

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 PROCESS-VERIFIED REINFORCEMENT LEARNING FOR THEOREM PROVING VIA LEAN

Anonymous authors

Paper under double-blind review

ABSTRACT

While reinforcement learning from verifiable rewards (RLVR) typically has relied on a single binary verification signal, symbolic proof assistants in formal reasoning offer rich, fine-grained structured feedback. This gap between structured processes and unstructured rewards highlights the importance of feedback that is both dense and sound. In this work, we demonstrate that the Lean proof assistant itself can serve as a symbolic process oracle, supplying both outcome-level and fine-grained tactic-level verified feedback during training. Proof attempts are parsed into tactic sequences, and Lean’s elaboration marks both locally sound steps and the earliest failing step, yielding dense, verifier-grounded credit signals rooted in type theory. We incorporate these structured rewards into a GRPO-style reinforcement learning objective with first-error propagation and first-token credit methods that balances outcome- and process-level advantages. Experiments with STP-Lean and DeepSeek-Prover-V1.5 show that tactic-level supervision outperforms outcome-only baselines in most settings, delivering improvements on benchmarks such as MiniF2F and ProofNet. Beyond empirical gains, our study highlights a broader perspective: symbolic proof assistants are not only verifiers at evaluation time, but can also act as process-level reward oracles during training. This opens a path toward reinforcement learning frameworks that combine the scalability of language models with the reliability of symbolic verification for formal reasoning.

1 INTRODUCTION

Automated theorem proving (ATP) is one of the long-term goals of AI (Newell et al., 1957). Compared to reasoning in natural language (NL), which often contains vague or ambiguous symbols, formal theorem proving based on formal logic and type theory provides technical and precise language for proving mathematical theorem (Church, 1940; Fitting, 1996). Currently, interactive theorem provers (ITP) such as Lean (de Moura et al., 2015; Moura & Ullrich, 2021), Isabelle (Nipkow et al., 2002) and Coq (Barras et al., 1997), serve as reliable and powerful tools for verifying mathematical proofs. Lean proofs are sequences of tactics, with automation handling routine arithmetic/logic and verification-so ITPs provide a middle ground between full automation and human guidance

By contrast, LLMs model next-token probabilities from large corpora via pre- and post-training, learning lexical correlations rather than rule-based symbolic manipulation (Brown et al., 2020). With further techniques such as instruction tuning and Reinforcement Learning from Human Feedback (RLHF), LLMs have evolved to handle a wide range of tasks, including question answering, summarization, dialogue (Ouyang et al., 2022; Bai et al., 2022). In particular, reinforcement learning (RL) approaches for reasoning tasks aim to enhance the model’s reasoning ability by encouraging the generation of long chains of thought rationale (DeepSeek-AI et al., 2025; OpenAI et al., 2024).

Compared to other reasoning tasks which often verify or reward LLMs’ response according to its final answer (Cobbe et al., 2021), the theorem prover can verify the correctness of entire proof when LLMs respond with formal language. In this context, given the human-in-the-loop nature of ITPs, there have been growing attempts to use LLMs for formal theorem proving tasks (AlphaProof and AlphaGeometry teams, 2024; Trinh et al., 2024). LLMs act as prover agents while theorem provers serve as verifiers, being used either at inference time-to search and validate tactics and premises-or for augmenting formal reasoning datasets with verified samples (Lample et al., 2022; Wang et al.,

054 2023; Ying et al., 2024a; Zhu et al., 2025). Furthermore, some recent studies incorporated binary
 055 feedback from the Lean theorem prover into its online RL framework (Xin et al., 2024b).
 056

057 The tactic-based proof structure in Lean contains information relevant for reasoning tasks such as
 058 the positions of tactics or the nature of proof errors or failures. This structured information captures
 059 not just the outcome of a proof, but also the underlying reasoning process. However, despite its
 060 potential, only a few works have explored incorporating this kind of fine-grained supervision into
 061 the training of LLMs (Ji et al., 2025). At the same time, recent RL approaches for reasoning have
 062 increasingly emphasized the use of process-based reward models (PRMs) to guide model behavior.
 063 While these models show promising performance, there is still a lack of clarity around how PRMs
 064 are constructed, how the reasoning step or step reward should be defined, what training signals or
 065 datasets they should depend on (Yuan et al., 2024; Luo et al., 2024; Cui et al., 2025).

066 Unlike recent approaches that rely on PRMs or long NL CoT (Lin et al., 2025a;b), we directly
 067 leverage the Lean proof assistant as a *symbolic process oracle* during RL training, without any
 068 natural-language guidance. For each generated proof, Lean provides (i) a global outcome signal and
 069 (ii) fine-grained tactic-level feedback via info trees and error logs.

070 While fine-grained tactic level signals are available from Lean, leveraging them effectively during
 071 RL training is nontrivial. Lean outputs symbolic, tree-structured language feedback, such as proof
 072 states and error locations, whereas LLMs operate over autoregressive token sequences and learn
 073 from scalar rewards in RL. This representational mismatch creates a credit-assignment challenge:
 074 symbolic verifier feedback must be transformed into structured token-level training signals. To
 075 bridge this gap, we introduce a structured credit assignment framework for integrating symbolic
 076 verifier signals into an online RL objective, requiring three components: (i) a formulation for in-
 077 corporating fine-grained signals, (ii) a principled rule for reducing Leans symbolic feedback into
 078 per-tactic scores, and (iii) a mapping from per-tactic scores to token-level advantages. We instanti-
 079 ate this pipeline using a tactic-level MDP, a first-error propagation rule grounded in Leans semantics,
 080 and a first-token credit assignment strategy.

081 We integrate the resulting per-tactic signals into a Group Relative Policy Optimization (GRPO) style
 082 objective combining outcome- and process-level advantages. This enables precise, type-theoretic
 083 credit assignment grounded in verifier feedback without the need for an auxiliary PRM. Empirically,
 084 we found that incorporating symbolic verifier feedback into the RL objective consistently improves
 085 performance on MiniF2F and ProofNet, demonstrating the value of fine-grained verifier signals for
 086 reliable credit assignment in reasoning tasks. Our key contributions are as follows:

- 087 • **Formalizing Lean’s Symbolic Feedback.** We formalize Leans symbolic, tactic-level feed-
 088 back and reduce it into scalar training signals that enable fine-grained, token-level credit
 089 assignment.
- 090 • **Symbolic verifier-guided RL.** We integrate outcome and tactic-level rewards derived from
 091 Lean into an RL framework, providing dense and verifiable credit assignment.
- 092 • **Stable improvements on benchmarks.** On MiniF2F and ProofNet, our approach consis-
 093 tently outperforms both outcome-only RL and vanilla baselines, yielding more stable and
 094 robust gains without NL notation or external PRM.

095 2 RELATED WORK

096 **Automatic Theorem Proving** An automated theorem prover typically consists of two stages.
 097 The first is autoformalization, i.e., translating natural language mathematical statements into formal
 098 ones. LLMs have been used for this task (Wu et al., 2022), producing datasets such as MiniF2F,
 099 ProofNet, Deepseek-Prover, and LeanWorkbook (Zheng et al., 2022; Azerbayev et al., 2023; Xin
 100 et al., 2024a; Ying et al., 2024a). The second stage is proof generation, which can be performed
 101 step-by-step via tree search (Polu & Sutskever, 2020; Azerbayev et al., 2024; Wu et al., 2024; Xin
 102 et al., 2024b) or by generating entire proofs at once (Xin et al., 2024a; Lin et al., 2025b). Existing
 103 approaches such as Lean-STaR and RMaxTS use Lean only as a step-checker during inference (Lin
 104 et al., 2025a; Xin et al., 2024b), whereas recent work has employed Lean as a whole-proof verifier
 105 during training (Wang et al., 2025a; Zhang et al., 2025; Ren et al., 2025). In this paper, we go
 106 beyond step-checking or whole-proof verification by using Leans fine-grained, tactic-level feedback
 107 as process-based rewards in online RL training.

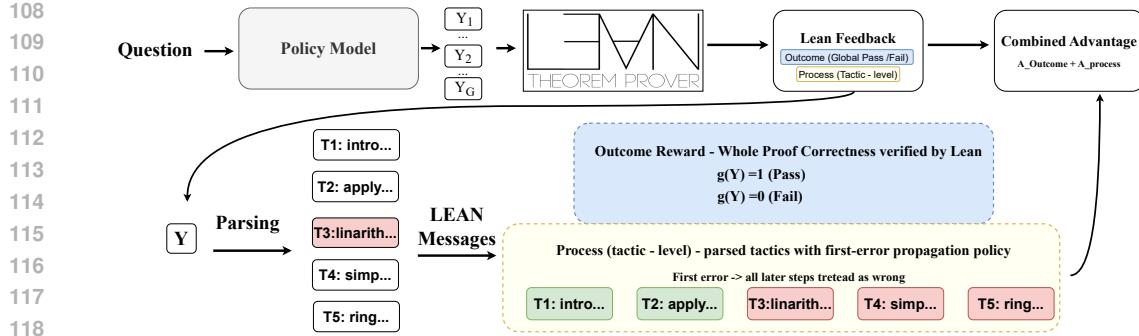


Figure 1: Overall framework for combining outcome and tactic level rewards via Lean: the proof Y is parsed into tactics T_1, \dots, T_5 , with Lean providing outcome feedback $g(Y)$ and step-level errors (e.g., failure at T_3 invalidates later tactics). Rewards are then assigned to the first token of each tactic.

Reinforcement Learning in Language Models Beyond algorithmic advances such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024), reward shaping and credit assignment remain core challenges in RL. Outcome-based rewards (Cobbe et al., 2021), though widely used in RLHF, suffer from sparsity (Chan et al., 2024; Zheng et al., 2023). Process-based reward models (PRMs) address this by assigning step-level rewards (Lightman et al., 2023; Setlur et al., 2024; Kazemnejad et al., 2024; Yuan et al., 2024; Cui et al., 2025). Rewards can be defined implicitly (Cui et al., 2025) or explicitly via correctness annotations (Lightman et al., 2023) or Monte Carlo rollout success rates (Wang et al., 2024), but existing methods require large annotated datasets of step-level correctness. This motivates our approach of leveraging the Lean prover itself as a process oracle, automatically verifying each step without human labels or sampling. Additional discussion is in Appendix A.

3 PRELIMINARIES

3.1 LEAN4

In Lean theorem proving, a statement to be established is represented as an initial goal and incrementally reduced into subgoals through a sequence of tactics. Each tactic is parsed and elaborated by unifying it with lemmas or theorems in the library, generating new subgoals, and verifying their validity. The elaboration stage produces structured info trees that record proof states and error messages. Finally, the kernel ensures that the elaborated proof is type-theoretically consistent and constitutes a valid proof for the original theorem.

