (a) Performance during training, measured by Pass@3, FVR and RVR of Qwen3-8B on ProverQA. (b) Proportions of each verification signal of Qwen3-8B. (c) Prediction error of the global and rule-conditioned centering values during training.

Conversely, the w/o ASDC variant achieves higher FVR and RVR than w/o MCFV on both backbones, showing that MCFV directly improves the semantic validity of generated proof steps.

Particularly, the relative answer accuracy of the two ablations reverses across backbones. On Qwen2.5-7B-Instruct, w/o ASDC outperforms w/o MCFV in Avg@3 (41.37 vs. 37.86), whereas on Qwen3-8B, w/o MCFV outperforms w/o ASDC (90.28 vs. 89.61). This pattern may suggest that *weaker models benefit more from improving the validity of proof steps, whereas stronger models benefit increasingly from distinguishing answer-supporting actions with irrelevant branches.*

4.4 DISCUSSION

Is PROOF-R1 effective across different backbone models? Table 3 shows that PROOF-R1 improves answer performance and rule-verified reasoning on Llama-3.1-8B and GLM-Z1-9B-0414. With Llama-3.1-8B, PROOF-R1 improves Avg@3 and Pass@3 over the backbone by 7.65 and 6.30 points, respectively. With GLM-Z1-9B-0414, it improves FVR and RVR over SFT by 11.98 and 30.03 points, respectively, while reducing RGD by 0.23. Across all four backbone models, PROOF-R1 *outperforms both the backbone model and SFT on all five metrics on ProverQA.*

Does PROOF-R1 generalize to a broader inference system? ProverQA proof traces are constructed under the training-time rule system $\mathcal{R}_{\text{train}}$. The evaluated FOLIO subset contains valid derivations that require a strictly broader system, $\mathcal{R}_{\text{FOLIO}} \supset \mathcal{R}_{\text{train}}$, adding universal and existential quantifier rule families absent from the training ontology (Appendix G). Table 4 reports the performance of ProverQA-trained models on FOLIO and ProofWriter. Notably, under $\mathcal{R}_{\text{FOLIO}}$, ProverQA-trained PROOF-R1 achieves an Avg@3 of 90.85, an FVR of 82.58, and an RVR of 59.70 on FOLIO, compared with 88.89, 64.46, and 38.87 for the backbone model. These results show that PROOF-R1 transfers beyond the rule ontology used to construct ProverQA. *The learned policy produces answer-supporting proofs that remain verifiable under a broader inference system.*

How does PROOF-R1 improve response quality? Figure 4(a) shows the model's performance on ProverQA throughout training. Complementing this, Figure 4(b) shows how the distribution of verification signals evolves during training. Answer accuracy remains high during training, while FVR increases to 96.56 and RVR increases to 94.36. Meanwhile, the proportion of actions receiving

the full verification signal ($R^v_{k,t} = 1.0$) increases, whereas the proportions receiving 0.1 or 0.3 decline, indicating failures in the semantic, rule, or proof-progress checks. This observation is consistent with the action-level supervision provided by MCFV. Even among samples with correct answers, *UNSAT-based verification and* RULECHECK *distinguish the semantic validity (measured by FVR) and rule faithfulness (measured by RVR) of individual proof steps, providing learning signals unavailable from answer correctness alone*.

Can rule-conditioned centering values account for differences in verification difficulty? Reasoning actions in formal verification have different rule constraints and dependency structures (Saparov et al., 2023). We investigate whether these structural properties characterize their verification difficulty? Figure 4(c) compares a global-mean centering value (labeled Global) with a rule-conditioned centering value (labeled Rule). We cross-validate both centering-value estimators on data unseen during fitting. The rule-conditioned centering value has lower prediction error than the global-mean centering value during training: at step 100, the error decreases from 0.24 to 0.18. These results show that *verification difficulty varies systematically with the logical operation performed by an action, rather than being uniform across actions*. This provides empirical support for the rule-conditioned centering value used to center verification signals in MCFV.

How do MCFV and ASDC jointly shape the optimization direction?

Figure 5 illustrates the gradient relationships between MCFV and ASDC across Transformer layers and parameter modules. We compute the gradients of the two supervision signals and construct the joint direction using their training weights. Figure 5(a) shows that the two components are nearly orthogonal in the early layers, with an angle of 96.6° between the concatenated gradients of layers 0–8. Directional competition increases with network depth, reaching 126.6° in the final layer. This shows that the two supervision signals do not simply reinforce the model along the same direction, but provide distinct optimization signals in parameter space (Yu et al., 2020). Therefore, answer support and formal correctness are not equivalent and require distinct supervision signals. Figure 5(b)(c) further shows the weighted joint direction. Across the 196 layer–module parameter blocks, its angle with MCFV is acute in every block, and its angle with ASDC is acute in 195 blocks. The weighted MCFV gradient norm is approximately 1.14 times that of ASDC, placing the two components at comparable magnitudes. These results provide mechanistic evidence for verification-aligned optimization: although MCFV and ASDC induce distinct and sometimes competing gradients, *verification-aligned optimization aligns their weighted contributions into a shared update direction that locally improves both objectives across almost all parameter blocks*.

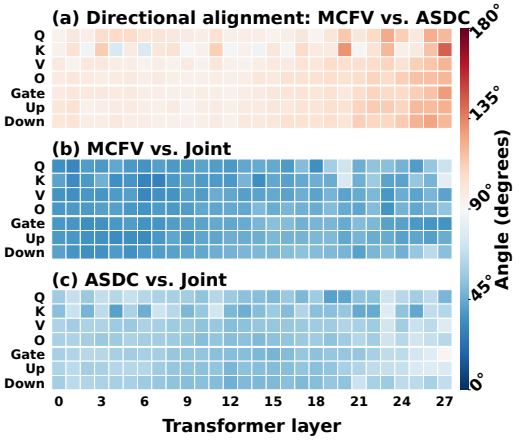


Figure 5: Layer-wise and module-wise gradient at training on Qwen2.5-7B-Instruct. Figures show angles between (a) MCFV and ASDC, (b) MCFV and the joint direction, and (c) ASDC and the joint direction.

5 LIMITATIONS

Since our main claim concerns verifiable natural-language logical reasoning, evaluating PROOF-R1 on English benchmarks with well-defined deductive structures necessarily covers only a controlled subset of natural-language reasoning. Such benchmarks necessarily abstract away some phenomena encountered in unrestricted natural language, including linguistic ambiguity, implicit background knowledge, and pragmatic interpretation. Moreover, while our evaluation covers multiple benchmarks, model families, and proof structures, it does not exhaust the broader reasoning landscape. Future work may investigate multilingual settings and additional logical formalisms.

6 CONCLUSION

We introduce PROOF-R1, a formal verification-driven reinforcement learning framework for verifiable natural-language logical reasoning. The framework combines UNSAT-based *machine-checkable formal verification* with an *answer-supporting dependency closure* to construct verified proof states and align policy learning with the final derivation. Experiments across multiple backbone models and benchmarks show that PROOF-R1 improves answer performance and verification pass rates for reasoning steps. This work integrates formal verification into proof construction and policy optimization on large language models, enabling models to perform verifiable proof-based reasoning over natural-language logical problems.

AI USE STATEMENT

In the preparation of this paper, GPT5.6-Sol was used exclusively for the purpose of polishing and refining the linguistic style of certain sentences to improve readability and fluency in section 2 and 5 and appendix E and J. As described in Appendix E, GPT-5.5 was used as an independent semantic reviewer to cross-check solver judgments. All core intellectual contributions are the product of the authors' own work, including but not limited to:

- Original ideas and novel viewpoints. Any innovative concepts, hypotheses, and critical analysis are the product of the authors' own work.
- Experimental results. All code, figures, findings, and interpretations thereof were generated, designed, collected, and analyzed by the authors.
- Substantive descriptions of the work. The description of the methodology, experimental procedures, and discussion of the research implications were entirely authored by the research team.

REPRODUCIBILITY STATEMENT

The datasets, backbone models, and baselines are described in Section 4, with metric definitions provided in Appendix B. The complete policy objective, verification specifications, inference systems, solver-validation procedures, implementation details, training configuration, computational cost, and prompt templates are documented in Appendices A–M. Source code is available at `https://anonymous.4open.science/r/Proof-R1/`.

REFERENCES

Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 13243 of *Lecture Notes in Computer Science*, pp. 415–442. Springer, 2022. doi: 10.1007/978-3-030-99524-9_24. URL `https://cvc5.github.io/papers/2022/BarbosaBBKLMMMN-TACAS22.pdf`.

Jialun Cao, Yaojie Lu, Meiziniu Li, Haoyang Ma, Haokun Li, Mengda He, Cheng Wen, Le Sun, Hongyu Zhang, Shengchao Qin, Shing-Chi Cheung, and Cong Tian. From informal to formal – incorporating and evaluating LLMs on natural language requirements to verifiable formal proofs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26984–27003. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1310. URL `https://aclanthology.org/2025.acl-long.1310/`.

Luoxin Chen, Yichi Zhou, and Huishuai Zhang. Learning to generate formally verifiable step-by-step logic reasoning via structured formal intermediaries. *arXiv preprint arXiv:2603.29500*, 2026. URL `https://arxiv.org/abs/2603.29500`.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp.

3882–3890, 2020. doi: 10.24963/ijcai.2020/537. URL https://arxiv.org/abs/2002.05867.

Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *Lecture Notes in Computer Science*, pp. 337–340. Springer, 2008. doi: 10.1007/978-3-540-78800-3_24. URL https://www.microsoft.com/en-us/research/publication/z3-an-efficient-smt-solver/.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Jiwei Fang, Bin Zhang, Changwei Wang, Jin Wan, and Zhiwei Xu. Graph of verification: Structured verification of LLM reasoning with directed acyclic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30665–30672, 2026. doi: 10.1609/aaai.v40i36.40322. URL https://ojs.aaai.org/index.php/AAAI/article/view/40322.

