INDUCING PROCESS SUPERVISION FROM OUTCOME-ONLY REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Process reward models (PRMs) have become a key component for LLMs, as their step-level feedback supports both post-training and test-time reasoning. However, training strong PRMs remains costly: human step annotation is difficult to scale, while Monte Carlo estimation is computationally expensive and can drift from the intrinsic correctness of steps. To get effective PRMs at low cost, we introduce TIPS (**T**hinking-**I**nduced **P**rocess **S**upervision), an outcome-only reinforcement learning (RL) framework for training generative PRMs. In TIPS, the model generates a chain-of-thought (CoT) followed by step-level labels and an outcome label. The reward depends solely on whether the predicted outcome matches the ground truth, and the resulting group-relative advantage is used to optimize the entire generated response. Intuitively, when checking intermediate steps helps determine the outcome, more accurate checks can lead to better outcome judgments and higher rewards. Outcome-only RL can therefore reinforce step-level verification without explicit process supervision. We validate the effectiveness of TIPS across math and agent benchmarks and four backbone families. Notably, TIPS-Qwen3-4B-Thinking-2507 reaches **85.2 F1** on ProcessBench with only 3.2K outcome-labeled trajectories, surpassing all evaluated trained PRMs and strong prompt-only judges such as GPT-5.4-Instruct and Claude-4.7-Opus, while still trailing o1-mini. Code and data are available at `https://anonymous.4open.science/r/TIPS-01B5`.

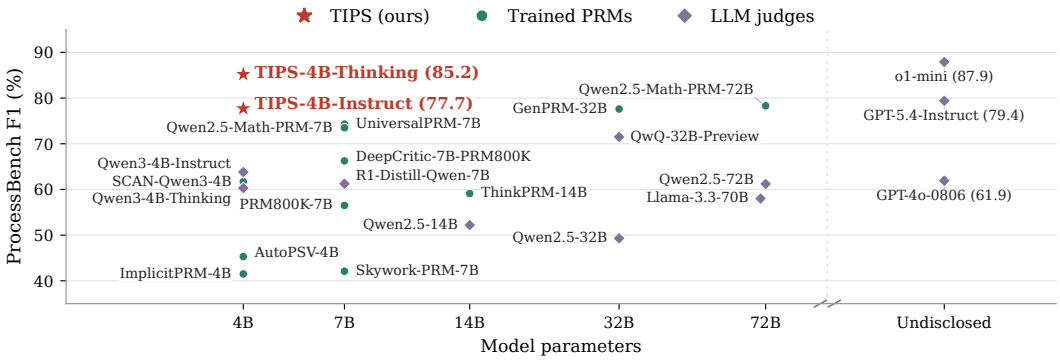


Figure 1: **ProcessBench performance across model scales.** TIPS achieves strong performance using a 4B-parameter model trained on only 3.2K publicly available, outcome-labeled reasoning traces.

1 INTRODUCTION

Process reward models (PRMs), which provide step-level feedback for reasoning traces and agent trajectories, have become a central component of test-time scaling and fine-grained credit assignment in large language models (LLMs) (Lightman et al., 2023; Wang et al., 2024). Despite their importance, building strong PRMs remains difficult. Human step annotation provides reliable supervision but is expensive and hard to scale (Lightman et al., 2023). The prevailing Monte Carlo bassed methods (Ding et al., 2026; Wang et al., 2024) reduce human cost, but require massive rollouts and defines step quality through the existence of a correct continuation. However, the possibility of reaching a correct

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

Problem. What is $0^{(5^{6431564})}$? **Candidate solution.** P_0: $5^{6431564}$ is a huge positive integer. P_1: "any positive integer raised to power 0 is 1," so $0^{(5^{6431564})} = 1$. P_2: $\boxed{1}$.

Thought of Qwen3-4B-Instruct-2507.
Paragraph 0: It correctly identifies that $5^{6431564}$ is a large positive integer. This is valid and provides context. No error here. → Label: 1
Paragraph 1: It says "any positive integer raised to the power of 0 is equal to 1" and applies that to $0^{(5^{6431564})}$. This is a critical error. The rule [...] is correct, but the expression here is **0 raised to a large positive power**, not a positive number raised to the 0th power. So, the logic is backwards. [...] → This paragraph is invalid. → Label: 0
Paragraph 2: It says the final answer is $\boxed{1}$, which is based on the incorrect reasoning in paragraph 1. Since paragraph 1 is wrong, paragraph 2 is also wrong. → Label: 0

Figure 2: Process verification emerges in generated thoughts.

answer from a step does not necessarily imply that the step itself is correct (Zhang et al., 2025a). Therefore, developing strong PRMs at low cost remains an open challenge.

Our starting point is that outcome verification can benefit from process verification. When prompted as an outcome reward model (ORM), an LLM can inspect intermediate steps in its chain of thought (CoT) before judging the final answer. For example, Figure 2 shows that Qwen3-4B identifies an erroneous step in its thought and consequently judges the final answer to be incorrect. This observation suggests that accurate step-level verification can support more reliable outcome judgments. It therefore raises a natural question: **can improving outcome verification, in turn, strengthen process verification?**

To investigate this question, we introduce **TIPS** (**T**hinking-**I**nduced **P**rocess **S**upervision), an outcome-only reinforcement learning framework for strengthening process verification through the model's generated thought. During training, as shown in Figure 3, TIPS prompts an LLM to generate a CoT followed by step-level labels and an outcome label. The reward depends solely on the correctness of the outcome prediction. Under GRPO (Shao et al., 2024), the same outcome-derived advantage is applied to every token in the generated response, including those in the CoT, step labels, and outcome label. Thus, outcome-level feedback shapes both the generated reasoning and the process judgments without direct process supervision. Intuitively, rollouts that accurately evaluate step validity within their thinking process are more likely to predict the correct outcome, thereby earning higher relative advantages. Consequently, **rewarding correct outcome judgments can strengthen step-level verification in the model's thoughts**, even without process supervision. Our theoretical analysis provides support for this intuition. When step validity determines the outcome and accurate outcome prediction requires the thought, that thought must contain information about step correctness. We validate TIPS on both mathematical reasoning and agent tasks across four backbone families: Qwen (Yang et al., 2025; 2024), DeepSeek (DeepSeek-AI, 2025), LlaMa (Grattafiori et al., 2024; Wang et al., 2025b), and SmolLM (Bakouch et al., 2025). The extensive experimental results demonstrate the effectiveness of TIPS. Notably, on widely used ProcessBench (Zheng et al., 2025b), TIPS-Qwen3-4B-Thinking-2507 achieves **85.2 F1** using only 3.2K outcome-labeled trajectories, outperforming substantially larger trained PRMs, as well as strong prompt-only proprietary judges.