Formally, let x denote a theorem statement provided to an LLM, and let Y be the response, a proof expressed in the Lean language. Write \mathcal{Y} for the set of Lean proofs and \mathcal{T} for the set of tactics. For $Y \in \mathcal{Y}$, we view a proof Y as a sequence of tactics $(T_1, T_2, \dots, T_{N(Y)})$ parsed from the Abstract Syntax Tree (AST) and sorted by their starting positions, where $N(Y)$ is the number of tactics in Y which aligns with the LLM’s autoregressive generation process. Each tactic T_i comprises corresponding tokens y_t in Y . Lean represents tactics as AST nodes; each node encodes the tactics syntactic structure and binding context, and may carry metadata such as error messages, proof states, and an index through which users (or training frameworks) can interact with Lean. If a tactic does not appear in the error log, then it has been elaborated successfully and passed Lean’s internal rule-based verification, which guarantees that the step is locally sound under dependent type theory. Thus, any tactic not marked as an error constitutes a verified reasoning step—even if it does not contribute to closing the proof because some subgoals remain or later tactics fail. In other words, Lean ensures tactic-level soundness, while proof-level completeness depends on whether the entire sequence resolves all goals. Leveraging this parsing and validation feedback, we define the parsing function $f : \mathcal{Y} \rightarrow \mathcal{T}^*$ to be the sequence obtained by sorting $\text{TacSet}(Y)$: $f(Y) = (T_1, \dots, T_{N(Y)})$. We also define the global scoring function $g : \mathcal{Y} \rightarrow [0, 1]$, where $g(Y) = 1$ if Y passes the Lean verifier and 0 otherwise, and the per-tactic scoring function $\varphi : \{(Y, T) \mid Y \in \mathcal{Y}, T \in \text{TacSet}(Y)\} \rightarrow \{1, d_1, d_2\}$.

162 Specifically,

163

$$\varphi(Y, T) = \begin{cases} 1, & \text{if } g(Y) = 1, \\ d_1, & \text{else if } g(Y) = 0 \text{ and } T \text{ has no errors in Lean,} \\ d_2, & \text{else if } g(Y) = 0 \text{ and } T \text{ contains errors.} \end{cases}$$

166 Combining these components, we represent Lean’s role via f, g, φ as

167

$$\begin{aligned} \text{Lean} : \mathcal{Y} &\rightarrow \{0, 1\} \times (\mathcal{T} \times \{1, d_1, d_2\})^*, \\ \text{Lean}(Y) &= (g(Y), [(T_1, \varphi(Y, T_1)), \dots, (T_{N(Y)}, \varphi(Y, T_{N(Y)}))])_{f(Y)=(T_1, \dots, T_{N(Y)})}. \end{aligned}$$

171 **3.2 TACTIC-LEVEL MDP**

173 We define a tactic-level Markov Decision Process (MDP) as the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, F, m)$. The
 174 state space \mathcal{S} contains partial formal proofs; each $s \in \mathcal{S}$ is the proof prefix produced so far. The
 175 action space \mathcal{A} coincides with the tactic space \mathcal{T} ; each action $a \in \mathcal{A}$ is a single Lean tactic. The
 176 reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ assigns a tactic-level reward $r(s, a)$. The transition function
 177 $F : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is deterministic: $s_{j+1} = F(s_j, a_j) = s_j \oplus a_j$, where \oplus denotes concatenation
 178 of the tactic a_j to the proof s_j at time step j . Transitions are pure concatenations; Lean feedback
 179 affects r , not F . Let $\mathcal{S}_{\text{term}} \subseteq \mathcal{S}$ be EOS absorbing states. Let $m \in \mathcal{S}$ be the initial state. In Section 4,
 180 we extend this formulation with outcome- and tactic-level rewards derived from the Lean theorem
 181 prover to obtain the final training signal.

182 **3.3 CREDIT ASSIGNMENT IN REINFORCEMENT LEARNING**

183 PPO assigns a sparse end-of-sequence reward and propagates credit with a value model with Gener-
 184 alized Advantage Estimate (GAE), reducing variance at the cost of extra learning complexity; full
 185 details are deferred to Appendix E.

186 In contrast, REINFORCE style GRPO optimizes directly from verifiable whole-trajectory rewards
 187 without a value model. For a prompt q , we sample G responses $\{y_i\}_{i=1}^G$ from π_{old} and obtain
 188 rewards r_i . A normalized, response-level advantage is applied uniformly to all tokens of y_i :

189

$$\hat{A}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}.$$

190 The objective is

191

$$L_{\text{GRPO}}(\theta) = \mathbb{E} \left[\frac{1}{G} \sum_{i=1}^G \left\{ \min \left(\frac{\pi_\theta(y_i | q)}{\pi_{\theta_{\text{old}}}(y_i | q)} \hat{A}_i, \text{clip} \left(\frac{\pi_\theta(y_i | q)}{\pi_{\theta_{\text{old}}}(y_i | q)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_i \right) - \beta D_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}] \right\} \right].$$

192 We make this dense and sound by injecting *Lean-derived tactic advantages* into GRPO: the outcome
 193 signal remains at response level, while tactic-level signals are mapped to tokens at the first token of
 194 each tactic (Sec. 4). This preserves GRPO’s simplicity while addressing sparse credit.

195 **4 METHOD**

196 **4.1 DEFINE TACTIC-LEVEL REWARDS**

197 In the previous section, we modeled the correctness of proofs Y generated by the Lean proof as-
 198 sistant and parsed and verified each tactic within Y . We now introduce a reward mechanism that
 199 integrates both outcome-based and process-based signals explicitly into the RL framework. Specifi-
 200 cally, we employ an outcome-based reward defined through a function $g(Y)$, similar to approaches
 201 used by (DeepSeek-AI et al., 2025), as a global reward evaluating the entire proof. Additionally, we
 202 define a process-based reward $\varphi(Y, T)$, assessing the correctness or validity at the level of individual
 203 tactics $T \in Y$. Unlike implicit rewards or Monte Carlo estimations typically interpreted as process
 204 rewards, our method explicitly assigns correctness-based rewards at each tactic step.

205 Assume that, analogous to the GRPO training rollout framework, given a question q , an LLM gener-
 206 ates a group of responses $\{Y_1, Y_2, \dots, Y_G\}$. Lean produces an outcome-based rewards:

$$r_{\text{outcome}}(Y_i) = g(Y_i)$$

207 We define the outcome-based advantage for any token $y_{i,t}$ in response Y_i as:

208

$$A_{\text{outcome}, i, t} = \frac{g(Y_i) - \text{mean}(g(Y_1), \dots, g(Y_G))}{\text{std}(g(Y_1), \dots, g(Y_G))}.$$

Beyond binary outcome verification signals, we further design elaborate rewards based on the AST feedback produced by the Lean parser as in section 3.1. We leverage this AST feedback to distinguish between different kinds of tactics: for example, whether a tactic is elaborated successfully (i.e., type-correct and locally sound), but may still leave unresolved subgoals that prevent the proof from being completed, or whether it has type errors or parser-level mismatches. This structured feedback allows us to assign more fine-grained process-based rewards. Since, we sorted the tree node containing proof state by increasing order, we apply a First Error Propagation rule when mapping Lean’s feedback into tactic-level rewards as (Lu et al., 2024; Lightman et al., 2023). Given a sequence of tactics (T_1, \dots, T_N) , once an error is observed at T_j , we propagate this failure to all subsequent tactics, i.e., every T_k with $k \geq j$ is treated as erroneous for the purpose of reward assignment.

$$\text{Let } j = \min\{i : T_i \text{ contains an error}\}. \quad \varphi(Y, T_k) = \begin{cases} 1, & g(Y) = 1. \\ d_1, & g(Y) = 0 \text{ and } k < j \text{ and no error,} \\ d_2, & g(Y) = 0 \text{ and } k \geq j, \end{cases}$$

Unlike Lean, which parses proofs into a tree structure, the LLM generates proofs in an autoregressive, causal manner. Once the first erroneous tactic T_j occurs, the continuation T_{j+1}, \dots, T_N is conditioned on an invalid prefix, and therefore cannot constitute a valid reasoning process. First-error propagation enforces this principle by assigning error signals to all subsequent tactics, ensuring causal and type-theoretic credit assignment.

For any arbitrary response Y_i , composed of tactics $Y_i = \{T_{i,1}, T_{i,2}, \dots\}$, if we set s_j, a_j as the state and tactic $T_{i,j}$ at step j in response Y_i , the process-based reward for tactic $T_{i,j}$ is:

$$r_{\text{process}}(s_j, a_j) = r_{\text{process}, i, j} = \varphi(Y_i, T_{i,j}).$$

The corresponding process-based advantage is

$$A_{\text{process}, i, j} = r_{\text{process}, i, j} - \text{mean}(g(Y_1), \dots, g(Y_G)).$$

Here, the subtraction of the mean outcome reward serves as a dynamic baseline reflecting the difficulty of the problem q as GRPO algorithm. If the problem is easier, the mean outcome reward becomes higher, thus penalizing incorrect proofs and their tactics more heavily. Conversely, for more challenging problems, the lower baseline imposes less severe penalties.

4.2 INTEGRATING LEAN INTO TACTIC-BASED REINFORCEMENT LEARNING

We then integrate these two types of advantages into the standard GRPO objective as follows.

$$A_{i,t} = A_{\text{outcome}, i, t} + \mathbf{1}\{t = \text{first}(T_{i,s(i,t)})\} \cdot A_{\text{process}, i, s(i,t)},$$

where $s(i, t) \in \{1, \dots, N\}$ is the index of the tactic containing the token t in Y_i , $\text{first}(T_{i,j})$ indicates the first token of the tactic. i.e., we assign the tactic advantage only to the first token of each tactic. We applied the advantage $A_{i,t}$ into GRPO objective function:

$$L(\theta) = \mathbb{E}_{q \sim P(Q), \{Y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(Y|q)} \left[\frac{1}{G} \sum_{i=1}^G \left\{ \frac{1}{|Y_i|} \sum_{t=1}^{|Y_i|} \min \left(\rho_{i,t} A_{i,t}, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon) A_{i,t} \right) - \beta D_{\text{KL}}[\pi_{\theta} \parallel \pi_{\text{ref}}] \right\} \right]. \quad (1)$$

where $\rho_{i,t} = \frac{\pi_{\theta}(y_{i,t}|q, Y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|q, Y_{i,<t})}$. This formulation explicitly leverages both the global correctness signal $A_{\text{outcome}, i, t}$ from proof outcomes and the detailed, tactic level correctness assessment $A_{\text{process}, i, s(i,t)}$. By combining them into a single advantage $A_{i,t}$, we enrich the learning signal provided to the LLM based proof generator under the GRPO framework. In Appendix J, we provide a mathematical grounding and interpretation of our method via a potential-based reward shaping.