Yu Feng, Nathaniel Weir, Kaj Bostrom, Sam Bayless, Darion Cassel, Sapana Chaudhary, Benjamin Kiesl-Reiter, and Huzefa Rangwala. VeriCoT: Neuro-symbolic chain-of-thought validation via logical consistency checks. *arXiv preprint arXiv:2511.04662*, 2025. URL https://arxiv.org/abs/2511.04662.

Alex Gu, Bartosz Piotrowski, Fabian Gloeckle, Kaiyu Yang, and Aram H. Markosyan. ProofOptimizer: Training language models to simplify proofs without human demonstrations. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=huptrb4JTa.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander Fabbri, Wojciech Maciej Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22017–22031. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.1229. URL https://aclanthology.org/2024.emnlp-main.1229/.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10864–10882. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.637. URL https://aclanthology.org/2024.findings-emnlp.637/.

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. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabian Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli, and David Silver. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2026. doi: 10.1038/s41586-025-09833-y. URL https://doi.org/10.1038/s41586-025-09833-y.

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*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. Safe: Enhancing mathematical reasoning in large language models via retrospective step-aware formal verification. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12171–12186. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.594. URL https://aclanthology.org/2025.acl-long.594/.

Theo X. Olausson, Alex Gu, Benjamin Lipkin, Cedegao E. Zhang, Armando Solar-Lezama, Joshua B. 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. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.313. URL https://aclanthology.org/2023.emnlp-main.313/.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang 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. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.248. URL https://aclanthology.org/2023.findings-emnlp.248/.

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. *arXiv preprint arXiv:2502.06563*, 2025. URL https://arxiv.org/abs/2502.06563.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.01240.

Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Mehran Kazemi, Najoung Kim, and He He. Testing the general deductive reasoning capacity of large language models using OOD examples. In *Advances in Neural Information Processing Systems*, volume 36, pp. 3083–3105. Curran Associates, Inc., 2023. doi: 10.52202/075280-0136. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/09425891e393e64b0535194a81ba15b7-Abstract-Conference.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Helmut Schwichtenberg and Stanley S. Wainer. *Proofs and Computations*. Cambridge University Press, 2012. doi: 10.1017/CBO9781139031905. URL https://doi.org/10.1017/CBO9781139031905.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024. URL https://arxiv.org/abs/2409.19256.

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. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.317. URL https://aclanthology.org/2021.findings-acl.317/.

THUDM. GLM-Z1-9B-0414 model card, 2025. URL https://huggingface.co/THUDM/GLM-Z1-9B-0414.

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. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://arxiv.org/abs/2201.11903.

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. URL https://arxiv.org/abs/2502.14768.

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 *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2512.18196.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

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. URL https://arxiv.org/abs/2505.09388.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3fe78a8acf5fda99de95303940a2420c-Abstract.html.

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. *arXiv preprint arXiv:2509.24765*, 2025. URL https://arxiv.org/abs/2509.24765.

Ruiwen Zhou, Wenyue Hua, Liangming Pan, Sitao Cheng, Xiaobao Wu, En Yu, and William Yang Wang. RuleArena: A benchmark for rule-guided reasoning with LLMs in real-world scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 550–572. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.27. URL https://aclanthology.org/2025.acl-long.27/.

Appendix

Contents of Appendix

A Policy Optimization Implementation **14**

B Evaluation Metric Definitions **15**

C Additional ProverQA Results **16**

C.1 Easy and Medium Difficulty Splits of Main Results 16

C.2 Cross-Dataset Results with Qwen2.5-7B-Instruct 16

C.3 Hard-Split Results across Additional Backbone Models 17

D Conditional Soundness of the Verified Proof Certificate **17**

E Cross-Backend Verification and Independent Semantic Review **18**

F Rule Ontology **20**

G Evaluation of PROOF-R1 on Out-of-Distribution Datasets **21**

H Formal Specification and ProofCheck **22**

I Structured Output Contract **22**

J Implementation Details **23**

J.1 Method-Specific Implementations 23

J.2 Training Configuration 24

K Additional Training Diagnostics **25**

L Computational Cost **25**

M Prompt Template Used by PROOF-R1 **25**

N Case Study **28**

N.1 Case 1: A Negative Conclusion about Animal Care 28

N.2 Case 2: Verified Dependencies for a Correct Answer. 29

N.3 Case 3: Generating the Dependencies That Support the Answer 30

N.4 Case 4: Linking Verified Branches to the Answer 31

A Policy Optimization Implementation

Let L_k and T_k denote the response length and number of structured actions in rollout k, respectively. The normalized action-to-token alignment matrix used in Equation 21 is

$$M^k_{j,t} = \frac{\mathbb{I}[j \in \mathrm{Span}(a^k_t)]}{|\,\mathrm{Span}(a^k_t)|}, \qquad M^k \in \mathbb{R}^{L_k \times T_k}. \tag{22}$$

Thus, its t-th column distributes the advantage of action a^k_t uniformly over the token positions in $\mathrm{Span}(a^k_t)$ and assigns zero weight elsewhere. For token w^k_j with context $h^k_j = (x^k, w^k_{<j})$, the policy ratio is $r_{k,j}(\theta) = \pi_\theta(w^k_j \mid h^k_j)/\pi_{\theta_{\mathrm{old}}}(w^k_j \mid h^k_j)$, where $\mathcal{B}$ indexes rollouts sampled from the

old policy $\pi_{\theta_{\text{old}}}$. The optimized objective is the KL-regularized clipped loss

$$\mathcal{L}_{\text{clip}}(\theta) = -\frac{1}{N_{\mathcal{B}}} \sum_{k \in \mathcal{B}} \sum_{j=1}^{L_k} \min\Big\{ r_{k,j}(\theta) A_j^{\text{tok},k}, \\ \text{clip}(r_{k,j}(\theta), 1-\epsilon_-, 1+\epsilon_+)\, A_j^{\text{tok},k} \Big\} \\ + \frac{\lambda_{\text{KL}}}{|\mathcal{B}|} \sum_{k \in \mathcal{B}} \sum_{j=1}^{L_k} d_{k,j}(\theta). \tag{23}$$

Here ϵ_- and ϵ_+ are the clipping thresholds, $N_{\mathcal{B}}$ is the number of mapped actions, clamped to at least one, and λ_{KL} controls reference-policy regularization. The sampled KL penalty is

$$d_{k,j}(\theta) = \frac{\pi_{\text{ref}}(w_j^k \mid h_j^k)}{\pi_\theta(w_j^k \mid h_j^k)} - \log \frac{\pi_{\text{ref}}(w_j^k \mid h_j^k)}{\pi_\theta(w_j^k \mid h_j^k)} - 1, \tag{24}$$

where π_{ref} is the fixed reference policy.

Let $g(r, A) = \min\{rA, \text{clip}(r, 1-\epsilon_-, 1+\epsilon_+)A\}$ denote the surrogate term in Equation 23. The current optimizer additionally applies dual clipping to negative advantages, replacing g with

$$g_{\text{dual}}(r, A) = \begin{cases} \max\{g(r, A), c_{\text{dual}} A\}, & A < 0, \\ g(r, A), & A \geq 0, \end{cases} \qquad c_{\text{dual}} > 1. \tag{25}$$

The implemented policy loss is therefore

$$\mathcal{L}_{\text{impl}}(\theta) = -\frac{1}{N_{\mathcal{B}}} \sum_{k \in \mathcal{B}} \sum_{j=1}^{L_k} g_{\text{dual}}\Big(r_{k,j}(\theta), A_j^{\text{tok},k} \Big) + \frac{\lambda_{\text{KL}}}{|\mathcal{B}|} \sum_{k \in \mathcal{B}} \sum_{j=1}^{L_k} d_{k,j}(\theta). \tag{26}$$

Group-level and negative-mass controls are applied to action credit before the token projection. VERL sums token losses within each response and averages over responses, multiplying the projected advantages by $|\mathcal{B}|/N_{\mathcal{B}}$ therefore yields the action-normalized policy term above.

B Evaluation Metric Definitions

Let $\mathcal{D}_{\text{eval}}$ contain N evaluation problems, let K denote the number of sampled outputs per problem, and let $r_{i,k} \in \{0, 1\}$ indicate whether response k for problem i gives the correct answer. Avg@K and Pass@K for answer correctness is measured by

$$\text{Avg@}K = \frac{1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} r_{i,k}, \tag{27}$$

$$\text{Pass@}K = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[\sum_{k=1}^{K} r_{i,k} > 0 \right]. \tag{28}$$

We use $K = 3$ in all reported experiments.

For proof verification, let $\mathcal{E}_{\text{sem}}$ and $\mathcal{E}_{\text{rule}}$ denote the generated proof steps eligible for semantic and rule verification at the corresponding verification level. For each eligible step e, the Boolean verdicts $v_e^{\text{sem}}, v_e^{\text{rule}} \in \{\text{true}, \text{false}\}$ record, respectively, whether the step is semantically valid and whether it is faithful to its declared rule. The two micro-averaged step pass rates are:

$$\text{FVR} = \frac{\sum_{e \in \mathcal{E}_{\text{sem}}} \mathbf{1}[v_e^{\text{sem}}]}{|\mathcal{E}_{\text{sem}}|}, \tag{29}$$

$$\text{RVR} = \frac{\sum_{e \in \mathcal{E}_{\text{rule}}} \mathbf{1}[v_e^{\text{rule}}]}{|\mathcal{E}_{\text{rule}}|}. \tag{30}$$

When an eligible set is empty, its rate is defined as zero.

For an output with an available reference proof, let $N_{i,k}^{\text{generated}}$ be the number of generated intermediate conclusions in its answer-supporting dependency closure and let $N_i^{\text{reference}}$ be the number of reference proof steps. Its granularity deviation and the dataset-level RGD are:

$$\text{RGD}_{i,k} = \left| \log \frac{\max(1, N_{i,k}^{\text{generated}})}{\max(1, N_i^{\text{reference}})} \right|, \tag{31}$$

$$\text{RGD} = \frac{1}{|\mathcal{Q}|} \sum_{(i,k)\in\mathcal{Q}} \text{RGD}_{i,k}, \tag{32}$$

where $\mathcal{Q}$ is the set of evaluated outputs with an available reference proof.