In summary, our main contributions are:

- We introduce TIPS, an outcome-only RL framework that computes rewards based solely on the correctness of outcome predictions and strengthens step-level verification as a byproduct.
- We provide an information-theoretic analysis of TIPS, deriving a lower bound on the information about step correctness encoded in the model's thought.
- We validate TIPS on mathematical reasoning and agent tasks across multiple backbones.

2 RELATED WORK

Process Reward Models. PRMs provide step-level supervision for LLM and have been shown to outperform outcome-only verification (Lightman et al., 2023). To avoid costly human step annotations, prior work commonly trains discriminative PRMs to fit automatically constructed process labels, such

as those derived from Monte Carlo estimation (Ding et al., 2026; Wang et al., 2024) or stronger-model distillation (Zhang et al., 2025a; Tan et al., 2026). However, these approaches typically use the LLM only as a representation encoder with a scalar scoring head, leaving its generative reasoning capability underexplored. Among these discriminative approaches, ImplicitPRM (Yuan et al., 2025) and AutoPSV (Lu et al., 2024) are closest to our supervision setting, as they learn process scores from trajectory-level outcome labels; however, their induced signals often fail to reliably localize step-level errors (Table 1). Recent generative PRMs instead prompt models to produce chain-of-thought rationales before making step-level judgments (Zhao et al., 2025; Khalifa et al., 2026), but still rely on explicit process supervision from human annotated steps or stronger judges. In contrast, TIPS uses only trajectory-level outcome labels and induces step-level verification through outcome-only RL.

Reinforcement Learning for LLMs. Since the success of DeepSeek-R1 (DeepSeek-AI, 2025), reinforcement learning has become an increasingly important post-training paradigm for LLMs. Recent studies have applied RL to improve LLMs in mathematical reasoning (Yu et al., 2026; Fan et al., 2026c), tool-using agents (Li et al., 2025; Fan et al., 2026a), retrieval (Jin et al., 2025; Chen et al., 2026a), code generation (Wang et al., 2025a; Team et al., 2025), demonstrating its effectiveness. Beyond policy model learning, a growing line of work explores RL for reward models (Chen et al., 2026b; Whitehouse et al., 2025; Guo et al., 2025), showing that RL can improve ORM accuracy. Our work differs in its objective: rather than using RL only to obtain a stronger ORM, we show that outcome-only RL can induce a strong PRM as a by-product.

3 THINKING-INDUCED PROCESS SUPERVISION

In this section, we present TIPS, a framework that induces process supervision from outcome-only reinforcement learning. We first motivate the key intuition (Section 3.1), then describe the training framework (Section 3.2), and finally provide an information-theoretic analysis (Section 3.3).

3.1 MOTIVATION

Recent work has shown that prompting LLMs to think before acting can improve performance (Wei et al., 2022; Yao et al., 2022). We observe a related phenomenon in reward modeling: when prompted as an ORM, an LLM often uses its thought to inspect intermediate steps before producing the final verdict. As illustrated in Fig. 2, such thoughts already contain useful step-level verification even without process-level training. This suggests a simple way to induce process supervision from outcome-only feedback. If a thought identifies step-level errors more faithfully, the model is more likely to predict the correct trajectory-level outcome; under RL, such thought patterns receive higher reward and are reinforced. **Thus, outcome-only optimization can indirectly strengthen process-aware verification, without requiring explicit step annotations.** Based on this insight, we propose TIPS, a simple framework that trains a generative reward model with outcome-only RL and obtains strong PRM capability as a byproduct.

3.2 METHODOLOGY

As shown in Fig. 3, TIPS trains a generative reward model to produce both outcome-level and step-level judgments, while using only outcome-level ground truth for supervision. Given a problem q and a candidate response $\mathbf{x} = (x_1, \ldots, x_n)$ with n intermediate steps, we prompt the model to first generate a thinking chain $Z \sim \pi_\theta(\cdot \mid q, \mathbf{x})$, where π_θ denotes the generative reward model. Conditioned on this thought, the model then outputs two types of judgments: a sequence of process labels $\hat{\boldsymbol{Y}} = (\hat{Y}_1, \ldots, \hat{Y}_n)$, where $\hat{Y}_t \in \{0, 1\}$ indicates whether step x_t is valid given the preceding steps, and an outcome label $\hat{D} \in \{0, 1\}$ predicting whether the full trajectory is correct. Let $D \in \{0, 1\}$ denote the ground-truth outcome label. We define the reward using only the outcome prediction:

$$r = \begin{cases} 1, & \text{if } \hat{D} = D, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Importantly, no process label is used in the reward. We optimize the reward model with GRPO (Shao et al., 2024), a critic-free PPO-style algorithm (Schulman et al., 2017). For each input $(q, \mathbf{x})$, we sample a group of G rollouts $\{(Z^{(i)}, \hat{\boldsymbol{Y}}^{(i)}, \hat{D}^{(i)})\}_{i=1}^{G}$ and compute a group-relative advantage based

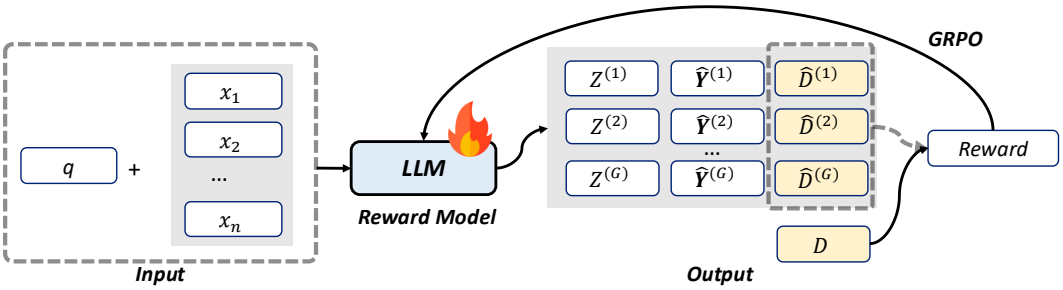


Figure 3: **Overview of TIPS.** Given an input trajectory, the model first generates a thinking chain and then predicts both process labels and an outcome label. During RL, only the outcome-level prediction is used to compute the reward, while the generated process labels receive no direct supervision.

on the rewards $\{r^{(i)}\}_{i=1}^{G}$ defined in Eq. 1. Rollouts that produce the correct outcome verdict receive higher relative advantage and are reinforced. Therefore, when correct outcome prediction depends on tracking step validity, outcome-only RL selects for thoughts that perform more faithful process verification. We provide the full GRPO objective and implementation details in Appendix C. At inference time, we use the generated process labels $\hat{\boldsymbol{Y}}$ directly as step-level labels. Thus, TIPS requires only outcome-labeled trajectories for training, but yields a PRM-style verifier at test time.