Rather than propagating cumulative rewards across an entire proof trajectory, we collapse credit assignment to Lean-verified, tactic-level signals. In general RL, a suboptimal step may still obtain positive return if later rewards are high, but in mathematical proof, this could be unsound: once a tactic fails, all subsequent steps are invalid under first-error propagation. Empirically, return-based credit led to unstable optimization, as it requires a value function or auxiliary estimator to normalize scale and reduce variance. Hence, we adopt a simpler formulation that combines normalized outcome-level signals with tactic-level rewards, without computing returns (See Appendix G).

¹**Budgets are not directly comparable:** tree-search budgets count expansions/verifier calls at inference, whereas our budgets count whole-proof samples. Our aim is to improve single-shot generation under a different compute regime.

Model	Model size	Budget ¹	MiniF2F-Test	ProofNet-Test
Whole-Proof Generation Methods				
DeepSeek-Prover-V1.5-SFT (Xin et al., 2024a)	7B	32	46.2% \pm 0.2	14.3% \pm 0.3
		64	47.5% \pm 0.1	15.05% \pm 1
DeepSeek-Prover-V1.5-RL (Xin et al., 2024a)	7B	32	48% \pm 0	16% \pm 1
		64	48.8% \pm 0.4	17.4% \pm 0.6
Goedel-Prover-SFT (Lin et al., 2025c)	7B	32	56.9% \pm 0.4	15.6% \pm 0.5
		64	57.9% \pm 0.5	16.7% \pm 0
STP-Lean (Dong & Ma, 2025)	7B	32	55.9% \pm 0.2	17.2% \pm 0
		64	56.7% \pm 0.2	19.1% \pm 0.4
STP-Lean + Ours	7B	32	57.1% \pm 0.8	18.6% \pm 0.3
		64	59.2% \pm 0.5	19% \pm 0.3
DeepSeek-Prover-V1.5 + STP	7B	32	54.9% \pm 0.7	16.8% \pm 0.3
		64	57.2% \pm 0.2	17.7% \pm 0
DeepSeek-Prover-V1.5 + STP + Ours	7B	32	56.3% \pm 0.6	17.6% \pm 0.8
		64	57.8% \pm 0.4	18.5% \pm 0.3
Tree Search Methods				
Lean-STaR	7B	64 \times 1 \times 50	46.3%	–
InternLM2-Math-Plus-7B (Ying et al., 2024b)	7B	1 \times 32 \times 100	48.8%	–
InternLM2.5-StepProver	7B	4 \times 32 \times 600	58.5% \pm 0.9	–
DeepSeek-Prover-V1.5-RL + RMaxTS (Xin et al., 2024a)	7B	3,200	55.0% \pm 0.7	21.5% \pm 0.8

Table 1: Budgets for whole-proof methods denote the *sample budget* (N) per problem; for tree-search methods, budgets denote the authors reported *search expansions counts*. We compare with InternLM family and DeepSeek-Prover based tree search methods for fair comparison with our method. Bold indicates the best number within the whole-proof block. All our GRPO-style runs use the same STP subset, generations per query, and a 15s Lean timeout. The notation $\mu \pm \sigma$ indicates the mean and the standard deviation each.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We trained on 10k samples randomly drawn from the STP dataset (3.26M proofs). Proofs were verified via Lean through a REPL interface, with a 15s timeout per attempt. Baselines included STP-Lean and DeepSeek-Prover-V1.5-SFT, the latter additionally fine-tuned on 500k STP samples before RL. We used non-CoT prompt, response styles as in (Xin et al., 2024b). We used tactic-level rewards $d_1 = -0.05$ and $d_2 = -0.1$ for the main experiment. Full hyperparameters and training details are provided in Appendix B.

5.2 MAIN RESULTS

In Table 1, the results on both the MiniF2F and ProofNet datasets demonstrate that training with tactic-based advantage via Lean consistently enhances model performance across most evaluation settings. For the STP-Lean model, our method improves MiniF2F performance up to +2.5% (pass@64), and ProofNet performance by +1.4%p (pass@32), while showing a negligible decrease of -0.1% p on pass@64. Similarly, for DeepSeek-Prover-V1.5, our approach achieves marginal yet consistent increases across all benchmarks.

Across both MiniF2F and ProofNet, leveraging Lean as a *process-level oracle* yields consistent, stable gains over outcome-only reinforcement learning, without increasing training cost. In particular, in Table 2, when applied to DeepSeek-Prover models, GRPO fails to yield any gains on the ProofNet-Test set, and in some cases even underperforms relative to the supervised baseline. This highlights a key limitation of purely outcome-based credit assignment: it often lacks stability and fails to provide consistent guidance for proof search.

By comparison, tactic-level credit assignment yields more reliable improvements. While minor drops appear in some settings, it generally provides stable gains over outcome-only GRPO. For example, on MiniF2F (pass@64), STP-Lean + Ours improves by +2.5%p over the baseline, compared

Model	Model Size	Budget	MiniF2F - Test	ProofNet - Test
STP + Outcome only (GRPO)	7B	32	55.7% \pm 1	17.4% \pm 0.6
		64	57.9% \pm 0.5	19% \pm 0.3
STP + Tactic only	7B	32	55.6% \pm 0.6	18.3% \pm 0
		64	56.8% \pm 0.6	17.9% \pm 0.8
STP + Outcome+Tactic RL (ours)	7B	32	57.1% \pm 0.8	18.6% \pm 0.3
		64	59.2% \pm 0.5	19% \pm 0.3
DeepSeek-Prover-V1.5 + Outcome only (GRPO)	7B	32	55.3% \pm 0.4	16.8% \pm 0.8
		64	57.4% \pm 0.4	17.6% \pm 0.8
DeepSeek-Tactic only	7B	32	54.9% \pm 0.7	16.8% \pm 0.8
		64	57.8% \pm 1	17.6% \pm 0.3
DeepSeek-Prover-V1.5 + Outcome+Tactic RL (ours)	7B	32	56.3% \pm 0.6	17.6% \pm 0.8
		64	57.8% \pm 0.4	18.5% \pm 0.3

Table 2: Ablation study of STP-Lean with various verifier methods on MiniF2F-Test and ProofNet-Test benchmarks.

to +1.2%p with GRPO. As shown in Table 1 and 2, tactic-based training consistently matches or surpasses both the supervised baseline and GRPO. Importantly, this stability comes with almost no extra cost: since both methods already use REPL interactions with Lean, the additional sorting and scoring overhead is negligible.

Compared to strong search-based baselines (e.g., InternLM families, DeepSeek-Prover-RL+RMaxTS), our single-shot, whole-proof training approaches their reported accuracy (e.g., 59.2% vs. 58.5% pass@64 on MiniF2F) while avoiding large search-time compute.

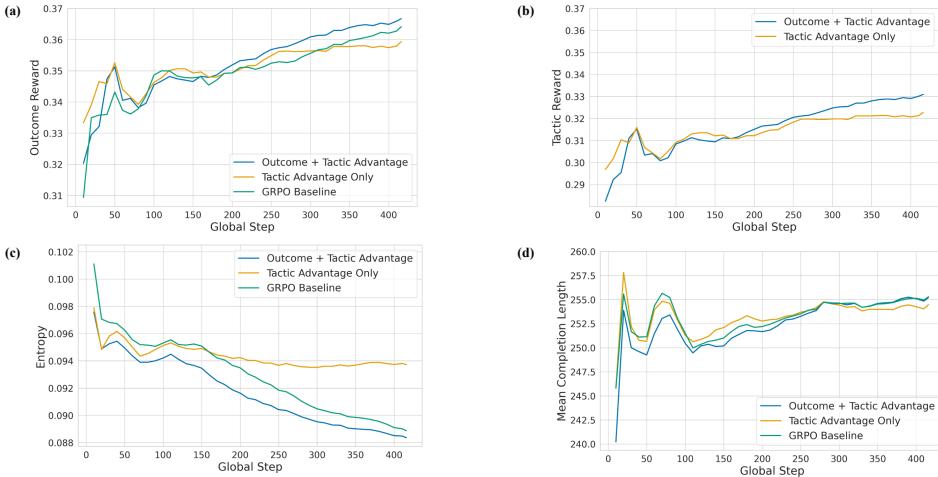


Figure 2: Training dynamics showing (a) outcome reward, (b) tactic reward, (c) entropy, and (d) mean of response length during reinforcement learning.

5.3 ANALYSIS

The Role of Outcome and Tactic Rewards. Integrating both outcome-level and tactic-level signals yields more effective learning than employing either signal in isolation. Outcome-only RL, as in GRPO, is constrained by the sparsity of binary feedback: improvements are gradual and the final performance plateaus at a relatively low level (Figure 2(a)). In contrast, tactic-only training provides dense feedback but lacks a global objective, resulting in premature convergence. When combined, outcome rewards serve as a global objective function, while tactic rewards provide local credit assignment, enabling both rapid progress and higher performance. This complementary relationship is further reflected in Figure 2(b), where tactic-only supervision's tactic reward plateaus, but outcome-tactic combined rewards continue to increase steadily. The results in Table 2 supports this finding:

Model	Model Size	Sample Budget	MiniF2F - Test	ProofNet - Test
All tokens	7B	32	56.3% \pm 0.6	18.1% \pm 0.8
		64	57.8% \pm 0.7	18.1% \pm 0.8
Entropy-based	7B	32	56.4% \pm 0.2	17.9% \pm 0.8
		64	57.1% \pm 0.5	18.5% \pm 0.3
Last token	7B	32	56.7% \pm 0.9	17.2% \pm 0
		64	57.5% \pm 0.6	17.7% \pm 0.5
First token	7B	32	57.1% \pm 0.8	18.6% \pm 0.3
		64	59.2% \pm 0.5	19% \pm 0.3

Table 3: Ablation study of STP-Lean on how to distribute tactic-level advantages across tokens.

outcome signals enforce proof-level correctness, while tactic signals supply verifiable intermediate feedback; only their integration consistently improves performance across benchmarks.

Entropy and Proof Length. The use of fine-grained rewards influences exploration not by indiscriminately broadening the search space but by focusing learning on more informative decision points. As shown in Figure 2(c), outcome+tactic training converges to lower entropy than tactic-only and outcome-only settings, indicating that the policy becomes more decisive as training progresses. This does not correspond to mode collapse: Figure 2(d) shows that the average proof length remains stable across all methods, suggesting that the performance gains are not attributable to trivial lengthening of outputs. Instead, denser intermediate rewards appear to reduce the need for broad stochastic exploration, guiding the model toward more efficient proof strategies.