C Additional ProverQA Results

C.1 Easy and Medium Difficulty Splits of Main Results

Table 5: Additional results of Qwen2.5-7B-Instruct and Qwen3-8B on the ProverQA Easy and Medium splits, reported as means followed by standard deviations in gray. ❄ and 🔥 denote training-free and training-based methods, respectively. **Bold** and underline denote the best and second-best means within each backbone model and split.

Model	Method	ProverQA-Easy					ProverQA-Medium				
		Avg@3 ↑	Pass@3 ↑	FVR ↑	RVR ↑	RGD ↓	Avg@3 ↑	Pass@3 ↑	FVR ↑	RVR ↑	RGD ↓
Qwen2.5-7B-Instruct	❄ Backbone	62.20 1.33	87.36 2.06	77.37 0.66	31.19 2.17	0.51 0.016	40.12 0.99	61.09 3.27	64.82 0.71	14.45 1.10	0.90 0.013
	❄ LogicAgent	76.50 0.68	91.57 1.72	68.30 1.83	46.17 1.23	0.46 0.010	56.26 0.84	82.81 2.54	51.32 1.63	25.57 1.37	0.97 0.011
	❄ GoV	69.09 1.51	90.42 1.82	76.67 1.06	37.46 1.26	0.47 0.016	39.82 0.91	64.25 3.20	61.27 0.90	19.39 0.53	0.92 0.019
	🔥 SFT	64.88 2.11	89.27 1.91	76.58 0.71	34.45 2.18	0.51 0.012	40.57 1.23	64.25 3.22	63.33 1.01	14.29 1.22	0.82 0.024
	🔥 GRPO	66.54 1.90	92.72 1.61	83.60 0.50	21.42 1.45	0.54 0.007	46.00 1.57	81.00 2.65	62.79 1.58	8.94 0.90	1.03 0.009
	🔥 PRoSFI	77.08 1.22	96.36 0.00	77.65 0.17	23.03 2.31	0.55 0.019	57.77 0.15	**86.65** 1.80	58.09 0.27	14.86 1.06	0.79 0.010
	🔥 **PROOF-R1**	**79.31** 0.80	**96.55** 1.13	**84.35** 0.55	**68.67** 1.08	**0.40** 0.016	**59.43** 0.66	85.07 2.39	**68.30** 1.07	**34.13** 0.52	**0.73** 0.030
Qwen3-8B	❄ Backbone	97.06 0.13	**100.00** 0.00	82.35 0.32	69.46 0.12	0.30 0.012	96.83 0.26	99.55 0.45	82.16 1.50	59.88 0.92	0.31 0.011
	❄ LogicAgent	**99.36** 0.13	99.62 0.38	97.08 0.43	90.35 0.43	0.27 0.004	**98.94** 0.40	**100.00** 0.00	95.02 0.36	81.38 0.55	0.29 0.012
	❄ GoV	98.08 0.44	99.23 0.54	93.12 0.93	81.43 0.63	0.26 0.002	97.59 0.15	98.64 0.78	89.32 0.61	70.43 1.23	0.26 0.003
	🔥 SFT	96.42 0.51	99.62 0.38	87.12 1.43	74.58 0.86	0.30 0.006	97.74 0.69	99.55 0.45	83.88 0.92	62.37 0.45	0.31 0.004
	🔥 GRPO	99.23 0.22	**100.00** 0.00	98.27 0.05	92.17 0.11	0.28 0.001	97.59 0.15	99.55 0.45	94.74 0.52	69.31 1.13	0.26 0.003
	🔥 PRoSFI	98.72 0.00	99.62 0.38	98.47 0.16	93.49 0.15	0.26 0.005	96.98 0.73	98.64 0.45	93.21 0.31	88.24 0.45	0.26 0.004
	🔥 **PROOF-R1**	98.72 0.34	**100.00** 0.00	**99.19** 0.18	**97.46** 0.30	**0.25** 0.012	97.44 0.15	98.64 0.78	**98.07** 0.04	**96.61** 0.14	**0.25** 0.005

Table 5 complements the results with the Easy and Medium splits of ProverQA. Results are averaged over 3 independent seeds.

On Qwen2.5-7B-Instruct, PROOF-R1 achieves the best Avg@3, FVR, RVR, and RGD on both difficulty splits, and also obtains the best Pass@3 on Easy. The largest separation appears in rule-grounded verification: PROOF-R1 improves RVR over the strongest baseline by 22.50 points on Easy and 8.56 points on Medium. These results show that the verification gains in the main table persist across easier reasoning regimes rather than arising only from the aggregate or Hard split.

On Qwen3-8B, answer accuracy is already near saturation across methods, while PROOF-R1 consistently provides the strongest proof-verification quality. It achieves the best FVR, RVR, and RGD on both Easy and Medium. Relative to the strongest baseline, its RVR improves by 3.97 points on Easy and 8.37 points on Medium. The same pattern across both backbone models shows that PROOF-R1 improves the validity and rule fidelity of generated proofs throughout the difficulty range.

C.2 Cross-Dataset Results with Qwen2.5-7B-Instruct

The FOLIO evaluation uses the broader trusted inference system $\mathcal{R}_{\text{FOLIO}} \supset \mathcal{R}_{\text{train}}$ described in Appendix G. Table 6 reports the performance of ProverQA-trained Qwen2.5-7B-Instruct models on FOLIO and ProofWriter.

On FOLIO, PROOF-R1 achieves the best Avg@3 and RVR, improving over the backbone model by 13.08 and 13.43 points, respectively, while attaining the joint-best Pass@3 of 84.31. On ProofWriter, it improves Avg@3, Pass@3, FVR, and RVR over the backbone model by 18.66, 21.34, 6.54, and 15.72 points, respectively. Together with the Qwen3-8B results in Table 4, these results show that *the cross-dataset gains of ProverQA-trained* PROOF-R1 *persist across two backbone models*.

Table 6: Additional cross-dataset results of ProverQA-trained Qwen2.5-7B-Instruct models on FOLIO and ProofWriter. ❄ and 🔥 denote training-free and training-based methods, respectively. **Bold** and underline denote the best and second-best results within each dataset.

Dataset	Method	Avg@3 ↑	Pass@3 ↑	FVR ↑	RVR ↑
FOLIO	❄ Backbone	49.67	72.55	68.97	15.04
	🔥 SFT	58.17	**84.31**	68.33	18.78
	🔥 **PROOF-R1**	**62.75**	**84.31**	**69.30**	**28.47**
ProofWriter	❄ Backbone	48.67	69.33	66.42	15.55
	🔥 SFT	50.22	68.67	65.15	14.91
	🔥 **PROOF-R1**	**67.33**	**90.67**	**72.96**	**31.27**

C.3 HARD-SPLIT RESULTS ACROSS ADDITIONAL BACKBONE MODELS

Table 7: Additional ProverQA-Hard results across backbone models. ❄ and 🔥 denote training-free and training-based methods, respectively. **Bold** and underline denote the best and second-best results within each backbone model.

Model	Method	Avg@3 ↑	Pass@3 ↑	FVR ↑	RVR ↑	RGD ↓
Llama-3.1 8B	❄ Backbone	35.94	68.33	42.21	5.74	0.85
	🔥 SFT	32.03	63.70	36.18	6.66	**0.83**
	🔥 **PROOF-R1**	**38.79**	**71.89**	**43.29**	**11.06**	0.87
GLM-Z1 9B-0414	❄ Backbone	62.31	84.92	76.00	41.89	1.01
	🔥 SFT	67.34	94.97	75.30	45.21	0.90
	🔥 **PROOF-R1**	**80.23**	**98.49**	**90.79**	**83.24**	**0.65**

Table 7 extends the ProverQA-Hard comparison to Llama-3.1-8B and GLM-Z1-9B-0414. Across both backbone models, PROOF-R1 achieves the best Avg@3, Pass@3, FVR, and RVR. Its RVR exceeds the strongest baseline by 4.40 points on Llama-3.1-8B and 38.03 points on GLM-Z1-9B-0414. On GLM-Z1-9B-0414, it also obtains the lowest RGD. These results extend *the answer and verification gains on the most challenging split beyond the Qwen model family*.

D CONDITIONAL SOUNDNESS OF THE VERIFIED PROOF CERTIFICATE

Guarantee. Let $\Pi_\tau = G_\tau[\mathcal{C}_\tau(c_g)]$ be the answer-supporting proof certificate defined in Equation 17. Assume faithful formal representations, consistent premises, acyclic dependencies that resolve only to original premises or earlier actions, and trusted parsing, solver, and rule-checking components. If every generated conclusion in Π_τ discharges its semantic proof obligation, then:

$$\bigwedge_{c_t \in \mathcal{C}_\tau(c_g) \setminus \mathcal{P}} v_t^{\text{sem}} \quad \Longrightarrow \quad \mathcal{P} \models c_g. \tag{33}$$

A valid `GOAL_BINDING` identifies c_g with the selected proposition $\tilde{y}$, yielding $\mathcal{P} \models \tilde{y}$. If every action in the certificate also discharges its rule obligation, each explicit inference additionally conforms to its declared rule schema.

Proof. Order the generated conclusions in the closure topologically as $c_{t_1}, \ldots, c_{t_m} = c_g$, and let a_{t_i} denote the action that generates c_{t_i}. By assumption, every dependency resolves to an original premise or an earlier action in this order. For the base case, c_{t_1} has no generated-conclusion ancestors, so the resolved dependencies of a_{t_1} are original premises. Its discharged semantic obligation

(Equation 6) establishes that these dependencies entail c_{t_1}, and hence $\mathcal{P} \models c_{t_1}$. For the induction step, suppose that $\mathcal{P}$ entails all earlier generated conclusions in the closure. Every dependency of a_{t_i} is then entailed by $\mathcal{P}$, while the discharged obligation establishes that those dependencies entail c_{t_i}. Transitivity gives $\mathcal{P} \models c_{t_i}$. Applying the same argument through $c_{t_m} = c_g$ proves Equation 33, and valid goal binding gives $\mathcal{P} \models \tilde{y}$.