3.3 ANALYSIS

We provide an information-theoretic account of why outcome-only optimization induces process supervision. We consider the random variables $(Q, \mathbf{X}) \sim \mathcal{D}$ corresponding to the instances $(q, \mathbf{x})$ defined in Sec. 3.2. Let $\mathbf{Y} = (Y_1, \ldots, Y_n)$ and $D \in \{0, 1\}$ denote the ground-truth step-level labels and outcome label, respectively. Let $R = \mathrm{enc}_\theta(Q, \mathbf{X})$ denote the model's pre-reasoning representation, and let $Z \sim \pi_\theta(\cdot \mid Q, \mathbf{X})$ denote the generated reasoning trace used to produce the final verdict. We write $\hat{D} = g(R, Z)$ for the parsed outcome prediction. For the analysis, we treat the parser g as deterministic, so the randomness of $\hat{D}$ comes only from the sampled trace Z. All mutual information quantities are taken over the joint distribution induced by $\mathcal{D}$ and π_θ.

Assumption 1 (Nontrivial Verification)**.** *The pre-reasoning representation does not fully determine the outcome label: there exists $\varepsilon > 0$ such that*

$$H(D \mid R) \geq \varepsilon. \tag{2}$$

This rules out the degenerate case where the model can infer the final verdict directly from R, which is natural in nontrivial verification settings where a single-pass representation is insufficient (Li et al., 2024).

Theorem 1 (Outcome Information Lower Bound)**.** *Let $P_e = \mathbb{P}[\hat{D} \neq D]$. Then*

$$I(Z; D \mid R) \geq H(D \mid R) - H_b(P_e), \tag{3}$$

where $H_b(\cdot)$ is the binary entropy function.

By Fano's inequality (Cover & Thomas, 2012), low outcome error implies that the thinking chain must explain a nontrivial fraction of the uncertainty about D that remains after observing R. We next relate this outcome information to process information. The key quantity is the residual uncertainty $H(D \mid \mathbf{Y}, R)$, which measures how much the outcome remains undetermined even after the step-level labels are known.

Theorem 2 (Process Information Lower Bound)**.** *For arbitrary process labels $\mathbf{Y}$,*

$$I(Z; \mathbf{Y} \mid R) \geq I(Z; D \mid R) - H(D \mid \mathbf{Y}, R). \tag{4}$$

Theorem 2 shows that outcome-relevant information in Z must also reveal process information when the outcome is largely determined by the step-level correctness pattern. This condition naturally holds in mathematical reasoning: once the validity of the intermediate steps is known, the correctness of the final answer is often nearly determined, making $H(D \mid \mathbf{Y}, R)$ small.

The proof of the above theorems can be found in Appendix D. Combining Theorems 1 and 2 gives

$$I(Z;\mathbf{Y} \mid R) \geq \underbrace{H(D \mid R)}_{\text{verification difficulty}} - \underbrace{H_b(P_e)}_{\text{outcome error}} - \underbrace{H(D \mid \mathbf{Y}, R)}_{\text{step–outcome residual}}. \tag{5}$$

Eq. 5 captures the central mechanism behind TIPS. When verification is nontrivial, outcome error is low, and the final outcome is tightly coupled with step-level correctness, the generated thought must encode information about the reasoning process. This also explains why outcome-only GRPO can favor process-aware thoughts. For a fixed input, GRPO samples multiple rollouts with different CoT and outcome predictions. Rollouts that produce the correct outcome judgment receive higher relative advantage and are reinforced. When correct outcome prediction depends on tracking step validity, this reward signal implicitly selects for thoughts that encode process-relevant information, even though the reward never directly observes step-level labels.

The same analysis identifies when the transfer should weaken. **First**, input-level shortcuts weaken the first term: if the outcome can already be predicted from R, then $H(D \mid R)$ is small and the thought channel need not encode process information. **Second**, ineffective outcome optimization weakens the second term: if RL fails to reduce the outcome error P_e, then $H_b(P_e)$ remains large and the lower bound in Eq. 5 becomes loose. **Third**, loose step–outcome coupling weakens the third term: when step labels only indirectly determine task success, $H(D \mid \mathbf{Y}, R)$ is large, so outcome information need not translate into process information. **Fourthly**, even when useful information is required beyond R, a weak backbone may fail to express it as faithful step-by-step reasoning. We examine these regimes empirically in Sec. 4.4.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Models. To evaluate the generalization of TIPS across different backbone models, we conduct experiments using both instruct and thinking models of various scales. The models we experiment include Qwen (Yang et al., 2025; 2024), DeepSeek (DeepSeek-AI, 2025), LlaMa (Grattafiori et al., 2024; Wang et al., 2025b), and SmolLM (Bakouch et al., 2025) families. The implementation details can be found in Appendix E.

Benchmarks and Metrics. We evaluate TIPS in two domains: mathematical reasoning and LLM-agent tasks. For mathematical reasoning, we first adopt the standard **Best-of-**8 (Lightman et al., 2023) evaluation to assess whether a PRM can improve downstream response selection. Given eight sampled responses from Qwen2.5-7B-Instruct for each problem, each PRM selects the response with the highest score, and we report the selected-answer accuracy on GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), College Math (Tang et al., 2024), Minerva (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). To ensure a fair comparison, all PRMs are evaluated on the same set of sampled responses. We additionally report Majority Vote@8 as a reference baseline and Pass@8 as an oracle upper bound. Beyond response selection, we further evaluate the PRMs' ability to detect step-level errors on ProcessBench (Zheng et al., 2025a), where models are required to identify the first erroneous step or determine that all steps are correct. For ProcessBench, we report the official F1 score. Finally, we evaluate on AgentProcessBench (Fan et al., 2026b), where PRMs assign labels to assistant responses in multi-turn agent trajectories. We report the benchmark's official First-Error Accuracy (FirstErrAcc).