Tactic to Token Level Credit Assignment. After defining tactic-level rewards, next step is how to distribute them across tokens. In our main method, the tactic advantage is assigned only to the first token of the tactic. For comparison, we conducted ablations where the tactic advantage was instead (i) distributed to all tokens of a tactic, (ii) assigned only to the last token, (iii) keep first token reward distribution, but additionally choose 10% tokens within the tactic with respect to high entropy. As Wang et al. (2025d) showed that high entropy tokens could be reasoning drive tokens, we speculated that this method can automatically select the tokens for serving as fork in formal reasoning. Assigning credit to the first token of each tactic achieves the most stable and consistent improvements, as evidenced by Table 3. Alternative strategies do not yield comparable gains and in some cases even degrade performance. This outcome aligns with the semantics of Lean proofs: the first token corresponds to the tactic keyword (e.g., `intro`, `apply`, `have`), determining the subsequent proof strategy and constrains the structure of subgoals. Concentrating credit on this decision point enhances the models ability to select tactics appropriately, resulting in more reliable downstream reasoning. This finding is also aligned with (Fang et al., 2025), showing that focusing on key tokens during training improves performance on long-context tasks.

Reward Strategy for Tactic-level Feedback. For tactic-level feedback to be effective, it must reflect the sequential dependency of proof construction, account for task difficulty, and distinguish between partially correct and erroneous steps. The first-error propagation rule ensures that once an error occurs, subsequent tactics are treated as invalid; removing this rule significantly reduces performance (Table 4), because once the first error occurs, the remaining tactics are evaluated in an invalid context and cannot salvage correctness. Incorporating a difficulty-normalized baseline further stabilizes training, while its absence leads to degraded results. Finally, differentiating penalties between partially correct tactics and outright erroneous ones proves essential: collapsing these into a single penalty $d_1 = d_2$ yields inconsistent outcomes- improvements on MiniF2F but declines on ProofNet. These results indicate that an effective tactic-level reward scheme must combine sequential error propagation, difficulty-aware normalization, and differentiated penalties in order to provide stable and semantically faithful learning signals. In the sensitivity analysis of Appendix C, assigning different values to d_1 and d_2 leads to robust performance, tending to outperform the GRPO baseline and yielding the strongest improvements on MiniF2F.

Effect of Verification Timeouts When using Lean as a verifier, long proofs can lead to excessive verification time, so we introduced timeout thresholds of 5, 10, 15, and 30s (Figure 3). A 5s limit gave the worst results, since even relatively simple proofs often exceeded this window and produced too few valid reward signals. In contrast, 10-30s yielded much stronger performance, with

Model	Model Size	Sample Budget	MiniF2F - Test	ProofNet - Test
No First Error	7B	32	56.4% \pm 0.9	17.4% \pm 0.3
		64	58.2% \pm 0.7	18.3% \pm 0.3
No Baseline	7B	32	56.7% \pm 0.2	17.9% \pm 0.3
		64	57.4% \pm 0.7	18.3% \pm 0.5
Same tactic reward	7B	32	57.7% \pm 0.2	17.6% \pm 0.6
		64	58.7% \pm 0.8	18.1% \pm 0.6
Outcome+Tactic RL (ours)	7B	32	57.1% \pm 0.8	18.6% \pm 0.3
		64	59.2% \pm 0.5	19% \pm 0.3

Table 4: Ablation study on reward strategies for tactic-level feedback in STP-Lean. Additional experiments include removing the first-error propagation policy (No First Error), removing the baseline extraction (No Baseline), and using equal penalties for all tactics (Same tactic reward).

15s giving the best overall balance. Interestingly, 10-15s sometimes outperformed 30s despite the shorter allowance. We attribute this to the fact that discarding overly complex proofs biases training toward shorter and more efficient proof strategies. This effect is amplified in our setting because we evaluate non-CoT responses purely by Lean verification (without NL commentary): longer outputs are not only slower to check but also more error-prone. As a result, shorter verification limits encourage the model to generate concise, canonical proofs, which we hypothesize leads to better generalization at test time.

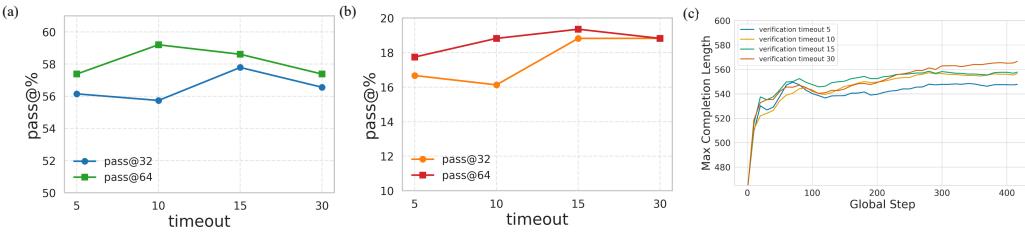


Figure 3: Ablation study of STP-Lean on different Lean verification timeouts (5, 10, 15, and 30 seconds) during outcome+tactic based training. We report evaluation performance on the MiniF2F and ProofNet benchmarks (a),(b), and the maximum response length observed during training (c).

Qualitative Analysis We conduct a qualitative analysis to better understand the differences between our tactic-reward-based approach and the baseline STP model. Specifically, we examine proofs from two benchmark problems: Imo_1960_p2 in the MiniF2F benchmark and 1_14 from ProofNet (See Appendix F). Table 8 presents the proof generated by our tactic-reward model, while Table 9 shows the corresponding proof from the STP model. The key difference in the first example lies in how the upper bound $x < 45/8$ is established. The STP model attempts to use the nonlinear inequality tactic `nlinarith`, which results in an error. By contrast, our tactic-reward model learns to penalize such invalid tactic choices. Instead, it carefully applies previously proven assumptions and intermediate lemmas before invoking `nlinarith`, thereby producing a correct and more robust proof. The second example comes from the ProofNet benchmark (1_14 exercise). As shown in Table 10, the tactic-reward model begins by normalizing the problem using a rewrite tactic. In contrast, the baseline model in Table 10 skips this normalization step and directly attempts inequality manipulations, which ultimately causes the proof to fail. Analysis for Failure case is in Appendix H. These anecdotal examples illustrate plausible mechanisms; for definitive evidence, see Table 1.

6 CONCLUSION

We introduced a reinforcement learning framework that uses the Lean proof assistant as a process-level reward oracle. Unlike prior outcome-only methods, our approach leverages Lean’s parsing and validation to provide both global outcome signals and fine-grained tactic rewards, integrated into a GRPO objective. This enables denser, verifiable credit assignment: outcome rewards enforce proof-level success, while tactic rewards guide step-level reasoning. Experiments on STP-Lean

486 and DeepSeek-Prover-V1.5 show consistent improvements on MiniF2F and ProofNet, with stable
 487 gains achieved by assigning tactic rewards to the first token of each tactic and first error propagation
 488 method. Overall, proof assistants can serve not only as checkers at inference but also as structured
 489 feedback sources during training, pointing toward more stable and effective RL for reasoning.
 490

491 LIMITATIONS

492 We did not compare against learned PRMs, as they rely on natural-language CoT supervision and
 493 large annotated datasets that are not yet available for Lean. Our models also generate pure Lean
 494 proofs without long CoT, leaving open how to design fine-grained rewards for long-form reasoning.
 495 In addition, tactic rewards in our method were fixed scores (d_1, d_2), which proved effective but
 496 somewhat sensitive across different models and datasets. Developing general advantage estimators
 497 and large-scale tactic-level datasets remains important future work.
 498

500 ETHICS STATEMENT

501 This research does not involve human subjects, personal data, or sensitive information that could
 502 raise ethical concerns. All experiments were conducted on publicly available formal mathematical
 503 datasets, and the proposed models are solely trained and evaluated for automated theorem proving
 504 tasks.
 505

507 REPRODUCIBILITY STATEMENT

509 To ensure the reproducibility of our work, we provide a comprehensive description of our and training
 510 process in Section 5.1. For further details such as hyperparameter, version of Lean, we introduced it in Appendix B. We utilized Huggingface and trl library for our experiments, and we plan
 511 to release the source codes to facilitate future research.
 512

515 REFERENCES

516 AlphaProof and AlphaGeometry teams. Ai achieves silver-medal standard
 517 solving international mathematical olympiad problems. Google DeepMind
 518 Blog, July 2024. URL <https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/>. Accessed: 17 April
 519 2025.
 520

522 Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev,
 523 and Jeremy Avigad. Proofnet: Autoformalizing and formally proving undergraduate-level mathe-
 524 matics, 2023. URL <https://arxiv.org/abs/2302.12433>.
 525

526 Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Al-
 527 bert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model
 528 for mathematics, 2024. URL <https://arxiv.org/abs/2310.10631>.
 529

530 Kaito Baba, Chaoran Liu, Shuhei Kurita, and Akiyoshi Sannai. Prover agent: An agent-based
 531 framework for formal mathematical proofs, 2025. URL <https://arxiv.org/abs/2506.19923>.
 532

533 Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova Dassarma, Dawn
 534 Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, John
 535 Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez,
 536 Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario
 537 Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Christopher Olah, Benjamin Mann, and
 538 Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human
 539 feedback. *ArXiv*, abs/2204.05862, 2022. URL <https://api.semanticscholar.org/CorpusID:248118878>.
 540

540 Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Jean-Christophe Filliâtre, Ed-
 541 uardo Giménez, Hugo Herbelin, Gérard Huet, César Muñoz, Chetan Murthy, Catherine Parent-
 542 vigouroux, Christine Paulin-Mohring, Amokrane Saibi, and Benjamin Werner. The coq proof
 543 assistant reference manual : Version 6.1. 06 1997.

544 Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhari-
 545 wal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal,
 546 Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M.
 547 Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz
 548 Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec
 549 Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL
 550 <https://arxiv.org/abs/2005.14165>.

551 Meng Cao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Scar: Shapley credit assignment
 552 for more efficient rlhf, 2025. URL <https://arxiv.org/abs/2505.20417>.

553 Alex J. Chan, Hao Sun, Samuel Holt, and Mihaela van der Schaar. Dense reward for free in re-
 554 inforcement learning from human feedback, 2024. URL <https://arxiv.org/abs/2402.00782>.

555 Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(3):
 556 114–115, 1940. doi: 10.2307/2266866.

557 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
 558 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John
 559 Schulman. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.

560 Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu
 561 Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang,
 562 Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning
 563 Ding. Process reinforcement through implicit rewards, 2025. URL <https://arxiv.org/abs/2502.01456>.

564 Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von
 565 Raumer. The lean theorem prover (system description). In *CADE*, 2015. URL <https://api.semanticscholar.org/CorpusID:232990>.

566 DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu,
 567 Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu,
 568 Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao
 569 Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan,
 570 Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao,
 571 Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding,
 572 Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang
 573 Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai
 574 Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang,
 575 Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang,
 576 Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang,
 577 Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang,
 578 R. J. Chen, R. L. Jin, Ruyi Chen, Shanghai Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng
 579 Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing
 580 Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen
 581 Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong
 582 Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu,
 583 Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xi-
 584 aoshua Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia
 585 Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng
 586 Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong
 587 Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong,
 588 Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxuan Liu, Yuyuan Zhou,

594 Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying
 595 Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda
 596 Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu,
 597 Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu
 598 Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforce-
 599 ment learning, 2025. URL <https://arxiv.org/abs/2501.12948>.