E Cross-Backend Verification and Independent Semantic Review

Evaluation protocol. The verification gains of PROOF-R1 remain unchanged when Z3 is replaced with cvc5 (Barbosa et al., 2022), veriT,[2] or Isabelle/HOL[3]. We recheck the frozen Qwen2.5-7B-Instruct outputs from the main ProverQA comparison: Backbone, SFT, and PROOF-R1 each contribute three responses to the same 681 problems, totaling 6,129 responses. We use Z3 (*versions 4.16.0*), cvc5 (*versions 1.4.0*), and veriT (*versions 2021.06.2-rmx*), with a 5-second timeout per query. cvc5 uses logic `ALL` and `finite-model-find=true`; veriT uses `UF` for quantified inputs and `QF_UF` otherwise. Isabelle2025-2/HOL uses `blast` (at most 1 second), followed, if needed, by `Nitpick` over domain sizes 1–10, within a shared 5-second budget per query.

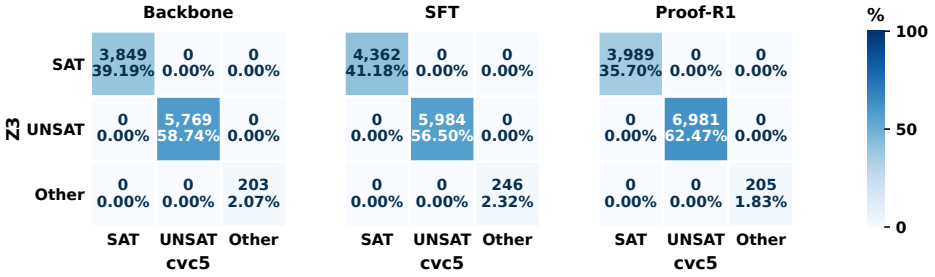


Figure 6: Z3–cvc5 paired local-check outcomes for Qwen2.5-7B-Instruct outputs from Backbone, SFT, and PROOF-R1. Rows denote Z3 and columns denote cvc5, using `SAT`/`UNSAT`/`Other`. Each cell shows its count and percentage of that method's local check attempts.

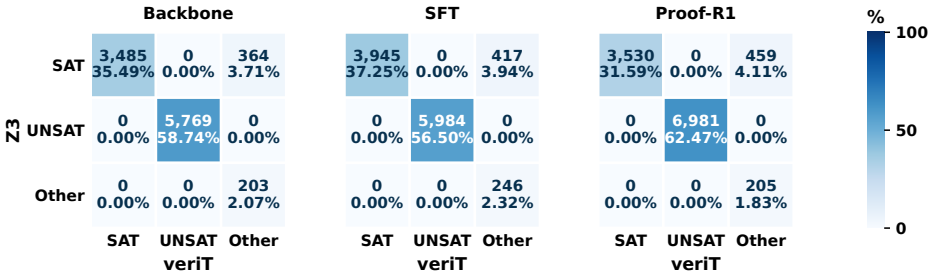


Figure 7: Z3–veriT paired local-check outcomes for the same frozen Qwen2.5-7B-Instruct outputs. Rows denote Z3 and columns denote veriT, using `SAT`/`UNSAT`/`Other`. Each cell shows its count and percentage of that method's local check attempts.

Paired local checks. The three SMT backends check the same semantic verification condition from Equation 6. For this comparison, dependencies resolve against the original premises and the same raw earlier conclusions. We record `SAT`, `UNSAT`, and `Other` separately. Of the 6,129 responses, 5,844 contain a parseable, nonempty structured summary, yielding 31,588 local check attempts. Z3 and cvc5 agree on all 30,934 determinate outcomes: 12,200 `SAT` and 18,734 `UNSAT`

[2] `https://www.verit-solver.org/`
[3] `https://isabelle.in.tum.de/doc/nitpick.pdf`

(Figure 6). veriT confirms all 18,734 `UNSAT` outcomes and 10,960 `SAT` outcomes, with no contradictory decision among the 29,694 jointly determinate pairs (Figure 7). The other 1,240 Z3-`SAT` queries contain quantifiers and return `unknown` in veriT; they are retained as `Other`. All pipelines additionally share 654 input-processing failures: 525 formula-parsing failures, 128 missing dependencies, and one duplicate identifier. No SMT solver timeout occurs in this fixed-query audit.

Native proofs and countermodels. Isabelle2025-2/HOL validates the same local queries through native `blast` proof search and Nitpick countermodel search, without calling an SMT backend. A separate encoder translates the shared FOL syntax tree directly to HOL, preserving quantifier scope, free constants, and nonempty-domain semantics. An oracle-free kernel proof of the requested entailment establishes `UNSAT`; it must have no open hypotheses or remaining subgoals. Nitpick uses Kodkod/SAT4J over domain sizes 1–10, and establishes `SAT` only when its final outcome is `genuine` and a countermodel is saved. These model-search results are distinct from kernel-certified theorems; an unsuccessful finite search remains `Other`. Each distinct query has a 5-second total budget, with at most 1 second for `blast` and the remaining time for Nitpick.

Across 6,880 distinct queries, Isabelle constructs 2,574 kernel proofs and 4,304 genuine countermodels. These confirm all 18,734 Z3-`UNSAT` steps and 12,198 of the 12,200 Z3-`SAT` steps, with no contradictory decision among 30,932 jointly determinate pairs (Figure 8). No Isabelle target-parsing, typing, or backend errors occur.

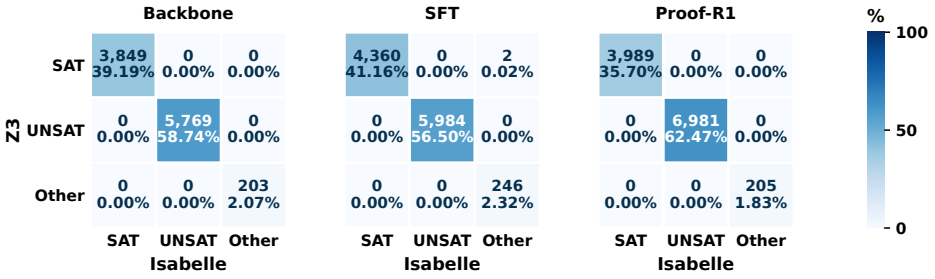


Figure 8: Z3–Isabelle paired local-check outcomes for the same frozen Qwen2.5-7B-Instruct outputs. Rows denote Z3 and columns denote Isabelle, using `SAT`/`UNSAT`/`Other`. Each cell shows its count and percentage of that method's local check attempts. Isabelle requires a genuine Nitpick countermodel for `SAT` and an oracle-free kernel proof for `UNSAT`. `Other` retains input-processing failures and unresolved searches.

Proof verification rates. FVR and RVR are recomputed by replaying each backend's own accepted semantic and rule-level prefixes. All semantic checks use the selected backend, including auxiliary entailment and equivalence queries inside RuleCheck. The metrics retain the micro-averaged definitions in Appendix B. The four backends produce identical FVR and RVR numerators and denominators for every response. Consequently, the rates also match within every difficulty split. Table 8 reports the Overall results. Proof-R1 retains its verification gains over Backbone and SFT under the alternative SMT implementations and native Isabelle proofs.

Table 8: Cross-backend verification of the same frozen Qwen2.5-7B-Instruct outputs on ProverQA Overall. Values are percentages; every semantic check uses the indicated backend.

	FVR ↑				RVR ↑			
Method	Z3	cvc5	veriT	Isabelle	Z3	cvc5	veriT	Isabelle
Backbone	67.64	67.64	67.64	67.64	18.91	18.91	18.91	18.91
SFT	66.39	66.39	66.39	66.39	19.94	19.94	19.94	19.94
Proof-R1	70.17	70.17	70.17	70.17	41.19	41.19	41.19	41.19

Independent semantic review. We obtain blinded local-entailment judgments from GPT-5.5 on 2,000 steps: 667 Backbone, 667 SFT, and 666 PROOF-R1 steps, spanning 625 problems and 1,680 generated responses.

GPT-5.5 receives only declared dependencies, their formulas, and the conclusion. Generating-model identity and solver results are hidden. It judges classical FOL entailment, returning *valid*, *invalid*, or *uncertain* with a derivation, counterexample, or reason for uncertainty.

Figure 9 shows the corresponding Z3–GPT-5.5 outcomes for each generating method. The heatmap reports raw sample counts and within-method sample percentages. Table 9 reports weighted agreement for the 2,000-step sample. GPT-5.5 endorses all 1,185 sampled `UNSAT` steps. Of 772 `SAT` steps, it labels 767 *invalid*, three *uncertain*, and two *valid*.

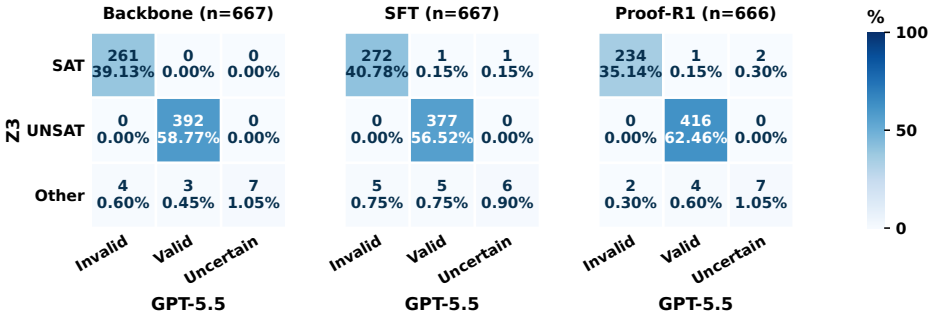


Figure 9: Z3–GPT-5.5 local semantic review of 2,000 sampled steps from frozen Qwen2.5-7B-Instruct outputs. Rows denote Z3 outcomes; columns denote GPT-5.5 judgments. `SAT` corresponds to *invalid* and `UNSAT` to *valid*. Each cell shows its count and percentage of reviewed steps.