Baselines. We compare TIPS with four groups of baselines, organized by the form and cost of their supervision. **(1) LLM-as-a-Judge** baselines use the open-source backbones as well as stronger proprietary GPT and Claude models as prompt-only critics, testing whether reliable process evaluation can be obtained without PRM training. **(2) Published discriminative PRMs** include representative step-wise reward models such as Math-Shepherd (Wang et al., 2024), Qwen2.5-Math-PRM (Zhang et al., 2025b), RLHFlow-PRM (Dong et al., 2024), EurusPRM (Cui et al., 2025), UniversalPRM (Tan et al., 2026), and Scan-PRM (Ding et al., 2026), whose supervision comes from Monte-Carlo-estimated step labels, stronger-model-distilled annotations, human process labels, or their combinations. **(3) Published generative PRMs** include GenPRM (Zhao et al., 2025) and

Table 1: **Evaluation results of F1 on ProcessBench** (%). `Human`: human-annotated step labels; `MC`: pseudo process labels obtained via Monte Carlo estimation; `Distill`: stronger-model-derived supervision; `Outcome`: trajectory-level outcome labels only.

Model	Supervision	GSM8K	MATH	Olympiad Bench	OmniMath	Avg.
LLM-as-a-Judge						
Qwen3-4B-Instruct-2507	N/A	64.8	68.7	63.6	58.1	63.8
Qwen3-4B-Thinking-2507	N/A	52.1	70.1	59.5	59.4	60.3
GPT-5.4-Instruct	N/A	89.8	84.6	74.1	69.1	79.4
Claude-4.7-Opus	N/A	85.3	82.9	75.7	73.4	79.3
o1-mini	N/A	93.2	88.9	87.2	82.4	87.9
Published Discriminative PRMs						
UniversalPRM-7B	Distill	85.8	77.7	67.6	66.4	74.3
Qwen2.5-Math-PRM-72B	MC+Distill	<u>87.3</u>	80.6	74.3	71.1	<u>78.3</u>
RLHFlow-PRM-Deepseek-8B	MC	38.8	33.8	16.9	16.9	26.6
Qwen2.5-Math-7B-Math-Shep	MC	62.5	31.6	13.7	7.7	28.9
EurusPRM-Stage1	Outcome	54.7	41.2	24.7	17.5	34.5
EurusPRM-Stage2	Outcome+Distill	67.0	53.2	35.4	30.7	46.6
Qwen2.5-Math-7B-PRM800K	Human	68.2	62.6	50.7	44.3	56.5
Published Generative PRMs						
ThinkPRM-14B	Human+Distill	65.3	64.0	53.9	52.9	59.1
GenPRM-32B	MC+Distill	83.1	81.7	72.8	<u>72.8</u>	77.6
Matched-Data Qwen3-4B Baselines						
SCAN-Qwen3-4B-Instruct-2507	MC	81.3	68.7	52.0	45.0	61.7
AutoPSV-Qwen3-4B-Instruct-2507 (3.2K)	Outcome	39.0	33.4	20.3	21.5	28.6
AutoPSV-Qwen3-4B-Instruct-2507 (197K)	Outcome	63.6	48.4	34.3	34.8	45.3
ImplicitPRM-Qwen3-4B-Instruct-2507 (3.2K)	Outcome	52.6	30.8	9.1	14.4	26.7
ImplicitPRM-Qwen3-4B-Instruct-2507 (197K)	Outcome	56.8	44.8	30.3	34.0	41.5
Ours						
TIPS-Qwen3-4B-Instruct-2507 (3.2K)	Outcome	84.6	<u>82.6</u>	<u>74.5</u>	69.0	77.7
TIPS-Qwen3-4B-Thinking-2507 (3.2K)	Outcome	**89.5**	**87.7**	**83.5**	**80.0**	**85.2**

ThinkPRM (Khalifa et al., 2026), which cast process evaluation as conditional generation rather than scalar scoring. Although they share the same formulation as TIPS, their supervision depends on 32B-level thinking LLMs to generate rationales, followed by costly filtering with Monte Carlo estimates or human step annotations. **(4) Outcome-supervised implicit PRMs** include AutoPSV (Lu et al., 2024) and ImplicitPRM (Yuan et al., 2025), which learn process evaluators from trajectory-level outcome labels only. These methods form the closest weak-supervision baselines to TIPS, while TIPS differs by explicitly verbalizing process rationale. Note that the published off-the-shelf PRMs are trained on different backbones with different data. For a fair comparison among outcome-supervised implicit PRMs, we re-implement AutoPSV and ImplicitPRM under the same setting as TIPS. We also re-implement SCAN-PRM, a representative state-of-the-art Monte-Carlo-based PRM, as a strong supervised reference baseline.

4.2 RESULTS ON MATH

Table 1 presents the results on ProcessBench, evaluating whether models can identify step-level errors in reasoning trajectories. We highlight four observations.

- **Prior outcome-supervised PRMs do not induce reliable process labels.** As shown in Table 1, prior outcome-supervised PRMs are weak step-level verifiers. Specifically, AutoPSV-197K and ImplicitPRM-197K achieve only 45.3 and 41.5 F1 on ProcessBench, respectively. This suggests that their preferences are not grounded in explicit analysis of intermediate reasoning steps.
- **TIPS turns outcome-only supervision into process-aware verification.** Despite using only 3.2K outcome-labeled trajectories, TIPS-Qwen3-4B-Instruct-2507 achieves 77.7 average F1 on ProcessBench, substantially outperforming AutoPSV-197K and ImplicitPRM-197K. This suggests

Table 2: Comparison of FirstErrAcc on AgentProcessBench (%).

Model	HotPotQA	GAIA	BFCL	τ^2-Bench	Avg.
Qwen3-4B-Instruct-2507	57.2	35.6	34.8	42.4	42.5
w/ TIPS	58.0 (+0.8)	38.0 (+2.4)	35.6 (+0.8)	48.4 (+6.0)	45.0 (+2.5)
Qwen3-4B-Thinking-2507	60.4	33.6	37.2	43.6	43.7
w/ TIPS	61.2 (+0.8)	41.2 (+7.6)	43.6 (+6.4)	50.4 (+6.8)	49.1 (+5.4)
Qwen3-8B-Thinking	56.8	31.6	30.8	46.8	41.5
w/ TIPS	53.6 (-3.2)	48.8 (+17.2)	32.4 (+1.6)	49.2 (+2.4)	46.0 (+4.5)

that the improvement comes not from simply scaling outcome data, but from the explicit CoT used in TIPS, which allows outcome supervision to be converted into process-level evidence. This is consistent with Eq. 5, since in mathematical reasoning the final outcome is often largely determined by intermediate-step correctness.