600 Kefan Dong and Tengyu Ma. Stp: Self-play llm theorem provers with iterative conjecturing and
 601 proving, 2025. URL <https://arxiv.org/abs/2502.00212>.

602

603 Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin
 604 Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling?,
 605 2025. URL <https://arxiv.org/abs/2410.23771>.

606 Melvin Fitting. *First-order logic and automated theorem proving (2nd ed.)*. Springer-Verlag, Berlin,
 607 Heidelberg, 1996. ISBN 0387945938.

608

609 Xinguang Ji, Yahui Liu, Qi Wang, Jingyuan Zhang, Yang Yue, Rui Shi, Chenxi Sun, Fuzheng
 610 Zhang, Guorui Zhou, and Kun Gai. Leanabell-prover-v2: Verifier-integrated reasoning for formal
 611 theorem proving via reinforcement learning, 2025. URL <https://arxiv.org/abs/2507.08649>.

612

613 Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée
 614 Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem
 615 provers with informal proofs, 2023. URL <https://arxiv.org/abs/2210.12283>.

616

617 Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy,
 618 Aaron Courville, and Nicolas Le Roux. Vineppo: Unlocking rl potential for llm reasoning through
 619 refined credit assignment, 2024. URL <https://arxiv.org/abs/2410.01679>.

620

621 Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel
 622 Ebner, Aurélien Rodriguez, and Timothée Lacroix. Hypertree proof search for neural theorem
 623 proving, 2022. URL <https://arxiv.org/abs/2205.11491>.

624

625 Chengpeng Li, Zhengyang Tang, Ziniu Li, Mingfeng Xue, Keqin Bao, Tian Ding, Ruoyu Sun,
 626 Benyou Wang, Xiang Wang, Junyang Lin, and Dayiheng Liu. Cort: Code-integrated reasoning
 627 within thinking, 2025. URL <https://arxiv.org/abs/2506.09820>.

628

629 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan
 630 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL
 631 <https://arxiv.org/abs/2305.20050>.

632

633 Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. Lean-star: Learning to interleave
 634 thinking and proving, 2025a. URL <https://arxiv.org/abs/2407.10040>.

635

636 Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou
 637 Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-prover: A frontier model for open-source
 638 automated theorem proving, 2025b. URL <https://arxiv.org/abs/2502.07640>.

639

640 Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan
 641 Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang,
 642 Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-prover-v2: Scaling
 643 formal theorem proving with scaffolded data synthesis and self-correction, 2025c. URL <https://arxiv.org/abs/2508.03613>.

644

645 Jianqiao Lu, Yingjia Wan, Zhengying Liu, Yinya Huang, Jing Xiong, Chengwu Liu, Jianhao Shen,
 646 Hui Jin, Jipeng Zhang, Haiming Wang, Zhicheng Yang, Jing Tang, and Zhijiang Guo. Process-
 647 driven autoformalization in lean 4, 2024. URL <https://arxiv.org/abs/2406.01940>.

648

649 Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li,
 650 Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in
 651 language models by automated process supervision, 2024. URL <https://arxiv.org/abs/2406.06592>.

648 Leonardo Moura and Sebastian Ullrich. *The Lean 4 Theorem Prover and Programming Language*,
 649 pp. 625–635. 07 2021. ISBN 978-3-030-79875-8. doi: 10.1007/978-3-030-79876-5_37.
 650

651 A. Newell, J. C. Shaw, and H. A. Simon. Empirical explorations of the logic theory machine: a case
 652 study in heuristic. In *Papers Presented at the February 26-28, 1957, Western Joint Computer
 653 Conference: Techniques for Reliability*, IRE-AIEE-ACM '57 (Western), pp. 218230, New York,
 654 NY, USA, 1957. Association for Computing Machinery. ISBN 9781450378611. doi: 10.1145/
 655 1455567.1455605. URL <https://doi.org/10.1145/1455567.1455605>.

656 Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations:
 657 Theory and application to reward shaping. In *Proceedings of the Sixteenth International Confer-
 658 ence on Machine Learning*, ICML '99, pp. 278287, San Francisco, CA, USA, 1999. Morgan
 659 Kaufmann Publishers Inc. ISBN 1558606122.

660 Tobias Nipkow, Markus Wenzel, and Lawrence C. Paulson. *Isabelle/HOL: a proof assistant for
 661 higher-order logic*. Springer-Verlag, Berlin, Heidelberg, 2002. ISBN 3540433767.

662

663 OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden
 664 Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko,
 665 Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally
 666 Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich,
 667 Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghor-
 668 bani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao,
 669 Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary
 670 Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang,
 671 Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel
 672 Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson,
 673 Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Eliz-
 674 abeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang,
 675 Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred
 676 von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace
 677 Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin,
 678 Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian
 679 O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever,
 680 Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng,
 681 Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish,
 682 Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan
 683 Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singh, Karina Nguyen, Karl
 684 Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu,
 685 Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam
 686 Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kon-
 687 draciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen,
 688 Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet
 689 Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael
 690 Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles
 691 Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil
 692 Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg
 693 Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov,
 694 Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar
 695 Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan
 696 Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agar-
 697 wal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu,
 698 Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph
 699 Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Tay-
 700 lor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson,
 701 Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna
 702 Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi
 703 Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen,
 704 Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li.
 705 Openai o1 system card, 2024. URL <https://arxiv.org/abs/2412.16720>.

702 Azim Ospanov, Farzan Farnia, and Roozbeh Yousefzadeh. Apollo: Automated llm and lean col-
 703 laboration for advanced formal reasoning, 2025. URL <https://arxiv.org/abs/2505.05758>.

704

705 Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong
 706 Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kel-
 707 ton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike,
 708 and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.
 709 URL <https://arxiv.org/abs/2203.02155>.

710

711 Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving,
 712 2020. URL <https://arxiv.org/abs/2009.03393>.

713

714 Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanja Zhao, Liyue
 715 Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang,
 716 Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. Deepseek-prover-v2: Advancing for-
 717 mal mathematical reasoning via reinforcement learning for subgoal decomposition, 2025. URL
 718 <https://arxiv.org/abs/2504.21801>.

719

720 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy
 721 optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.

722

723 Amrit Sethur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal,
 724 Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated
 725 process verifiers for llm reasoning, 2024. URL <https://arxiv.org/abs/2410.08146>.

726

727 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
 728 Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathe-
 729 matical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.

730

731 Trieu Trinh, Yuhuai Tony Wu, Quoc Le, He He, and Thang Luong. Solving olympiad geometry
 732 without human demonstrations. *Nature*, 625:476–482, 2024. URL <https://www.nature.com/articles/s41586-023-06747-5>.

733

734 Haiming Wang, Ye Yuan, Zhengying Liu, Jianhao Shen, Yichun Yin, Jing Xiong, Enze Xie, Han Shi,
 735 Yujun Li, Lin Li, Jian Yin, Zhenguo Li, and Xiaodan Liang. DT-solver: Automated theorem prov-
 736 ing with dynamic-tree sampling guided by proof-level value function. In Anna Rogers, Jordan
 737 Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Associa-
 738 tion for Computational Linguistics (Volume 1: Long Papers)*, pp. 12632–12646, Toronto, Canada,
 July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.706. URL
 739 <https://aclanthology.org/2023.acl-long.706/>.

740

741 Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood
 742 Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey,
 743 Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu,
 744 Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani,
 745 Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Baroyer, Wen-Ding
 746 Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying
 747 Liu, and Jia Li. Kimina-prover preview: Towards large formal reasoning models with reinforce-
 748 ment learning, 2025a. URL <https://arxiv.org/abs/2504.11354>.

749

750 Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang
 751 Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024.
 752 URL <https://arxiv.org/abs/2312.08935>.

753

754 Ruida Wang, Yuxin Li, Yi R. Fung, and Tong Zhang. Let’s reason formally: Natural-formal hybrid
 755 reasoning enhances llm’s math capability, 2025b. URL <https://arxiv.org/abs/2505.23703>.

756

757 Ruida Wang, Rui Pan, Yuxin Li, Jipeng Zhang, Yizhen Jia, Shizhe Diao, Renjie Pi, Junjie Hu, and
 758 Tong Zhang. Ma-lot: Multi-agent lean-based long chain-of-thought reasoning enhances formal
 759 theorem proving, 2025c. URL <https://arxiv.org/abs/2503.03205>.

756 Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen,
 757 Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen
 758 Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive
 759 effective reinforcement learning for llm reasoning, 2025d. URL <https://arxiv.org/abs/2506.01939>.

760

761 Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Chris-
 762 tian Szegedy. Autoformalization with large language models, 2022. URL <https://arxiv.org/abs/2205.12615>.

763

764

765 Zijian Wu, Suozhi Huang, Zhejian Zhou, Huaiyuan Ying, Jiayu Wang, Dahua Lin, and Kai Chen.
 766 Internlm2.5-stepprover: Advancing automated theorem proving via expert iteration on large-scale
 767 lean problems, 2024. URL <https://arxiv.org/abs/2410.15700>.

768

769 Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li,
 770 and Xiaodan Liang. Deepseek-prover: Advancing theorem proving in llms through large-scale
 771 synthetic data, 2024a. URL <https://arxiv.org/abs/2405.14333>.

772

773 Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu,
 774 Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F.
 775 Wu, Fuli Luo, and Chong Ruan. Deepseek-prover-v1.5: Harnessing proof assistant feedback
 776 for reinforcement learning and monte-carlo tree search, 2024b. URL <https://arxiv.org/abs/2408.08152>.

777

778 Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. Lean workbook:
 779 A large-scale lean problem set formalized from natural language math problems, 2024a. URL
 780 <https://arxiv.org/abs/2406.03847>.

781

782 Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma,
 783 Jiawei Hong, Kuikun Liu, Ziyi Wang, Yudong Wang, Zijian Wu, Shuaibin Li, Fengzhe Zhou,
 784 Hongwei Liu, Songyang Zhang, Wenwei Zhang, Hang Yan, Xipeng Qiu, Jiayu Wang, Kai Chen,
 785 and Dahua Lin. Internlm-math: Open math large language models toward verifiable reasoning,
 786 2024b. URL <https://arxiv.org/abs/2402.06332>.

787

788 Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian
 789 Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng,
 790 Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen,
 791 Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu,
 792 Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An
 793 open-source llm reinforcement learning system at scale, 2025. URL <https://arxiv.org/abs/2503.14476>.

794

795 Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan
 796 Liu, and Hao Peng. Free process rewards without process labels, 2024. URL <https://arxiv.org/abs/2412.01981>.

797

798 Jingyuan Zhang, Qi Wang, Xingguang Ji, Yahui Liu, Yang Yue, Fuzheng Zhang, Di Zhang, Guorui
 799 Zhou, and Kun Gai. Leanabell-prover: Posttraining scaling in formal reasoning, 2025. URL
 800 <https://arxiv.org/abs/2504.06122>.

801

802 Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for
 803 formal olympiad-level mathematics, 2022. URL <https://arxiv.org/abs/2109.00110>.