Table 9: Agreement is measured on determinate Z3/cvc5 outcomes. The weighted percentage uses the sampling weights of the 2,000-step cohort.

Reviewer	Number	Valid	Invalid	Uncertain	Agreed / reference	Weighted (%)
GPT-5.5	2,000	1,199	778	23	1,952 / 1,957	99.73

What do these independent checks establish? Together, the cross-backend verification and independent semantic review establish that the verification gains of PROOF-R1 are not specific to Z3. Replacing Z3 with cvc5, veriT, or Isabelle preserves every response-level FVR and RVR, while Isabelle independently confirms 30,932 local judgments using native kernel proofs or genuine finite countermodels. The blinded GPT-5.5 review further achieves 99.73% weighted agreement with the determinate Z3/cvc5 outcomes. Together, these results demonstrate that *the verification gains of* PROOF-R1 *are robust across heterogeneous formal backends*.

F RULE ONTOLOGY

RuleCheck. Each canonical rule $\rho \in \mathcal{R}$ is associated with a finite family $\mathfrak{S}_\rho$ of admissible schemas. A schema $(\Gamma, \gamma) \in \mathfrak{S}_\rho$ consists of premise patterns Γ and a conclusion pattern γ. Applying the resolver $\Phi_t^{\text{rule}}(\cdot)$ from Equation 8 to D_t yields the dependency formulas passed to RULECHECK, which we abbreviate as ψ_t^{rule}:

$$\psi_t^{\text{rule}} := \Phi_t^{\text{rule}}(D_t) = \left\{ \left(\mathcal{P} \cup \mathcal{S}_{t-1}^{\text{rule}}\right)(d) \;\middle|\; d \in D_t \right\}.$$

Using these resolved formulas, RULECHECK is defined by

$$\text{RuleCheck}\left(\rho_t, \psi_t^{\text{rule}}, c_t\right) = \mathbb{I}\left[\rho_t \in \mathcal{R} \wedge \exists (\Gamma, \gamma) \in \mathfrak{S}_{\rho_t},\ \exists \sigma : \psi_t^{\text{rule}} \models \sigma(\Gamma) \wedge c_t \equiv \sigma(\gamma)\right]. \quad (34)$$

Here, σ is a substitution over schema variables, $\psi_t^{\text{rule}} \models \sigma(\Gamma)$ means that every instantiated premise pattern is established by the resolved dependency formulas, and $\equiv$ denotes logical equivalence.

Consequently, an unrecognized rule, an untrusted dependency, or the absence of a matching schema yields a negative rule verdict.

Table 10: Predicate-logic rules covered by the ProverQA training data.

Rule	Symbolic form
IE: IMPLICATION_ELIMINATION	$A \to B,\ A \vdash B$
MT: MODUS_TOLLENS	$A \to B,\ \neg B \vdash \neg A$
XOI: EXCLUSIVE_DISJUNCTION_INTRODUCTION	$A,\ \neg B \vdash A \oplus B;\ \neg A,\ B \vdash A \oplus B$
XOE: EXCLUSIVE_DISJUNCTION_ELIMINATION	$A \oplus B,\ A \vdash \neg B;\ A \oplus B,\ \neg A \vdash B$
DS: DISJUNCTIVE_SYLLOGISM	$A \vee B,\ \neg A \vdash B$
CI: CONJUNCTION_INTRODUCTION	$A,\ B \vdash A \wedge B$
CE: CONJUNCTION_ELIMINATION	$A \wedge B \vdash A;\ A \wedge B \vdash B$
UE: UNIVERSAL_ELIMINATION	$\forall x\, \varphi(x) \vdash \varphi(a)$
GB: GOAL_BINDING	$i_g = i(\widetilde{y}),\quad c_g = \widetilde{y}$

In the last row, $i(\widetilde{y})$ denotes the terminal action identifier assigned to the selected answer proposition. GOAL_BINDING is accepted only when both that identifier and the conclusion match.

G Evaluation of Proof-R1 on Out-of-Distribution Datasets

ProverQA is generated by a theorem prover, and its proof traces are constructed from the predicate-logic ontology used for training. FOLIO (Han et al., 2024), in contrast, is a human-authored natural-language reasoning benchmark whose instances and annotations are independent of that generator. We therefore evaluate the frozen Qwen3-8B Backbone, SFT, and Proof-R1 outputs on the FOLIO validation subset: 51 problems with three outputs per problem, or 153 outputs per model. In FOLIO, semantically valid derivations include universal and existential quantifier steps whose rule families are absent from the ProverQA training ontology. Let $\mathcal{R}_{\text{unseen}}$ denote the seven unseen quantifier rule families listed in Table 11. The trusted inference system used for this FOLIO evaluation is therefore:

$$\mathcal{R}_{\text{FOLIO}} := \mathcal{R}_{\text{train}} \cup \mathcal{R}_{\text{unseen}} \supset \mathcal{R}_{\text{train}}. \tag{35}$$

To verify these deductions without changing the frozen outputs or the original checker, a separate coverage checker operationalizes $\mathcal{R}_{\text{FOLIO}}$ by adding the seven guarded rule families in Table 11 to $\mathcal{R}_{\text{train}}$. Every formula, dependency, step identifier, and GOAL_BINDING action remains fixed.

Table 11: Training-time-unseen rule families and their representative symbolic forms.

Rule	Symbolic form
UIC: UNIVERSAL_IMPLICATION_CHAINING	$\forall x(A \to B),\ \forall x(B \to C) \vdash \forall x(A \to C)$
UPI: UNIVERSAL_PROPOSITIONAL_INFERENCE	$\forall x(A \to B),\ \forall x(A \to C) \vdash \forall x(A \to (B \wedge C))$
EI: EXISTENTIAL_INTRODUCTION	$A(a) \vdash \exists x\, A(x);\ A(a), B(a) \vdash \exists x(A(x) \wedge B(x))$
ECE: EXISTENTIAL_CONJUNCTION_ELIMINATION	$\exists x(A \wedge B) \vdash \exists x\, A;\ \exists x(A \wedge G) \vdash G\ (x \notin \mathrm{FV}(G))$
EIE: EXISTENTIAL_IMPLICATION_ELIMINATION	$\exists x A,\ \forall x(A \to B) \vdash \exists x B$
ECI: EXISTENTIAL_CONJUNCTION_INTRODUCTION	$\exists x A, G \vdash \exists x(A \wedge G)\ (x \notin \mathrm{FV}(G))$
QN: QUANTIFIER_NEGATION	$\neg\forall x A \Leftrightarrow \exists x \neg A;\ \neg\exists x A \Leftrightarrow \forall x \neg A$

Using the frozen outputs, we evaluate every step under $\mathcal{R}_{\text{FOLIO}}$. Let ρ_t denote the rule family of the current step. To isolate generalization to unseen rules, we group steps by whether $\rho_t \in \mathcal{R}_{\text{train}}$ or $\rho_t \in \mathcal{R}_{\text{unseen}}$. The group assignment depends only on the current step.

Table 12: FVR/RVR (%) under $\mathcal{R}_{\mathrm{FOLIO}}$ on frozen Qwen3-8B FOLIO outputs, grouped by whether the current rule belongs to $\mathcal{R}_{\mathrm{train}}$ or $\mathcal{R}_{\mathrm{unseen}}$.

Model	Rule group	FVR	RVR
Backbone	$\mathcal{R}_{\mathrm{train}}$	64.71	39.73
	$\mathcal{R}_{\mathrm{unseen}}$	63.24	34.92
SFT	$\mathcal{R}_{\mathrm{train}}$	62.02	39.77
	$\mathcal{R}_{\mathrm{unseen}}$	67.50	45.71
PROOF-R1	$\mathcal{R}_{\mathrm{train}}$	**81.55**	**58.87**
	$\mathcal{R}_{\mathrm{unseen}}$	**86.21**	**63.08**

Under the expanded checker, PROOF-R1 achieves the highest FVR and RVR in both groups: 81.55/58.87 for $\mathcal{R}_{\mathrm{train}}$ and 86.21/63.08 for $\mathcal{R}_{\mathrm{unseen}}$. On $\mathcal{R}_{\mathrm{unseen}}$, this improves over the Backbone by 22.97 FVR points and 28.16 RVR points. Overall, PROOF-R1 reaches 59.70% RVR under the expanded ontology. These results show that PROOF-R1 *transfers from the prover-generated training distribution to human-authored FOLIO and retains verifiable reasoning under rule families absent from the training ontology.*

H FORMAL SPECIFICATION AND PROOFCHECK

Action specification. Each action proposes a proof-state transition whose admissibility is governed by a formal specification:

$$\begin{aligned}\mathrm{Spec}(a_t;\mathcal{S}_{t-1}) = \ & \mathrm{WellFormed}(a_t) \wedge \mathrm{TrustedDep}(D_t;\mathcal{S}_{t-1}) \\ & \wedge \mathrm{SemanticValid}(D_t, c_t) \wedge \mathrm{RuleFaithful}(D_t, c_t, \rho_t) \\ & \wedge \mathrm{Admissible}(a_t).\end{aligned} \tag{36}$$

$\mathrm{WellFormed}$ checks the action schema, identifier, conclusion, and rule field, while $\mathrm{TrustedDep}$ restricts dependencies to original premises or earlier conclusions admitted to the relevant verified state. $\mathrm{SemanticValid}$ requires entailment between the resolved dependency formulas and the conclusion. $\mathrm{RuleFaithful}$ requires the deduction to instantiate the declared rule. For ordinary steps, $\mathrm{Admissible}$ is evaluated by $\mathrm{ProofCheck}$. for the terminal action, it requires agreement between the identifier, conclusion, and selected candidate.