- **Thinking models amplify the emergence of process verification.** When directly prompted as a judge, Qwen3-4B-Thinking-2507 is slightly weaker than Qwen3-4B-Instruct-2507 on ProcessBench. After training with TIPS, however, the thinking backbone reverses this gap and improves the average F1 from 77.7 to 85.2. This observation is consistent with our analysis in Sec. 3.3: thinking models generate longer thoughts that encode richer step-level information, which can be further reinforced during RL and lead to stronger process verification.
- **TIPS achieves the strongest performance among evaluated trained PRMs.** With only a 4B-scale backbone, TIPS-Qwen3-4B-Thinking-2507 achieves **85.2** average F1 on ProcessBench, outperforming previous state-of-the-art baselines, including Qwen2.5-Math-PRM-72B and GenPRM-32B, and proprietary models like GPT-5.4-Instruct and Claude-4.7-Opus, while still trailing o1-mini.

4.3 RESULTS ON AGENT

Table 2 reports FirstErrAcc on AgentProcessBench. We highlight two observations.

- **Outcome-only RL induces step-level localization beyond mathematics.** On AgentProcessBench, TIPS improves average FirstErrAcc for all three backbones, showing that the effect is not limited to mathematical reasoning. Same to the math domain, the gains are larger for thinking models: Qwen3-4B-Thinking-2507 and Qwen3-8B-Thinking improve by $+5.4$ and $+4.5$ average FirstErrAcc, respectively, compared with $+2.5$ for Qwen3-4B-Instruct-2507. This supports our central hypothesis: when a model is rewarded only for the final judgment, RL can still reinforce intermediate verification behaviors that help produce that judgment.
- **Agent tasks exhibit weaker outcome–process coupling.** AgentProcessBench yields smaller and more heterogeneous gains than ProcessBench. This is because of weaker outcome–process coupling. In math, judging the final answer typically requires checking each derivation step. In agent tasks, failures may instead come from missing actions, insufficient evidence, or unproductive exploration, which are not always tied to a clearly invalid observed step. Outcome feedback is therefore less localized, making process supervision harder to induce.

4.4 DISCUSSIONS

Best-of-8 Evaluation. Table 3 reports Best-of-8 reranking accuracy with Qwen2.5-7B-Instruct as the policy model. For TIPS, we first filter the candidate trajectories predicted to have correct final outcomes and then selects the retained trajectory with the highest mean predicted step-validity score. For baselines, we follow their default settings. We highlight two observations.

- **Strong prompt-only judges can outperform prior state-of-the-art trained PRMs.** Without any PRM training, Qwen3-4B-Thinking-2507 achieves 72.8 average accuracy, outperforming the state-of-the-art MC-supervised SCAN-PRM with 69.2 and the outcome-supervised ImplicitPRM with 68.8. This suggests that generative judges are already strong process-aware rerankers even without PRM training.
- **TIPS further improves generative PRMs with much lower supervision cost.** Using only 3.2K outcome-only trajectories, TIPS-Qwen3-4B-Thinking-2507 reaches 73.3 Avg, surpassing the 197K-scale ImplicitPRM baseline by $+4.5$ points. This demonstrates the data efficiency of

Table 3: Best-of-8 reranking evaluation with Qwen2.5-7B-Instruct as the policy model.

Model	Supervision	GSM8K	MATH	College Math	Olympiad Bench	Minerva	Avg.
Baselines and Upper Bound							
Majority Vote@8	N/A	93.8	80.1	66.4	45.9	52.6	67.8
Pass@8 (Upper Bound)	Oracle	97.2	89.0	74.5	62.7	66.9	78.1
LLM-as-a-Judge							
Qwen3-4B-Instruct-2507	N/A	95.1	85.2	69.1	52.6	56.3	71.6
Qwen3-4B-Thinking-2507	N/A	95.0	**87.5**	**69.5**	56.2	55.9	72.8
Matched-Data Qwen3-4B Baselines							
SCAN-PRM-Qwen3-4B-Instruct-2507 (197K)	MC	94.8	82.1	68.1	48.3	52.6	69.2
AutoPSV-Qwen3-4B-Instruct-2507 (3.2K)	Outcome	93.3	77.6	65.7	43.4	49.6	65.9
AutoPSV-Qwen3-4B-Instruct-2507 (197K)	Outcome	94.2	79.0	66.5	43.9	53.3	67.4
ImplicitPRM-Qwen3-4B-Instruct-2507 (3.2K)	Outcome	93.8	78.4	66.8	43.3	50.4	66.5
ImplicitPRM-Qwen3-4B-Instruct-2507 (197K)	Outcome	94.2	81.8	68.4	46.2	53.3	68.8
Ours							
TIPS-Qwen3-4B-Instruct-2507 (3.2K)	Outcome	95.0	86.9	69.4	55.4	**57.4**	72.8
TIPS-Qwen3-4B-Thinking-2507 (3.2K)	Outcome	**95.5**	87.4	**69.5**	**56.7**	**57.4**	**73.3**

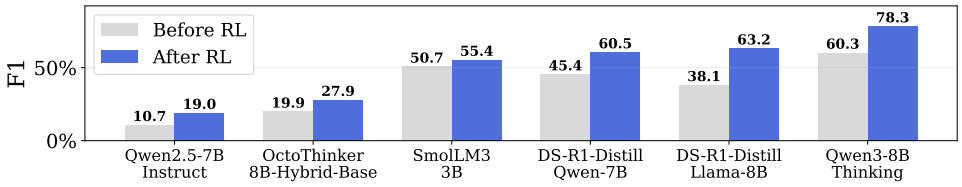


Figure 4: TIPS improves ProcessBench F1 across diverse backbones.

our framework in converting outcome-only supervision into stronger process evaluation ability. Meanwhile, TIPS still trails the Pass@8 oracle by 4.8 points, with 73.3 Avg compared to 78.1 for Pass@8, leaving further headroom within the same candidate set.

TIPS generalizes across model families and scales. To test whether TIPS is specific to the Qwen-2507-4B series, we run additional math-domain experiments on diverse backbones. As shown in Figure 4, TIPS improves ProcessBench F1 across DeepSeek-R1-distilled models, Llama-based models, SmolLM, Qwen2.5-7B-Instruct and Qwen3-8B. This indicates that the gains are not tied to a single backbone family, but extend to models with different architectures and scales. In Appendix H, we examine whether these gains can be explained solely by improved problem-solving ability.