804

805 Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin,
 806 Qin Liu, Yuhao Zhou, Limao Xiong, Lu Chen, Zhiheng Xi, Nuo Xu, Wenbin Lai, Minghao Zhu,
 807 Cheng Chang, Zhangyue Yin, Rongxiang Weng, Wensen Cheng, Haoran Huang, Tianxiang Sun,
 808 Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. Secrets of rlhf in large language
 809 models part i: Ppo, 2023. URL <https://arxiv.org/abs/2307.04964>.

810

811 Thomas Zhu, Joshua Clune, Jeremy Avigad, Albert Qiaochu Jiang, and Sean Welleck. Premise
 812 selection for a lean hammer, 2025. URL <https://arxiv.org/abs/2506.07477>.

APPENDIX

A ADDITIONAL RELATED WORKS

Automatic Theorem Proving An automated theorem prover typically consists of two stages. The first is the process of translating mathematical statements written in natural language into formal statements. Wu et al. (2022) utilized large language models to translate mathematical questions into formal languages such as Isabelle and HOL. This process, known as autoformalization, is primarily used for constructing datasets intended for formal reasoning. Benchmarks or training datasets such as MiniF2F, LeanWorkbook, ProofNet, Deepseek-Prover have employed LLMs to translate natural language mathematical statements into formal expressions, contributing to the creation of high-quality formal reasoning datasets (Zheng et al., 2022; Azerbayev et al., 2023; Xin et al., 2024a; Ying et al., 2024a).

The second stage involves generating a formal proof from the translated formal statement. This proof generation process is typically divided into two approaches: one involves step-by-step inference, such as tree search during inference time (Polu & Sutskever, 2020; Azerbayev et al., 2024; Wu et al., 2024; Xin et al., 2024b), and the other generates the entire proof at once (Xin et al., 2024a; Lin et al., 2025b). Ospanov et al. (2025); Wang et al. (2025c); Baba et al. (2025) use Lean compiler as agent for complementing formal reasoning ability of LLMs, while (Dong & Ma, 2025) enhances formal reasoning by augmenting problems via conjecture. (Jiang et al., 2023) presents a unified framework that combines both autoformalization and proof generation in a single pipeline.

Existing methods such as Lean-STaR and RMaxTS (Lin et al., 2025a; Xin et al., 2024b) utilize Lean as a step-checker during inference, generating steps sequentially and searching optimally via tree search to find valid proofs. In contrast, in this paper, similar to (Wang et al., 2025a; Zhang et al., 2025; Ren et al., 2025; Ji et al., 2025), we utilize Lean as a whole-proof verifier during the training stage. Additionally, beyond merely providing correctness checks for the entire proof, we leverage Lean’s parsing and elaboration capabilities to validate each individual tactic step, integrating this step-level validation into the training process. In other words, we employ the Lean proof assistant as a process-based reward model for validating the correctness of each reasoning steps. (Lightman et al., 2023).

Unlike prior work that leverages dense feedback from proof assistants, our research takes a different perspective: we rely solely on the rule-based signals of the symbolic engine, without introducing any natural language. Approaches such as Lin et al. (2025a;c); Wang et al. (2025b); Li et al. (2025), exploit natural language reasoning as a form of annotation to enhance LLMs formal reasoning abilities. In contrast, our method improves performance exclusively through reward signals provided by Lean, without any reliance on natural language.

Reinforcement Learning in Language Models While developing or applying algorithms such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) plays a significant role in reinforcement learning, reward shaping and credit assignment are central challenges in reinforcement learning. (Cobbe et al., 2021) introduced a reward model based on the outcome of a response. However, similar to other areas of RLHF, this approach suffers from the limitation of sparse rewards (Chan et al., 2024; Zheng et al., 2023).

To address this, process-based reward model (PRM) assigns step-level rewards during inference to guide rationale generation (Lightman et al., 2023), and can also be used to reward responses during training (Setlur et al., 2024; Kazemnejad et al., 2024). (Yuan et al., 2024; Cui et al., 2025) derived an implicit PRM from the ORM without any data annotation or additional training. When assigning scores to reasoning steps, (Lightman et al., 2023) defined the reward as the correctness of each step, which required substantial human annotation effort. (Wang et al., 2024) instead adopted a Monte Carlo approach, defining the score of a step as the proportion of successful rollouts originating from that step. While recent PRM approaches show promise in natural language reasoning, they require large annotated datasets of step-level correctness. To the best of our knowledge, no such dataset exists for Lean or formal theorem proving, making a direct comparison with a learned PRM baseline infeasible. This further motivates our approach of leveraging the Lean verifier itself as a process oracle. In contrast, our process-based reward leverages the Lean theorem prover to automatically

864 verify the correctness of each step, thereby eliminating the need for human annotators or sampling
 865 many proofs steps.
 866

867 Our work can also be interpreted through the lens of reward shaping (Ng et al., 1999). Prior ap-
 868 proaches have explored different mechanisms for distributing reward signals: Chan et al. (2024)
 869 leverages the internal attention patterns of LLMs to assign higher weights to important tokens, Cao
 870 et al. (2025) employs Shapley values to allocate credit across actions, and Kazemnejad et al. (2024)
 871 uses Monte Carlo rollouts to estimate and distribute rewards over intermediate steps. In contrast,
 872 our method relies on an external parser-the Lean theorem prover-to parse tactics and assign reward
 873 to the first token, thereby implementing a form of credit assignment.
 874

875 B EXPERIMENTAL DETAIL

876 **Data.** We randomly sampled 10k instances from the STP dataset (3.26M total) for RL training.
 877 For DeepSeek-Prover-V1.5-SFT, we applied an additional supervised fine-tuning step on 500k STP
 878 samples before RL, since the vanilla model produced low-quality proofs during RL training.
 879

880 **Verification.** We use Lean 4.9.0-rc1 for all experiments in the paper. During training, we used
 881 a REPL (read-eval-print loop) interface with Lean to verify proofs and assign outcome- and tactic-
 882 level rewards. Each proof attempt was given a maximum of 15 seconds for verification; longer runs
 883 were treated as failures (both outcomes, tactic rewards are zero).
 884

885 **RL configuration.** For GRPO training, we used $G = 4$ generations per prompt, sampling temper-
 886 ature 0.9, KL coefficient 0.04, clipping $\epsilon = 0.2$, and the DAPO upper bound 0.28 (Yu et al., 2025).
 887 Tactic-level rewards were fixed at $d_1 = -0.05$ and $d_2 = -0.1$ for partially valid and erroneous
 888 tactics in the main experiments, respectively. All experiments used non-CoT prompts, following
 889 Xin et al. (2024b).
 890

891 **Training details.** We fine-tuned the models with LoRA (rank 64, $\alpha = 64$) using bf16 precision.
 892 The AdamW optimizer was used with a learning rate of 1.0×10^{-5} . Maximum response length was
 893 set to 1024 tokens during both training and evaluation.
 894

895 **Evaluation.** For decoding we used temperature 1.0 and top-p 0.95. We re-evaluated all baselines
 896 under the same non-CoT and budget settings (32/64 samples). All reported results are from the final
 897 checkpoint.
 898

899 **Compute.** Training was conducted on $4 \times$ NVIDIA A6000 GPUs, requiring approximately 21-23
 900 hours.
 901

902 C HYPERPARAMETER ABLATIONS ON d_i

905 Setting	906 Model Size	907 Sample Budget	908 MiniF2F - Test	909 ProofNet - Test
910 STP-baseline	911 7B	912 32	913 $55.9\% \pm 0.2$	914 $17.2\% \pm 0$
		915 64	916 $56.7\% \pm 0.2$	917 $19.1\% \pm 0.4$
918 GRPO baseline	919 7B	920 32	921 $55.7\% \pm 1$	922 $17.4\% \pm 0.6$
		923 64	924 $57.9\% \pm 0.5$	925 $19\% \pm 0.3$
926 $d_1 = -0.05, d_2 = -0.10$	927 7B	928 32	929 $57.1\% \pm 0.8$	930 $18.6\% \pm 0.3$
		931 64	932 $59.2\% \pm 0.5$	933 $19\% \pm 0.3$
934 $d_1 = d_2 = -0.10$	935 7B	936 32	937 $57.7\% \pm 0.2$	938 $17.6\% \pm 0.6$
		939 64	940 $58.7\% \pm 0.8$	941 $18.1\% \pm 0.6$
942 $d_1 = -0.05, d_2 = -0.50$	943 7B	944 32	945 $57\% \pm 0.4$	946 $17.6\% \pm 0.3$
		947 64	948 $59.2\% \pm 0.5$	949 $18.6\% \pm 0.8$

950 Table 5: Ablation study on tactic-level penalties d_1, d_2 . We compare outcome-only GRPO base-
 951 line with three variants of (d_1, d_2) settings. Results are reported as pass@32 and pass@64 (%) on
 952 MiniF2F and ProofNet test sets. The experiment is conducted with STP-Lean model.
 953

918 This ablation shows that introducing a gap between d_1 and d_2 makes the method more robust: per-
 919 formance remains consistently above the GRPO baseline, with stable gains across different penalty
 920 scales and especially clear improvements on MiniF2F.
 921

922 D PROMPTS

924 For training and evaluation, we used non-COT evaluation followed by (Dong & Ma, 2025) and (Xin
 925 et al., 2024b). The examples are introduced in Table 6, 7.
 926

927 **Prompt Template**
 928
 929 Complete the following Lean 4 code:\n\n
 930 ```lean4\n{ header }\n{ formal_statement }
 931
 932
 933

934 Table 6: Prompt template used in training and evaluation. We selected non-COT generation which
 935 is appropriate with our MDP setting
 936

937 **Prompt Example**
 938
 939 Complete the following Lean 4 code:
 940
 941 ```lean4
 942 import Mathlib
 943 import Aesop
 944 set_option maxHeartbeats 0
 945 open BigOperators Real Nat Topology Rat
 946
 947 theorem theorem_exercise_2011_2_257 (G : Type*) [Group G] [Fintype
 948 ↪ G]
 949 (h : Fintype.card G = 2) (x : G) : x ^ 2 = 1
 950 (x y : G, x * y = y * x) (a : G, a = 1) a : G, a^2 = 1
 951 let p x : G G := by

951 Table 7: A training sample used in training and evaluation. We selected non-COT generation which
 952 is appropriate with our MDP setting.
 953