ProofCheck. For ordinary actions, PROOFCHECK rejects repeated conclusions, restatements of dependencies, and tautologies:

$$\begin{aligned}v_t^{\mathrm{pro}} = \mathrm{ProofCheck}(a_t) = \ & \neg\,\mathrm{Repeated}(c_t) \wedge \neg\,\mathrm{Restatement}(c_t, D_t) \\ & \wedge \neg\,\mathrm{Tautological}(c_t).\end{aligned} \tag{37}$$

Proof progress is separate from semantic soundness. A repeated conclusion or tautology may be entailed without advancing the proof, so nontriviality is not an entailment requirement.

I STRUCTURED OUTPUT CONTRACT

Output organization. Following the rollout notation in Equation 3, the free-form analysis region represents z, while the structured summary serializes $(a_1, \ldots, a_T)$ as a JSON list. Its terminal action a_g binds $c_g = \tilde{y}$ for the selected candidate $\tilde{y} \in \mathcal{Y}$.

Action schema. As shown in Figure 10, each action contains a unique identifier, a list of dependencies, one formal conclusion, and its declared rule. For a candidate set $\mathcal{Y} = \{y_1, \ldots, y_N\}$, the final action uses the identifier and formal proposition of the selected option.

```
[
  {"id": "s1", "dependencies": ["p1", "p2"],
   "conclusion": "Q(a)",
   "rule": "IMPLICATION_ELIMINATION"},
  {"id": "option_3", "dependencies": ["s1"],
   "conclusion": "Q(a)",
   "rule": "GOAL_BINDING"}
]
```

Figure 10: Example of the structured JSON proof-action sequence.

J Implementation Details

We implement semantic verification with Z3 (de Moura & Bjørner, 2008), and use RuleCheck and ProofCheck for the corresponding rule and proof-admissibility conditions. Appendix E reports paired cvc5, veriT, and Isabelle checks of the frozen Qwen2.5-7B-Instruct evaluation outputs. Policy optimization uses LoRA (Hu et al., 2022) within VERL/HybridFlow (Sheng et al., 2024).

J.1 Method-Specific Implementations

To ensure a controlled comparison, all methods use the same data splits, formal inputs, and structured output specification, and their final outputs are scored by a common evaluator for answer accuracy and proof verifiability. SFT, GRPO, PRoSFI, and Proof-R1 use the same training configuration and LoRA setup and are initialized from the same format-warm-up adapter. The methods differ mainly in their inference procedures, training objectives, and credit-assignment mechanisms.

Backbone is the pretrained model without task-specific training. Given the common prompt, it directly generates a complete response containing a reasoning trace, structured proof steps, and a final answer, without candidate selection or verification feedback at inference time. The formal verifier is used only for evaluation after generation and does not participate in the generation process.

LogicAgent (Zhang et al., 2025) is a training-free neuro-symbolic reasoning agent. It first uses an LLM to reason natural-language problems from multiple semantic perspectives, delegates deduction to symbolic solvers, and determines the final answer through reflection. We use the official code repository.[4] To conform to the common evaluation protocol, we only serialize its selected reasoning process into the standard structured proof format, without using reference answers or our verifier to alter its result.

GoV (Fang et al., 2026) is a training-free, verification-guided reasoning agent. It first generates multiple candidate solutions and organizes their reasoning steps as nodes in dependency graphs and then checks the nodes in their declared dependency order and selects the final output from the candidates that pass verification. Candidate selection uses GoV's own LLM-based verifier. We use the official code repository.[5] To standardize the output, we only serialize the selected candidate into the common structured proof format, without changing its selection result or reasoning content.

SFT uses verified structured proof trajectories for supervised fine-tuning. Given a problem and its formal representation, it learns through token-level cross-entropy to generate a reasoning trace, canonicalized proof steps, and a final answer. SFT performs neither online sampling nor additional feedback based on the semantic or rule validity of generated proofs. Its supervision therefore comes entirely from the provided demonstration trajectories.

GRPO (Shao et al., 2024) is initialized from the format-warm-up adapter and samples a group of complete responses for each problem. Its reward is determined jointly by output-format validity and final-answer correctness, and relative rewards within each group are used to compute the advantage of each response. This advantage is applied to the entire generated sequence and the training reward

[4] https://github.com/AI4SS/Logic-Agent

[5] https://github.com/Frevor/Graph-of-Verification

does not separately check the semantic validity of intermediate conclusions, the applicability of declared rules, or proof dependencies. We implement GRPO using the official verl codebase.[6]

PRoSFI (Chen et al., 2026) is an RL method designed for natural-language logical reasoning. It retains GRPO's grouped sampling and relative optimization, is initialized from the format-warm-up adapter, and samples a group of complete responses for each problem. For each response, it parses the structured proof, checks intermediate conclusions with Z3 in dependency order, and assigns the trajectory-level reward in Equation 38. Because the authors have not released their code, we construct a faithful reimplementation following the method described in the paper.

$$R_{\mathrm{PRoSFI}} = \begin{cases} 1.0, & \text{correct answer; all steps verified,} \\ 0.3, & \text{correct answer; some steps unverified,} \\ 0.1, & \text{valid output format but incorrect answer,} \\ 0.0, & \text{invalid output format or other failure.} \end{cases} \tag{38}$$

J.2 Training Configuration

The training configuration used in our experiments is summarized below.[7]

Table 13: Training and rollout parameters. Unless stated otherwise, these settings are shared by SFT, GRPO, PRoSFI, and PROOF-R1.

Category	Parameter	Value
LoRA	Rank	8
	Alpha	16
	Target modules	[Q, K, V, O, GATE, UP, DOWN]
Rollout	Group size (K)	8
	Sampling temperature	0.8
	Top-p	0.95
Training	Training batch size	8
	Sampled responses per batch	64
	Total training steps	1,098
	Training epochs	3
Optimization	PPO epochs per update	2
	PPO mini-batch size	8
	Clip ratio (ϵ_-, ϵ_+)	$[0.20, 0.28]$
	Learning rate	5×10^{-6}
	KL coefficient (λ_{KL})	0.001
	Loss aggregation	seq-mean-token-sum
PROOF-R1 Parameters	*Machine-Checkable Formal Verification (MCFV)*	
	Initial value ($\mathrm{EMA}_\rho^{(0)}$)	0.5
	EMA coefficient (β)	0.9
	Clip range ($b_{\min}, b_{\max}$)	$[0.30, 0.70]$
	Verification-Aligned Optimization	
	Verification scale (λ_v)	3
	Outcome scale (λ_o)	1
	Goal-binding verification scale	1
	No-positive-action weight	0.1
Reasoning	Maximum prompt length	4,096
	Maximum response length	4,096
	Maximum model context length	8,192

[6] https://github.com/verl-project/verl

[7] Code is available at https://anonymous.4open.science/r/Proof-R1/.

K Additional Training Diagnostics

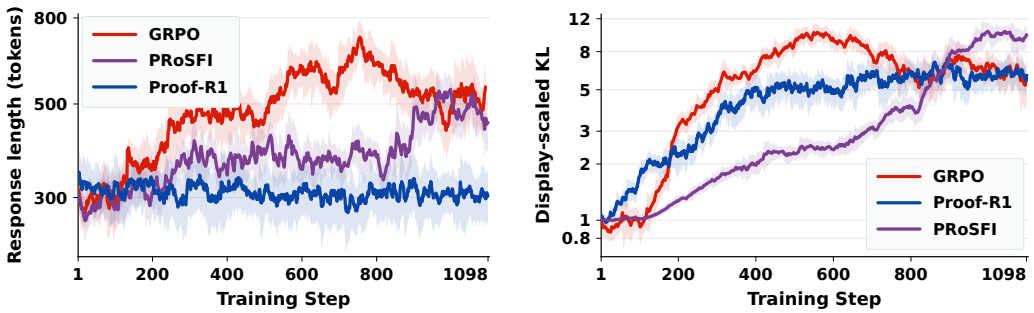


Figure 11: Training trajectories of response length and policy divergence during reinforcement learning. Left: Mean response length of different methods on Qwen2.5-7B-Instruct during training. Right: Sequence-level KL divergence of the Qwen3-8B policy from the frozen reference policy during training. Solid lines denote EMA-smoothed trajectories, and shaded regions indicate one exponentially weighted local standard deviation.

Figure 11 shows that, compared with the response growth of GRPO and PRoSFI as training proceeds, PROOF-R1 maintains an overall stable response length. This indicates that *the training improvements of* PROOF-R1 *do not rely on continually increasing response length, while the policy gradually diverges from the initial reference policy under a stable generation length.*

L Computational Cost

All experiments use a PyTorch 2.5.1 base image with Python 3.12 on Ubuntu 22.04 and CUDA 12.4. The machine provides four RTX 4090 GPUs with 48 GB of memory each, 80 Intel(R) Xeon(R) Platinum 8470Q vCPUs, and 384 GB of system memory. Table 14 reports wall-clock training time per 100 optimization steps and the mean number of generated tokens consumed per response at inference time. Training-free agents have zero training time.

Table 14: Computational cost on ProverQA. Time is measured in hours per 100 optimization steps. Inference-time token consumption is the mean number of generated tokens per response.

	Qwen2.5-7B-Instruct		**Qwen3-8B**	
Method	h/100 steps	tokens/response	h/100 steps	tokens/response
Backbone	0.00	354.2	0.00	1,213.8
LogicAgent	0.00	2,054.5	0.00	12,572.6
GoV	0.00	2,004.7	0.00	9,127.9
SFT	0.11	373.6	0.19	1,229.9
GRPO	2.64	836.4	4.70	1,176.3
PRoSFI	2.98	645.9	5.31	1,208.1
PROOF-R1	2.73	354.9	4.86	1,214.2

Across two backbones, PROOF-R1, GRPO, and PRoSFI have comparable training speeds. PROOF-R1 *consumes substantially fewer inference-time tokens than the LogicAgent and GoV baselines.*

M Prompt Template Used by PROOF-R1

The model receives a single-turn chat prompt consisting of a shared system instruction and an instance-specific user message. The structured-action schema and user-message template are shared

across format warm-up, policy rollouts, and evaluation, while the inference-system block is instantiated for the current dataset. Qwen2.5-7B-Instruct and Llama-3.1-8B use the explicit-thinking instruction below, whereas Qwen3-8B and GLM-Z1-9B-0414 use their native thinking mode.