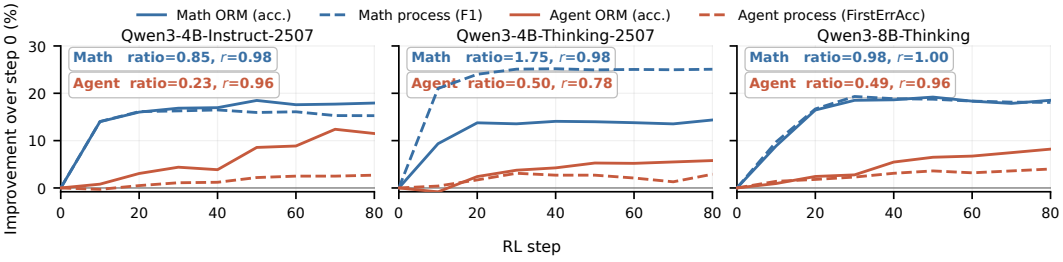


Figure 5: **Outcome–process coupling during outcome-supervised RL.** Solid lines show the absolute improvement in ORM accuracy, while dashed lines show the absolute improvement in the corresponding PRM metric. All values are measured relative to the base model. The annotated gain ratio is defined as $\Delta_{\text{process}}/\Delta_{\text{ORM}}$, where each Δ is the endpoint improvement. Pearson r denotes the correlation between ORM accuracy and the corresponding process metrics.

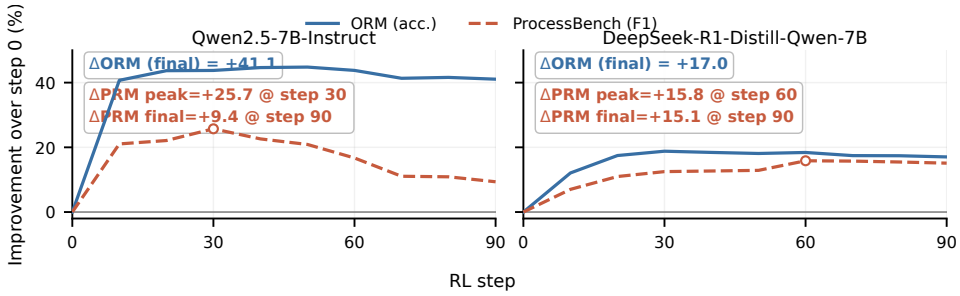


Figure 6: **Model priors affect outcome-to-process transfer.** Qwen2.5-7B-Instruct improves ORM rapidly but shows degraded ProcessBench F1 at later checkpoints, while the DeepSeek-R1-distilled model maintains coupled ORM and PRM improvements.

Outcome–process coupling during outcome-supervised RL. We analyze the training dynamics of three backbones trained only with outcome rewards. As shown in Figure 5, ORM accuracy and process-level metrics generally improve together, with Pearson correlations ranging from 0.78 to 1.00. This suggests that outcome optimization can implicitly induce process-level verification. Meanwhile, we observe that math tasks show larger gain ratios, indicating more direct transfer from outcome judgment to step-level verification, while agent tasks show weaker transfer. This is likely because, in agent settings, the relation between local step correctness and final task success is more indirect.

Model priors affect the stability of outcome-to-process transfer. To probe a possible failure mode of TIPS, we compare Qwen2.5-7B-Instruct with its DeepSeek-R1-distilled counterpart. The results on ProcessBench are shown in Figure 6. The distilled model shows coupled improvement: both ORM accuracy and ProcessBench F1 increase and then stabilize. In contrast, Qwen2.5-7B-Instruct improves ORM rapidly, but its ProcessBench F1 peaks early and then deteriorates. This suggests that TIPS relies on the model's initial prior. A weaker model may fit the outcome objective through a shortcut: it learns to predict whether the solution is ultimately wrong, while its step-level judgments remain incomplete or weakly grounded. The DeepSeek-distilled model, which has a stronger step-by-step verification prior, instead converts outcome rewards into more stable process-level gains. These findings suggest applying TIPS to backbones with strong CoT priors. Our observations also suggest a process-label-free early-stopping heuristic: stop training when ORM accuracy plateaus and response length shows a decline. This heuristic effectively help mitigate degradation in our experiments.

Thought is essential for PRM learning. To isolate the role of explicit reasoning, we replace CoT prompting with a label-only prompt that directly outputs process and outcome labels. As shown in Table 4, the label-only variant has much lower base ProcessBench F1 and gains little from outcome-supervised RL, while the CoT variant achieves a large PRM improvement. These results show that TIPS does not learn reliable process verification from labels alone; the gains are mediated by reasoning traces that RL can shape. The label-only variant's near-zero entropy, shorter outputs, and smaller KL loss further suggest that it receives little effective learning signal.

Table 4: **Prompting ablation on Qwen3-4B-Instruct-2507.** Values are measured at the end of the first epoch.

Metric	CoT	Label-only
Base F1	63.8	47.5
PRM gain	13.9	0.7
ORM gain	17.6	-5.6
Entropy	0.31	0.002
Avg. length	1828.7	47.9
Max KL loss	1×10^{-3}	1.5×10^{-5}

5 CONCLUSION

We presented TIPS, an outcome-only RL framework that induces process supervision through the chain-of-thought of a generative reward model. With only 3.2K outcome-labeled examples, TIPS-Qwen3-4B-Thinking-2507 achieves **85.2 F1** on ProcessBench and improves AgentProcessBench FirstErrAcc across all tested backbones. Our analyses show that CoT and the backbone's CoT prior are key carriers of the induced signal, while weak outcome–process coupling and outcome shortcuts limit the transfer. Future work may combine TIPS with light Monte Carlo priors or CoT distillation.

AI USE STATEMENT

We used LLMs for language polishing, but not for generating research ideas or analyzing experimental results.

ETHICS STATEMENT

We use open-weight policy models and publicly available datasets, and evaluate on public benchmarks. We do not foresee additional ethical concerns.

REPRODUCIBILITY STATEMENT

We detail the backbone models, datasets, and hyperparameters in Sections 4.1 and Appendix E to facilitate reproducibility.

REFERENCES

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Haotian Chen, Xin Cong, Shengda Fan, Yuyang Fu, Ziqin Gong, Yaxi Lu, Yishan Li, Boye Niu, Chengjun Pan, Zijun Song, et al. Agentcpm-explore: Realizing long-horizon deep exploration for edge-scale agents. *arXiv preprint arXiv:2602.06485*, 2026a.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru WANG, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. RM-r1: Reward modeling as reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=1ZqJ6jj75q.