954 E CREDIT ASSIGNMENT IN REINFORCEMENT LEARNING

957 Let y_t be the t -th token of y , R denote the reward model, π_θ represent the policy model, and
 958 π_{ref} be the reference model. L denote the response length and B be a coefficient controlling
 959 the distance between the policy and the reference policy. In PPO, the token-level reward at
 960 position t is defined as: $r_t(x, y_t) = R(x, y) \mathbf{1}(y_t = L) - B \log \left(\frac{\pi_\theta(y_t | x)}{\pi_{\text{ref}}(y_t | x)} \right)$, where the non-zero
 961 reward $R(x, y)$ is assigned only to the last token. For all other tokens, only a KL divergence
 962 penalty is applied via a log ratio $\log \left(\frac{\pi_\theta(y_t | x)}{\pi_{\text{ref}}(y_t | x)} \right)$. Direct usage of rewards can lead to high var-
 963 iance; therefore, PPO reduces variance by utilizing a learned value model V . This value network
 964 assigns a value to each token y_t , from which the Temporal-Difference (TD) error is computed as:
 965 $\delta_t = r_t + \gamma V(y_{t+1}) - V(y_t)$ where γ is discounted factor. Then, the advantage for each token is
 966 recursively calculated as follows: $A_L = \delta_L$, $A_t = \delta_t + \gamma \lambda A_{t+1}$, for $t = L-1, L-2, \dots, 1$.
 967 Subsequently, because the computed advantages A_t can exhibit high variance during exploration,
 968 normalization or similar techniques are applied, resulting in the final adjusted advantage A_t . This
 969 adjusted advantage is then utilized in the PPO loss defined as:
 970

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(\frac{\pi_\theta(y_t | x)}{\pi_{\theta_{\text{old}}}(y_t | x)} A_t, \text{clip} \left(\frac{\pi_\theta(y_t | x)}{\pi_{\theta_{\text{old}}}(y_t | x)}, 1 - \epsilon, 1 + \epsilon \right) A_t \right) \right] \quad (2)$$

972 In contrast, REINFORCE-based methods such as GRPO and RLOO have proposed algorithms that
 973 optimize policies directly from verifiable rewards without requiring a value model, due to concerns
 974 about the computational cost and estimation capability associated with training value networks.

975 GRPO generates multiple response groups $\{y_{(i)}\}_{i=1}^G$ for a given question q from an old policy π_{old} .
 976 Subsequently, a reward function outputs reward $r = \{r_{(i)}\}_{i=1}^G$ for each response group. If we set
 977 $y_{i,t}$ as t -th token index of response y_i . The advantage for $y_{i,t}$, $A_{i,t}$ is then computed by normalizing
 978 these rewards as follows: $\hat{A}_{i,t} = \frac{r_i - \text{mean}(r)}{\text{std}(r)}$.

980 This advantage is uniformly assigned to each token $y_{i,t}$ constituting the response y_i . Subsequently,
 981 this identical token-level advantage is utilized in calculating the following loss:

$$983 L_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q), \{y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(Y|q)} \\ 984 \left[\frac{1}{G} \sum_{i=1}^G \left\{ \min \left(\frac{\pi_{\theta}(y_{i,t} | q)}{\pi_{\theta_{\text{old}}}(y_{i,t} | q)} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_{\theta}(y_{i,t} | q)}{\pi_{\theta_{\text{old}}}(y_{i,t} | q)}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right) - \beta D_{\text{KL}}[\pi_{\theta} \| \pi_{\text{ref}}] \right\} \right]. \quad (3)$$

987 As our goal is to integrate Lean-based fine-grained rewards to GRPO algorithm, our method keep
 988 balance between value-based PPO's fine-grained credit assignmethod and GRPO's simple unifrom
 989 credit assignment approach.

990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1020
 1021
 1022
 1023
 1024
 1025

1026 F GENERATED PROOFS EXAMPLES
1027

1028

1029 imo_1960_p2 in minif2f from tactic-reward-trained model
1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

```

import Mathlib
import Aesop

set_option maxHeartbeats 0

open BigOperators Real Nat Topology Rat

theorem imo_1960_p2 (x : \mathbb{R}) (h_0 : 0 ≤ 1 + 2 * x)
  (h_1 : (1 - Real.sqrt (1 + 2 * x)) ^ 2 ≥ 0)
  (h_2 : 4 * x ^ 2 / (1 - Real.sqrt (1 + 2 * x)) ^ 2 ≤ 2 * x + 9) :
  -(1 / 2) * x ≤ 45 / 8 := by
  have h_3 : 0 ≤ (1 + 2 * x) := by linarith
  have h_4 : 0 ≤ Real.sqrt (1 + 2 * x) := Real.sqrt_nonneg(1 + 2 * x)
  have h_5 : 0 ≤ (1 + 2 * x) * Real.sqrt (1 + 2 * x) := by positivity
  have h_6 : 0 ≤ (1 - Real.sqrt (1 + 2 * x)) ^ 2 := by
    apply sq_pos_of_ne_zero
    intro h
    apply h_1
    nlinarith
  constructor
  nlinarith [sq_sqrt (show 0 ≤ 1 + 2 * x from by linarith), h_2]
  have h_7 : x ≤ 45 / 8 := by
    apply lt_of_not_ge
    intro hge
    have h_8 : 4 * x ^ 2 / (1 - Real.sqrt (1 + 2 * x)) ^ 2 ≤ 2 * x + 9
    := by
      rw [ge_iff_le]
      rw [le_div_iff h_6]
      nlinarith [sq_sqrt (show 0 ≤ 1 + 2 * x from by linarith), hge]
    linarith
  linarith

```

Table 8: Proof of imo_1960_p2 from MiniF2F generated by our tactic-reward-trained model.

```

1080
1081 imo_1960_p2 in minif2f from baseline model
1082
1083 import Mathlib
1084 import Aesop
1085 set_option maxHeartbeats 0
1086
1087 open BigOperators Real Nat Topology Rat
1088
1089 theorem imo_1960_p2 (x : \mathbb{R})
1090   (h_0 : 0 ≤ 1 + 2 * x) (h_1 : (1 - Real.sqrt (1 + 2 * x)) ^ 2 ≥ 0)
1091   (h_2 : 4 * x ^ 2 / (1 - Real.sqrt (1 + 2 * x)) ^ 2 ≤ 2 * x + 9) :
1092     -(1 / 2) * x ≤ 45 / 8 := by
1093   have h_3 : 0 < (1 - Real.sqrt (1 + 2 * x)) ^ 2 := by positivity
1094   have h_4 : 0 ≤ Real.sqrt (1 + 2 * x) := Real.sqrt_nonneg _
1095   have h_5 : 0 ≤ 1 + 2 * x := by linarith
1096   have h_6 : Real.sqrt (1 + 2 * x) ≤ 1 + x := by
1097     rw [Real.sqrt_le_iff]
1098     constructor
1099     · linarith
1100     · nlinarith
1101   have h_7 : 4 * x ^ 2 / (1 - Real.sqrt (1 + 2 * x)) ^ 2 ≤ 2 * x + 9
1102     :=
1103   by assumption
1104   have h_8 : -(1 / 2) * x := by
1105     rw [div_lt_iff h_3] at h_7
1106     nlinarith
1107   have h_9 : x ≤ 45 / 8 := by
1108     rw [div_lt_iff h_3] at h_7
1109     nlinarith
1110   exact h_8, h_9
1111 <;> assumption
1112 <;> assumption
1113 <;> assumption
1114 <;> assumption
1115 <;> assumption
1116 <;> assumption
1117 <;> assumption
1118 <;> assumption
1119 <;> assumption
1120 <;> assumption
1121 <;> assumption
1122 <;> assumption
1123 <;> assumption
1124 <;> assumption
1125 <;> assumption
1126 <;> assumption
1127 <;> assumption
1128 <;> assumption
1129 <;> assumption
1130 <;> assumption
1131
1132
1133

```

Table 9: Proof of `imo_1960_p2` from MiniF2F generated by the baseline STP model.

```

1118 exercise_1_14 in ProofNet from tactic-reward-trained model.
1119
1120 import Mathlib
1121
1122 open Topology Filter Real Complex TopologicalSpace Finset
1123 open scoped BigOperators
1124
1125 theorem exercise_1_14
1126   (z : \mathbb{C}) (h : abs z = 1)
1127   : (abs (1 + z)) ^ 2 + (abs (1 - z)) ^ 2 = 4 := by
1128   rw [Complex.norm_sq_eq_abs, Complex.norm_sq_eq_abs]
1129   simp [Complex.norm_sq_add, Complex.norm_sq_sub, h]
1130   ring_nf
1131   <;> simp [h, Complex.norm_sq_eq_abs]
1132   <;> ring_nf
1133

```

Table 10: Proof of `exercise_1_14` from ProofNet generated by our tactic-reward-trained model.

```

1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154 exercise_1_14 in ProofNet from baseline
1155
1156 import Mathlib
1157
1158 open Topology Filter Real Complex TopologicalSpace Finset
1159 open scoped BigOperators
1160
1161 theorem exercise_1_14
1162   (z : \mathbb{C}) (h : abs z = 1)
1163   : (abs (1 + z)) ^ 2 + (abs (1 - z)) ^ 2 = 4 := by
1164   have h_0 : abs z ^ 2 = 1 := by simp [h]
1165   simp [pow_two, abs_add, abs_sub, h_0]
1166   ring_nf
1167   simp_all [sq_abs]

```

Table 11: Proof of `exercise_1_14` from ProofNet generated by the baseline STP model.

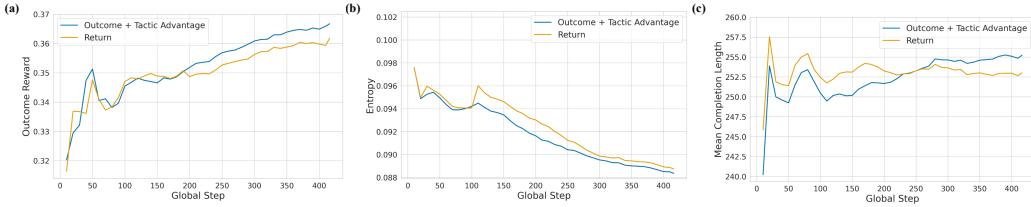
```

1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187

```

1188 **G RESULTS ON RETURN-BASED ADVANTAGE**
1189

Model	Model Size	Sample Budget	MiniF2F - Test	ProofNet - Test
Return	7B	32	$55.3\% \pm 0.4$	$18.1\% \pm 0.3$
		64	$57.5\% \pm 0.2$	$18.6\% \pm 0.3$
Outcome+Tactic RL (ours)	7B	32	$57.1\% \pm 0.8$	$18.6\% \pm 0.3$
		64	$59.2\% \pm 0.5$	$19\% \pm 0.3$

1196 Table 12: Ablation study on reward strategies for return-based advantage.
11971207 Figure 4: Training dynamics showing (a) outcome reward,(b) entropy, and (c) mean of response
1208 length during reinforcement learning in our method and return-based method.
1209

H FAILURE CASE AND LIMITATION OF OUR METHOD

```

1245 imo_2019_p1 in MiniF2f from tactic-reward-trained model.
1246
1247 import Mathlib
1248 import Aesop
1249 set_option maxHeartbeats 0
1250
1251 open BigOperators Real Nat Topology Rat
1252
1253 theorem imo_2019_p1 (f : \Z \Z) :
1254   (\forall a b, f (2 * a) + 2 * f b = f (f (a + b)))
1255   \n \iff (\forall z, f z = 0)
1256   \n (\exists c, \forall z, f z = 2 * z + c) := by
1257   constructor
1258   intro h
1259   intro z
1260   have h_1 := h 0 0
1261   have h_2 := h 0 1
1262   have h_3 := h 1 0
1263   have h_4 := h 1 1
1264   have h_5 := h 2 0
1265   have h_6 := h 2 1
1266   have h_7 := h 0 2
1267   have h_8 := h 1 2
1268   have h_9 := h 2 2
1269   simp at h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8 h_9
1270   norm_num at h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8 h_9
1271   omega
1272   intro h
1273   intro a b
1274   have h_1 := h (2 * a)
1275   have h_2 := h b
1276   have h_3 := h (a + b)
1277   cases' h_1 with h_1 h_1 < ; > cases' h_2 with h_2 h_2
1278   \n < ; > cases' h_3 with h_3 h_3 < ; > simp_all
1279   < ; > omega

```

Table 13: Proof of `imo_2019_p1` in MiniF2f generated by our tactic-reward-trained model.