The marker `[OUTPUT-MODE INSTRUCTION]` is replaced verbatim by one of the two mode-specific blocks below. The marker `[INFERENCE SYSTEM BLOCK]` is instantiated with $\mathcal{R}_{\text{train}}$ for ProverQA and ProofWriter, and with $\mathcal{R}_{\text{FOLIO}} = \mathcal{R}_{\text{train}} \cup \mathcal{R}_{\text{unseen}}$ for FOLIO. The latter is formed by appending the FOLIO extension block below to the training inference-system block.

System Prompt

You are a math reasoner. For each problem, you will receive a set of formal premises and a multiple–choice logical reasoning question.

[OUTPUT–MODE INSTRUCTION]

Each JSON object in <summary> must have exactly these fields:
– id: a unique identifier for this step.
– dependencies: a list of premise ids or earlier step ids.
– conclusion: exactly one formal formula.
– rule: the single inference rule applied, chosen from the supplied inference system.

[INFERENCE SYSTEM BLOCK]

The rule must describe the operation that produces the declared conclusion. If an implication first activates a compound consequent and the same action then selects one XOR/OR/AND branch, use the corresponding elimination rule, not IMPLICATION_ELIMINATION. Do not invent rule names, aliases, abbreviations, or alternative capitalization.

Use GOAL_BINDING for the final JSON object whose id is one of [option A id], [option B id], ..., or [option N id].

Critical requirements:
– Use ”s1”, ”s2”, ”s3”, etc. for intermediate steps.
– The last JSON object must be the final answer.
– The final answer id must exactly match one of the option ids shown in the problem.
– The final answer conclusion must match the formal statement of that option.
– The JSON array must be valid and parsable.
– Each step must use fewer than 5 dependencies.
– Do not output any additional text after </summary>.

Explicit-Thinking Instruction (Qwen2.5-7B-Instruct and Llama-3.1-8B)

Answer with exactly two parts:
1. Put brief natural–language reasoning inside <think>...</think>.
2. Put a JSON array of structured formal intermediate steps inside <summary>...</summary>.

Native-Thinking Instruction (Qwen3-8B and GLM-Z1-9B-0414)

Use the model’s native thinking mode for brief natural–language reasoning.
After native thinking ends, output exactly one JSON array of structured formal intermediate steps inside <summary>...</summary>.
Do not add commentary between the native </think> and <summary> sections.

The training prompt instantiates the inference system $\mathcal{R}_{\text{train}}$ with the following canonical rules.

Training Inference System Block ($\mathcal{R}_{\text{train}}$)

Canonical rule options:
Use exactly one of these canonical rule names:
– IMPLICATION_ELIMINATION: from A −> B and support for A, derive the entire consequent B.
– MODUS_TOLLENS: from A −> B and support for not B, derive not A (including a justified component of a compound A).
– EXCLUSIVE_DISJUNCTION_INTRODUCTION: from A and not B, or from not A and B, derive A XOR B.
– EXCLUSIVE_DISJUNCTION_ELIMINATION: from A XOR B plus one known branch or its negation, derive the forced other branch or its negation.
– DISJUNCTIVE_SYLLOGISM: from A OR B and not A derive B, or from A OR B and not B derive A.
– CONJUNCTION_INTRODUCTION: from A and B derive A AND B.
– CONJUNCTION_ELIMINATION: from A AND B derive A or derive B.
– UNIVERSAL_ELIMINATION: instantiate a universally quantified formula for one concrete entity.
– GOAL_BINDING: only for a final_answer action whose id and conclusion exactly match an answer option.

The FOLIO prompt extends $\mathcal{R}_{\text{train}}$ to $\mathcal{R}_{\text{FOLIO}}$ with $\mathcal{R}_{\text{unseen}}$.

Unseen Inference System Block ($\mathcal{R}_{\text{unseen}}$)

– UNIVERSAL_IMPLICATION_CHAINING: from forall x (A −> B) and forall x (B −> C), derive forall x (A −> C).
– UNIVERSAL_PROPOSITIONAL_INFERENCE: from forall x (A −> B) and forall x (A −> C), derive forall x (A −> (B AND C)).
– EXISTENTIAL_INTRODUCTION: from A(a), derive exists x A(x); premises sharing witness a may be combined under the existential.
– EXISTENTIAL_CONJUNCTION_ELIMINATION: from exists x (A AND B), derive exists x A; or derive a conjunct G that does not contain x free.
– EXISTENTIAL_IMPLICATION_ELIMINATION: from exists x A and forall x (A −> B), derive exists x B.
– EXISTENTIAL_CONJUNCTION_INTRODUCTION: from exists x A and G, derive exists x (A AND G) when x is not free in G.
– QUANTIFIER_NEGATION: apply the equivalences not forall x A iff exists x not A, and not exists x A iff forall x not A.

For each problem, the bracketed fields are populated from the corresponding natural-language and formalized instance. Premise formulas are converted to the prefix notation consumed by the verifier.

User Prompt Template

Context:
1. [natural−language premise 1]. Formal statement: 'h1 : [formal premise 1]'.
2. [natural−language premise 2]. Formal statement: 'h2 : [formal premise 2]'.
...
N. [natural−language premise N]. Formal statement: 'hN : [formal premise N]'.

Question: [multiple−choice question]

Options:
A) [option A text]. Answer id: '[option A id]'. Formal statement: '[option A formula]'.
B) [option B text]. Answer id: '[option B id]'. Formal statement: '[option B formula]'.
...
N) [option N text]. Answer id: '[option N id]'. Formal statement: '[option N formula]'.

The correct option is:

N CASE STUDY

N.1 CASE 1: A NEGATIVE CONCLUSION ABOUT ANIMAL CARE

PROOF-R1 connects a natural-language answer to an explicit sequence of checkable reasoning actions. The case below presents a ProverQA example generated by PROOF-R1 with Qwen3-8B.

Premises $\mathcal{P}$

`p1` Either Alonzo helps animals or harms animals, but not both.

`p2` Alonzo has compassion.

`p3` Alonzo loves wildlife.

`p4` Anyone who loves wildlife and provides care is helping animals.

`p5` Anyone who feels empathy or has compassion can provide care.

Query q

Alonzo does not harm animals.

Candidates $\mathcal{Y}$ $\quad \tilde{y} = y_1$

y_1 Alonzo does not harm animals.

y_2 Alonzo harms animals.

y_3 It is uncertain whether Alonzo harms animals.

PROOF-R1 $\quad \tilde{y} = y_1$

`s1` Alonzo has compassion, so `p5` implies that he provides care.

$D_t =$ `[p2, p5]` $\quad c_t =$ `provides_care(Alonzo)` $\quad \rho_t = \text{IE}$

`s2` Combine his love of wildlife with the derived fact that he provides care.

$D_t =$ `[s1, p3]` $\quad c_t =$ `loves_wildlife(Alonzo)` $\wedge$ `provides_care(Alonzo)` $\quad \rho_t = \text{CI}$

`s3` By `p4`, loving wildlife and providing care imply that he helps animals.

$D_t =$ `[s2, p4]` $\quad c_t =$ `helps_animals(Alonzo)` $\quad \rho_t = \text{IE}$

`s4` He helps animals; `p1` excludes helping and harming together, so he does not harm them.

$D_t =$ `[s3, p1]` $\quad c_t =$ `¬harms_animals(Alonzo)` $\quad \rho_t = \text{XOE}$

`h_goal_true` Bind this conclusion to the selected candidate y_1.

$D_t =$ `[s4]` $\quad c_t =$ `¬harms_animals(Alonzo)` $\quad \rho_t = \text{GB}$

The blue highlights show how PROOF-R1 turns an intermediate result into a trusted dependency. The conclusion `provides_care(Alonzo)` at `s1` is admitted to the semantic and rule-verified states after its action passes verification (Section 3.3). The explicit `s1` reference in `s2` then reuses this fact to construct the conjunction required by `p4`, enabling `s3` to derive that Alonzo helps animals. *This link makes the source of the intermediate result and its role in subsequent reasoning directly checkable*. The yellow highlights show how a constraint in the input supports the negative answer. The phrase "but not both" in `p1` excludes simultaneous helping and harming. Given the helping conclusion at `s3`, exclusive disjunction elimination at `s4` yields `¬harms_animals(Alonzo)`, which `h_goal_true` binds to candidate y_1. The negative conclusion therefore has explicit supporting dependencies and a matching inference rule. All five actions pass schema validation, UNSAT-based semantic verification, and RULECHECK.

N.2 CASE 2: VERIFIED DEPENDENCIES FOR A CORRECT ANSWER

MCFV makes the validity of an intermediate conclusion decisive for its reuse in a verified proof. This ProverQA pair uses Qwen2.5-7B-Instruct on the same problem: the Backbone answers incorrectly, while PROOF-R1 derives the correct answer with verified actions.

Premises $\mathcal{P}$

- `p1` Jayceon does not have raw talent.
- `p2` If Jayceon is dedicated, then he can either achieve success or stay humble, but not both.
- `p3` Jayceon either has raw talent or is dedicated, but not both.
- `p4` Anyone who sets goals or works hard is dedicated.
- `p5` Jayceon achieves success.
- `p6` Superstar achieves success.

Query q

Jayceon stays humble.

Candidates $\mathcal{Y}$ $y^* = y_2$

- y_1 Jayceon stays humble.
- y_2 Jayceon does not stay humble.
- y_3 It is uncertain whether Jayceon stays humble.