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

T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 2012. ISBN 9781118585771. URL https://books.google.com.tw/books?id=VWq5GG6ycxMC.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Yuyang Ding, Xinyu Shi, Juntao Li, xiaobo liang, Zhaopeng Tu, and Min Zhang. SCAN: Self-denoising monte carlo annotation for robust process reward learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=ifsyZYYDNs.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.

Shengda Fan, Xin Cong, Zhong Zhang, Yuepeng Fu, Yesai Wu, Hao Wang, Xinyu Zhang, Enrui Hu, and Yankai Lin. Generalizing experience for language agents with hierarchical metaflows. *Advances in Neural Information Processing Systems*, 38:64103–64132, 2026a.

Shengda Fan, Xuyan Ye, Yupeng Huo, Zhi-Yuan Chen, Yiju Guo, Shenzhi Yang, Wenkai Yang, Shuqi Ye, Jingwen Chen, Haotian Chen, et al. Agentprocessbench: Diagnosing step-level process quality in tool-using agents. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 8823–8834, 2026b.

Shengda Fan, Xuyan Ye, and Yankai Lin. Darc: Decoupled asymmetric reasoning curriculum for llm evolution. *arXiv preprint arXiv:2601.13761*, 2026c.

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

Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Reward reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=V8Kbz7l2cr.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

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

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=FPVCb0WMuN. J2C Certification.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations*, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025b.

Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yingjia Wan, Yunlong Feng, and Zhijiang Guo. Autopsv: Automated process-supervised verifier. *Advances in Neural Information Processing Systems*, 37:79935–79962, 2024.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

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

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

Xiaoyu Tan, Tianchu Yao, Chao Qu, Bin Li, Minghao Yang, Dakuan Lu, Haozhe Wang, Xu Yinghui, and Xihe Qiu. Aurora: Automated training framework of universal process reward models via ensemble prompting and reverse verification. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 1378–1389, 2026.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. Mathscale: Scaling instruction tuning for mathematical reasoning. In *Forty-first International Conference on Machine Learning*, 2024.

MiniCPM Team, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, et al. Minicpm4: Ultra-efficient llms on end devices. *arXiv preprint arXiv:2506.07900*, 2025.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Yanlin Wang, Yanli Wang, Daya Guo, Jiachi Chen, Ruikai Zhang, Yuchi Ma, and Zibin Zheng. Rlcoder: Reinforcement learning for repository-level code completion. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pp. 1140–1152. IEEE, 2025a.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *arXiv preprint arXiv:2505.10320*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=2a36EMSSTp.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. In *International Conference on Machine Learning*, pp. 73511–73525. PMLR, 2025.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.547. URL https://aclanthology.org/2025.findings-acl.547/.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025b.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. Genprm: Scaling test-time compute of process reward models via generative reasoning. *arXiv preprint arXiv:2504.00891*, 2025.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, 2025b.

A Limitations

Our study has three main limitations. First, we focus on pure-text reasoning and agent trajectories, and do not evaluate multimodal settings where process verification must be grounded in images, videos,

or other non-textual observations. Second, TIPS relies on sufficient outcome–process coupling: the induced process signal is weaker when final task success is only indirectly related to local step correctness. Third, the method depends on the backbone's ability to express verification behavior through its generated thoughts. For weaker backbones or label-only prompting, outcome-supervised RL may improve outcome prediction without reliably yielding faithful step-level judgments.

B BROADER IMPACTS

TIPS may reduce the cost of building process reward models by inducing step-level verification from trajectory-level outcome labels, rather than requiring human step annotations or extensive Monte-Carlo estimation. This can benefit reasoning evaluation, educational feedback, and agent debugging, where localizing errors is often more informative than judging only final outcomes. However, stronger reward models may also increase the capability of systems optimized with them, including potentially harmful or unsafe agents. Moreover, because the process labels produced by TIPS are induced indirectly from outcome rewards, they may be biased, miscalibrated, or unreliable in domains where final success is weakly tied to local step correctness. These labels should therefore not be treated as ground truth in high-stakes settings. Responsible use of TIPS requires domain-specific validation, human oversight, and separate evaluation of both outcome-level and step-level behavior. Future work should study robustness, calibration, and misuse risks in broader agentic and multimodal applications.

C GRPO OBJECTIVE

For completeness, we provide the full GRPO objective used in our RL training. For each input $(q, \mathbf{x})$, we sample a group of G rollouts $\{\mathbf{o}^{(1)}, \ldots, \mathbf{o}^{(G)}\}$ from the old policy $\pi_{\theta_{\text{old}}}(\cdot \mid q, \mathbf{x})$. Each rollout receives a binary reward $r^{(i)}$ according to Eq. 1. We then compute a group-relative advantage as

$$\hat{A}^{(i)} = r^{(i)} - \frac{1}{G}\sum_{j=1}^{G} r^{(j)}. \tag{6}$$

Following recent practice (Liu et al., 2025a;b), we use a non-normalized variant of GRPO, i.e., we do not divide the advantage by the group standard deviation.

The model is updated by maximizing the clipped surrogate objective:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,\mathbf{x})\sim\mathcal{D},\, \{\mathbf{o}^{(i)}\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q,\mathbf{x})} \Bigg[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|\mathbf{o}^{(i)}|} \sum_{t=1}^{|\mathbf{o}^{(i)}|} \min \Bigg(\rho_t^{(i)}(\theta)\hat{A}^{(i)}, \\ \text{clip}\Big(\rho_t^{(i)}(\theta), 1-\epsilon, 1+\epsilon\Big)\,\hat{A}^{(i)} \Bigg) \Bigg],\end{aligned} \tag{7}$$

where

$$\rho_t^{(i)}(\theta) = \frac{\pi_\theta(o_t^{(i)} \mid q, \mathbf{x}, \mathbf{o}_{<t}^{(i)})}{\pi_{\theta_{\text{old}}}(o_t^{(i)} \mid q, \mathbf{x}, \mathbf{o}_{<t}^{(i)})}, \tag{8}$$

$|\mathbf{o}^{(i)}|$ is the length of the i-th rollout, and ϵ is the clipping threshold.

D DERIVATIONS FOR THINKING-INDUCED PROCESS SUPERVISION

This appendix provides proofs of Theorems 1 and 2.