Consider a function $f : \mathbb{Z} \rightarrow \mathbb{Z}$ satisfying

$$\forall a, b \in \mathbb{Z}, \quad \quad f(2a) + 2 f(b) = f(f(a+b)).$$

The task is to prove that necessarily one of the following holds:

(i) $\forall z \in \mathbb{Z}, f(z) = 0$, or
(ii) $\exists c \in \mathbb{Z}, \forall z \in \mathbb{Z}, f(z) = 2z + c$.

Our model first introduced the assumption

$$h : \forall a, b \in \mathbb{Z}, f(2a) + 2 f(b) = f(f(a + b)),$$

and then instantiated it at several concrete pairs to create hypotheses h_i (e.g., $h_1 := h(0, 0)$, $h_2 := h(0, 1)$, \dots). After some local simplification steps (e.g., `simp`, `norm_num`), it attempted to close the goal using the `omega` tactic, a decision procedure for Presburger arithmetic (linear integer arithmetic).

However, the `omega` call produced the first Lean error. While our method correctly assigns the d_2 penalty to this failing `omega` tactic under first-error propagation, it does not penalize the preceding tactics (`intro`, `have`, `simp`) because they elaborate successfully and thus appear locally valid. In other words, although introducing h and instantiating h_i is not logically incorrect, this route is strategically unproductive for this problem: the remaining goal still involves quantifiers, disjunction, and

1296 an uninterpreted function f , which lie outside omega's theory. Consequently, our current scheme
 1297 only punishes the terminal failing step and fails to capture that the earlier (locally successful) steps
 1298 did not make meaningful progress toward solving the global goal.
 1299

1300 I LARGE LANGUAGE MODEL USAGE

1302 In preparing this manuscript, we made limited use of large language models strictly for writing
 1303 assistance. Specifically, we used ChatGPT-5 and Gemini-2.5 to improve grammar, enhance clarity
 1304 of expression, and polish the overall presentation.
 1305

1306 J MATHEMATICAL GROUND FOR TACTIC REWARD

1308 In this section, we provide a conceptual interpretation of our tactic-level rewards using a simple value
 1309 model and potential-based reward shaping. Our goal is not to claim a formal optimality guarantee,
 1310 but rather to clarify how the structure of our discrete Lean-based rewards is aligned with an idealized
 1311 concept of proof success under a First error propagation assumption with potential function.
 1312

1313 J.1 DEFINE VALUE FUNCTION

1315 Consider a Lean proof trajectory

$$1316 \quad s_0 \xrightarrow{T_1} s_1 \xrightarrow{T_2} \dots \xrightarrow{T_N} s_N,$$

1318 where s_t denotes the prefix of tactics (T_1, \dots, T_{t-1}) , and s_N refers to a completed proof.
 1319

1320 We adopt the modeling assumption already used in the main paper:

1321 From the first error propagation, once the first erroneous tactic occurs, the proof can no longer be
 1322 repaired into a valid Lean proof.

1323 Formally, let j be the index of the first tactic for which Lean reports an error. Then all states s_t with
 1324 $t \geq j$ lie in an absorbing failed state. Instead of assuming independent Bernoulli errors, we consider
 1325 a more general and realistic model with conditional valid probabilities. For a valid prefix s_{k-1} , let

$$1327 \quad q(s_{k-1}) = P(\text{no error at step } k \mid s_{k-1} \text{ is valid}).$$

1328 Under the first error propagation assumption, we define the value function as the probability of
 1329 eventually producing a valid proof from a valid prefix s_t is

$$1331 \quad V(s_t) = P(\text{success} \mid s_t) = \prod_{k>t} q(s_{k-1}).$$

1333 Along a valid trajectory, we have

$$1335 \quad V(s_{t+1}) = \frac{V(s_t)}{q(s_t)} \geq V(s_t),$$

1337 so $V(s_t)$ is monotone increasing until no errors are founded. if the first error occurs at step j , the
 1338 success probability collapses to zero:

$$1339 \quad V(s_j) = V(s_{j+1}) = \dots = 0.$$

1341 Qualitatively, this yields the following structure:

- 1343 • For a successful proof, $V(s_t)$ increases from a small value at s_0 to $V(s_N) = 1$.
- 1344 • For a failed proof, $V(s_t)$ increases along the correct prefix, and then drops to 0 at the first
 1345 error and stays at 0 afterwards.

1346 Thus, the ideal value function encodes (i) Monotone growth along error-free prefixes and (ii) Irre-
 1347 versible collapse after the first error.
 1348

1349 This structure motivates using stronger positive credit for tactics on an error-free prefix and negative
 or neutral credit after the first failure.

1350 J.2 POTENTIAL-BASED REWARD SHAPING WITH $V(s)$
1351

1352 The value defined in previous section suggests a way to define potential-based shaping. In an ide-
1353 alized MDP setting where the environment state is exactly s_t and the agent has access to the value
1354 function $V(s)$, one could define a potential

$$1356 \quad \Phi^*(s) = f(V(s)),$$

1357 where f is any monotonically increasing transformation (e.g., $f(v) = v$). A shaped reward can then
1358 be written as

$$1360 \quad r_t^* = r_{\text{outcome},t} + \gamma \Phi^*(s_{t+1}) - \Phi^*(s_t),$$

1362 where $r_{\text{outcome},t}$ is the sparse end-of-proof reward derived from $g(Y)$. Under the assumptions of (Ng
1363 et al., 1999), such potential-based shaping preserves the set of optimal policies.

1364 Intuitively, using $\Phi^*(s) = f(V(s))$ means that the potential is highest on globally successful trajec-
1365 tories, increases along error-free prefixes, and collapses after the first error, similar with the structure
1366 of $V(s)$ in previous section. The corresponding temporal-difference term

$$1368 \quad \gamma \Phi^*(s_{t+1}) - \Phi^*(s_t)$$

1369 acts as a local improvement signal: it is positive along valid contexts, negative when the value
1370 collapses at the first error, and zero afterwards.

1372 In our setting, however, we neither assume access to the true $V(s)$. We therefore view this potential-
1373 based construction as a *normative model* that suggests the qualitative shape of a desirable local credit
1374 signal, rather than as a source of formal optimality guarantees.

1375 J.3 LEAN-BASED DISCRETE APPROXIMATION AS QUANTIZED LOCAL SHAPING
1376

1377 In practice, we did not estimate $V(s)$ or $\Phi^*(s)$ explicitly. Instead, we exploit Lean’s symbolic
1378 feedback (AST errors, first-error propagation) to construct discrete tactic-level scores.

1380 For a proof Y , with first error index j (if any), recall that we define

$$1381 \quad \varphi(Y, T_t) = \begin{cases} 1, & \text{if } g(Y) = 1, \\ 1382 \quad d_1, & \text{if } g(Y) = 0 \text{ and } t < j, \\ 1383 \quad d_2, & \text{if } g(Y) = 0 \text{ and } t \geq j, \end{cases} \quad 1 > d_1 > d_2.$$

1385 as the process-level reward for tactic T_t .

1386 Conceptually, $\varphi(Y, T_t)$ is a coarse, Lean-driven *quantization* of the ideal local improvement signal
1387 suggested by the value model. States on globally successful trajectories receive the highest score
1388 (1); states on error-free prefixes of failed proofs receive an intermediate score (d_1); and states at or
1389 after the first error receive the lowest score (d_2). This partitions trajectories into three regions whose
1390 ordering (success > pre-error > post-error) is aligned with the ordering of $V(s)$ implied by previous
1391 section.

1392 Assuming $\gamma = 1$ and a finite horizon, any such per-step process reward sequence can be written as
1393 a difference of a state potential. For a fixed trajectory Y of length N , define Φ backwards by

$$1395 \quad \Phi(s_N) = 0, \quad \Phi(s_{t+1}) - \Phi(s_t) = r_{\text{process},t}, \quad t = N-1, \dots, 0.$$

1397 By construction,

1398 so the total shaped reward becomes

$$1400 \quad r'_t = r_{\text{outcome},t} + r_{\text{process},t} = r_{\text{outcome},t} + \Phi(s_{t+1}) - \Phi(s_t).$$

1401 This potential Φ is not intended as an estimate of the true value function $V(s)$; rather, it is an implicit
1402 potential induced by our discrete Lean-based scoring rule. The key point is that its level sets respect
1403 the same qualitative ordering (success > pre-error > post-error) as the ideal value model, providing
a theoretically motivated yet practical shaping signal.

1404
1405 J.4 DISCUSSION AND LIMITATIONS
14061407 Our analysis suggests the following:
14081409

- Under a first-error propagation assumption, an ideal value function $V(s)$ for Lean proofs
1410 increases along error-free prefixes and collapses to zero after the first error.
- Our discrete tactic-level scores $\varphi(Y, T_t) \in 1, d_1, d_2$ can be viewed as a quantized local
1411 improvement signal, capturing this qualitative structure without estimating $V(s)$ explicitly.
- For any given trajectory, the resulting process rewards $r_{\text{process},t}$ can be written as a
1413 potential-based shaping term $r_{\text{process},t} = \Phi(s_{t+1}) - \Phi(s_t)$ for a suitable potential Φ .

14141415 Consequently, our use of potential-based shaping should be understood as a theoretical framework
1416 that explains the structure of our rewards and motivates our design choices, rather than as a strict
1417 proof that our procedure preserves the optimal policy for the original sparse outcome reward. Empir-
1418 ically, we observe that this verifier-informed, discretized shaping leads to more stable training and
1419 consistent improvements over outcome-only GRPO on MiniF2F and ProofNet. We emphasize that
1420 we do not claim any formal optimality guarantee for this shaped reward in our large-scale LLM and
1421 Lean setting; the potential-based perspective is used purely as a conceptual framework for designing
1422 and interpreting our tactic-level rewards.
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457