Backbone $\tilde{y} = y_1$

s1 Claims that Jayceon is not dedicated from the exclusive choice and his lack of talent.
$D_t =$ `[p1, p3]` $c_t =$ `¬dedicated(Jayceon)` $\rho_t =$ XOI

s2 Uses this rejected conclusion with the success premise to claim that he stays humble.
$D_t =$ `[p2,p5,s1]` $c_t =$ `stay_humble(Jayceon)` $\rho_t =$ XOE

PROOF-R1 $\tilde{y} = y_2$

s1 He has no raw talent, so the exclusive choice in `p3` establishes that he is dedicated.
$D_t =$ `[p1, p3]` $c_t =$ `dedicated(Jayceon)` $\rho_t =$ XOE

s2 Given his dedication and success, `p2` excludes staying humble.
$D_t =$ `[p2,s1,p5]` $c_t =$ `¬stay_humble(Jayceon)` $\rho_t =$ XOE

h_goal_false Bind the negative conclusion to the correct candidate y_2.
$D_t =$ `[s2]` $c_t =$ `¬(stay_humble(Jayceon))` $\rho_t =$ GB

The red highlights trace an unsupported conclusion into the Backbone's wrong answer. Its first action passes schema validation, but the semantic check returns SAT and RULECHECK rejects the declared XOI rule: `p1` and `p3` entail `dedicated(Jayceon)`, not its negation. The next action still cites `s1`, so verification reports a missing trusted dependency at the action that states the wrong answer. The blue highlights show the corresponding verified transition in PROOF-R1: exclusive disjunction elimination establishes dedication, which is admitted to both verified states and reused by `s2`. This fact activates `p2`; together with success, it yields the yellow negative conclusion and the correct answer y_2. All three actions pass schema, *UNSAT-based semantic, and rule verification, yielding a continuous chain of trusted dependencies*.

N.3 Case 3: Generating the Dependencies That Support the Answer

This ProverQA pair uses Qwen2.5-7B-Instruct on the same problem and sample. The Backbone mixes two characters' facts; PROOF-R1 builds the dependencies establishing Clay's correct answer.

Premises $\mathcal{P}$

- `p1` Clay either learns magic theory or studies ancient tomes.
- `p2` Clay practices intricate spells.
- `p3` If Clay studies ancient tomes and practices intricate spells, then he can become a powerful mage.
- `p4` If Mitchell studies ancient tomes and practices intricate spells, then he can become a powerful mage.
- `p5` Clay does not learn magic theory.
- `p6` Mitchell practices intricate spells.

Query q

Clay can become a powerful mage.

Candidates $\mathcal{Y}$ $\quad y^* = y_1$

- y_1 Clay can become a powerful mage.
- y_2 Clay cannot become a powerful mage.
- y_3 It is uncertain whether Clay can become a powerful mage.

Backbone $\quad \tilde{y} = y_2$

s1 Introduces a choice between Clay and Mitchell studying ancient tomes.

$D_t =$ `[p1, p5]` $\quad c_t =$ `(study_ancient_tomes(Clay)` $\vee$ `study_ancient_tomes(Mitchell))` $\quad \rho_t =$ DS

s2 Combines the facts that Clay and Mitchell both practice intricate spells.

$D_t =$ `[p2, p6]` $\quad c_t =$ `(practice_intricate_spells(Clay)` $\wedge$ `practice_intricate_spells(Mitchell))` $\quad \rho_t =$ CI

s3 Combines the two characters' conditions into a disjunction.

$D_t =$ `[s1, s2]` $\quad c_t =$ `(study_ancient_tomes(Clay)` $\wedge$ `practice_intricate_spells(Clay))` $\vee$ `(study_ancient_tomes(Mitchell)` $\wedge$ `practice_intricate_spells(Mitchell))` $\quad \rho_t =$ XOI

s4 Cites both rules but introduces an unsupported disjunction with a negated Clay predicate.

$D_t =$ `[s3,p3,p4]` $\quad c_t =$ `¬(become_a_powerful_mage(Clay))` $\vee$ `(become_a_powerful_mage(Mitchell))` $\quad \rho_t =$ IE

s5 Uses this disjunction and Mitchell's practice to claim that Clay cannot become a powerful mage.

$D_t =$ `[s4, p6]` $\quad c_t =$ `¬(become_a_powerful_mage(Clay))` $\quad \rho_t =$ MT

PROOF-R1 $\quad \tilde{y} = y_1$

s1 Clay does not learn magic theory, so `p1` implies that he studies ancient tomes.

$D_t =$ `[p1, p5]` $\quad c_t =$ `study_ancient_tomes(Clay)` $\quad \rho_t =$ DS

s2 Combine this fact with Clay's practice to establish both conditions required by `p3`.

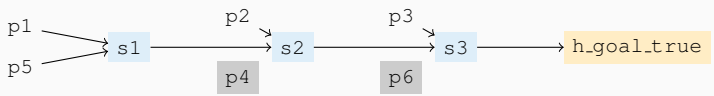
$D_t =$ `[s1, p2]` $\quad c_t =$ `study_ancient_tomes(Clay)` $\wedge$ `practice_intricate_spells(Clay)` $\quad \rho_t =$ CI

s3 Apply Clay's rule to these conditions, proving the correct candidate y_1.

$D_t =$ `[s2, p3]` $\quad c_t =$ `become_a_powerful_mage(Clay)` $\quad \rho_t =$ IE

h_goal_true Bind this conclusion to the selected candidate y_1.

$D_t =$ `[s3]` $\quad c_t =$ `become_a_powerful_mage(Clay)` $\quad \rho_t =$ GB



Red marks the Backbone's rule mismatch at `s1`, the resulting untrusted dependency at `s3`, and the unsupported negation introduced at `s4` and propagated to `s5`. Its first two conclusions are semantically valid, but the mixed path fails to prove the query. The blue dependencies in PROOF-R1 connect Clay's facts to `p3`, yielding the yellow correct answer; all four actions pass schema, semantic, and rule verification. Backward traversal in the candidate proof-certificate graph selects all four actions and premises `p1`, `p2`, `p3`, and `p5`; the gray-shaded `p4` and `p6` lie outside the closure. *This is the answer-supporting structure targeted by ASDC training.*

N.4 CASE 4: LINKING VERIFIED BRANCHES TO THE ANSWER

This ProverQA pair uses Qwen2.5-7B-Instruct on the same problem. The Backbone leaves a verified fact disconnected from its answer, whereas PROOF-R1 joins two verified branches to establish the correct conclusion.

Premises $\mathcal{P}$

- `p1` Vance will make groundbreaking discoveries if he conducts rigorous research.
- `p2` If Ivan studies ocean depths, then he either makes groundbreaking discoveries or inspires new generations, but not both.
- `p3` Vance conducts rigorous research.
- `p4` Vance studies ocean depths.
- `p5` If Vance studies ocean depths, then he either makes groundbreaking discoveries or inspires new generations, but not both.

Query q

Vance inspires new generations.

Candidates $\mathcal{Y}$ $y^* = y_2$

- y_1 Vance inspires new generations.
- y_2 Vance does not inspire new generations.
- y_3 It is uncertain whether Vance inspires new generations.

Backbone $\tilde{y} = y_3$

s1 Derives the discovery fact from Vance's rigorous research.
$D_t =$ `[p3, p1]` $c_t =$ `make_groundbreaking_discoveries(Vance)` $\rho_t =$ IE

s2 Derives the exclusive choice from Vance's study of ocean depths.
$D_t =$ `[p4, p5]` $c_t =$ `make_groundbreaking_discoveries(Vance)` $\oplus$ `inspire_new_generations(Vance)` $\rho_t =$ XOI

s3 States that discovering and inspiring cannot both hold.
$D_t =$ `[s2]` $c_t = \neg($`make_groundbreaking_discoveries(Vance)` $\wedge$ `inspire_new_generations(Vance)`$)$ $\rho_t =$ MT

h_goal_uncertain Selects uncertainty without using the discovery fact from `s1`.
$D_t =$ `[s2, s3]` $c_t =$ `inspire_new_generations(Vance)` $\rho_t =$ GB

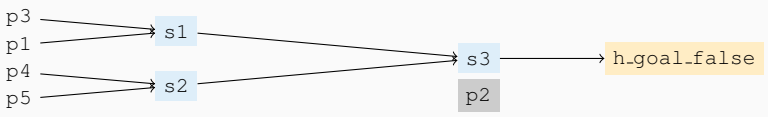
PROOF-R1 $\tilde{y} = y_2$

s1 Vance's research establishes that he makes groundbreaking discoveries.
$D_t =$ `[p3, p1]` $c_t =$ `make_groundbreaking_discoveries(Vance)` $\rho_t =$ IE

s2 His study of ocean depths establishes the exclusive choice in `p5`.
$D_t =$ `[p4, p5]` $c_t =$ (`make_groundbreaking_discoveries(Vance)` $\oplus$ `inspire_new_generations(Vance)`) $\rho_t =$ IE

s3 Join both results: making discoveries excludes inspiring new generations.
$D_t =$ `[s1, s2]` $c_t = \neg($`inspire_new_generations(Vance)`$)$ $\rho_t =$ XOE

h_goal_false Bind this conclusion to the selected candidate y_2.
$D_t =$ `[s3]` $c_t = \neg($`inspire_new_generations(Vance)`$)$ $\rho_t =$ GB



The gray-highlighted Backbone conclusion at `s1` passes all four checks, yet it lies outside the dependency closure of `h_goal_uncertain`. Its analysis likewise treats the exclusive choice as unresolved despite having established the discovery fact. Red marks the rule mismatch at `s2`, the dependency at `s3` missing from the rule-verified state, and the final positive conclusion that fails semantic verification while the model selects y_3. In PROOF-R1, the blue references join `s1` and `s2` at `s3`: the established discovery resolves the exclusive choice and yields the yellow correct negative answer. All four actions pass schema, semantic, and rule verification. Backward traversal from the final node `s3` includes both branches and premises `p1`, `p3`, `p4`, and `p5`, leaving Ivan's `p2` outside the closure in gray. PROOF-R1 *can identify answer-relevant dependencies and integrate verified results across reasoning branches to construct a coherent proof of the correct answer.*