Proof of Theorem 1. By the definition of conditional mutual information,

$$I(Z; D \mid R) = H(D \mid R) - H(D \mid R, Z). \tag{9}$$

It therefore suffices to upper-bound $H(D \mid R, Z)$. Since $\hat{D} = g_\theta(R, Z)$ is a deterministic function of (R, Z), the pair $(R, \hat{D})$ is also a deterministic function of (R, Z). Therefore,

$$H(D \mid R, Z) \leq H(D \mid R, \hat{D}). \tag{10}$$

Moreover, conditioning cannot increase entropy, so

$$H(D \mid R, \hat{D}) \leq H(D \mid \hat{D}). \tag{11}$$

Now D and $\hat{D}$ are binary random variables. Applying Fano's inequality with error probability $P_e = \mathbb{P}[\hat{D} \neq D]$ gives

$$H(D \mid \hat{D}) \leq H_b(P_e) + P_e \log(|\{0, 1\}| - 1) = H_b(P_e). \tag{12}$$

Combining the inequalities above,

$$H(D \mid R, Z) \leq H(D \mid R, \hat{D}) \leq H(D \mid \hat{D}) \leq H_b(P_e), \tag{13}$$

and substituting back yields

$$I(Z; D \mid R) = H(D \mid R) - H(D \mid R, Z) \geq H(D \mid R) - H_b(P_e). \tag{14}$$

Proof of Theorem 2. Apply the chain rule for conditional mutual information to $I(Z; \mathbf{Y}, D \mid R)$ in two different orders. First,

$$I(Z; \mathbf{Y}, D \mid R) = I(Z; \mathbf{Y} \mid R) + I(Z; D \mid \mathbf{Y}, R). \tag{15}$$

Second,

$$I(Z; \mathbf{Y}, D \mid R) = I(Z; D \mid R) + I(Z; \mathbf{Y} \mid D, R). \tag{16}$$

Equating the right-hand sides and rearranging gives

$$I(Z; \mathbf{Y} \mid R) = I(Z; D \mid R) + I(Z; \mathbf{Y} \mid D, R) - I(Z; D \mid \mathbf{Y}, R). \tag{17}$$

Since mutual information is nonnegative,

$$I(Z; \mathbf{Y} \mid R) \geq I(Z; D \mid R) - I(Z; D \mid \mathbf{Y}, R). \tag{18}$$

Finally,

$$I(Z; D \mid \mathbf{Y}, R) = H(D \mid \mathbf{Y}, R) - H(D \mid \mathbf{Y}, R, Z) \leq H(D \mid \mathbf{Y}, R), \tag{19}$$

because entropy is nonnegative. Substituting yields

$$I(Z; \mathbf{Y} \mid R) \geq I(Z; D \mid R) - H(D \mid \mathbf{Y}, R). \tag{20}$$

E Implementation Details.

For the math domain, we sample 3,200 trajectories from the published SCAN-Pro[1] dataset, which is constructed from the MATH training set. For the agent domain, we construct an RL training corpus of 2,905 trajectories from HotpotQA Yang et al. (2018), GAIA Mialon et al. (2023), τ^2-Bench Barres et al. (2025), and BFCL Patil et al. (2025). To mitigate test-set leakage, we decontaminate the training corpus against AgentProcessBench by excluding trajectories with task-prompt 5-gram Jaccard similarity above 0.8. All experiments are conducted on 8 NVIDIA A800 GPUs with 80GB memory each. We conduct RL training with the `verl` framework. We set the maximum rollout response length to 16K tokens, set the KL penalty coefficient to 0, train for 1 epoch with AdamW, and use batch size of 32 and a learning rate of 1×10^{-6}.

F Prompt Design

In this section, we present the used prompt in Tips.

[1] https://huggingface.co/datasets/dyyyyyyyy/SCAN-Pro

Prompt for Mathematical Reasoning

The following is a math problem and a solution split into paragraphs and indexed from 0.

[Math Problem]
`PROBLEM`

[Solution]
`TAGGED_RESPONSE`

Your task is to judge the solution paragraph by paragraph.
First, think through the solution carefully and identify where the reasoning first becomes invalid.
Then output a JSON object with:

- `step_labels`: a mapping from paragraph index to 1 or 0.
  - 1 means the paragraph is correct and remains logically valid given the previous paragraphs.
  - 0 means the paragraph is incorrect, unsupported, or depends on an earlier incorrect paragraph.
- `final_label`: 1 if the final answer of the whole solution is correct, otherwise 0.

Rules:

- You must reason before the final JSON.
- `step_labels` must contain every paragraph index exactly once.
- Use only 1 or 0 for every label.
- If an earlier paragraph is wrong and a later paragraph relies on it, that later paragraph should also be 0.
- The final output must end with a ```` ```json ```` code block containing only the final JSON object.

Illustrative output format:

````
```json
{
  "step_labels": {"0": 1, "1": 1, "2": 0},
  "final_label": 0
}
```
````

G Human Evaluation of Process-Level Feedback

To assess the quality of the process-level labels generated by TIPS, we sampled 60 trajectories from ProcessBench, with 15 trajectories from each subset. Two PhD-level annotators evaluated the feedback generated by TIPS-Qwen3-4B-Instruct-2507. The evaluation examined whether the model correctly localized the first error and whether its accompanying rationale accurately explained the underlying error. The model correctly localized the first error in 47/60 cases (78.3%). Among these correctly localized cases, the rationale accurately explained the underlying error in 38/47 cases (80.9%). Thus, 38/60 cases (63.3%) exhibited both correct error localization and a human-validated explanation. These results provide additional evidence that TIPS can generate process-level feedback that both identifies and explains reasoning errors.

H Does TIPS Simply Improve Problem-Solving Ability?

To investigate whether the improvements from TIPS can be explained by stronger mathematical problem solving, we conduct two complementary analyses. First, we evaluate how TIPS training affects the direct problem-solving performance of its backbones. Second, we compare TIPS with a solver-format RL baseline and additional models with higher solver benchmark scores. We evaluate direct problem solving using mean@4 accuracy on AIME 2026 and HMMT February 2026, and process verification using F1 on ProcessBench.

Direct problem-solving performance. As shown in Table 5, TIPS improves ProcessBench F1 by 24.9 points for Qwen3-4B-Thinking-2507 and 13.9 points for Qwen3-4B-Instruct-2507. These improvements are accompanied by much smaller changes in direct problem solving: the thinking variant improves modestly on both solver benchmarks, whereas the Instruct variant remains unchanged on AIME 2026 and declines on HMMT February 2026. Thus, **the substantial gains in process verification are not accompanied by comparable improvements on the two solver benchmarks**.

Table 5: **Direct problem solving and process verification.** AIME 2026 and HMMT February 2026 scores are mean@4 accuracy (%). ProcessBench scores are evaluated in F1. The solver RLVR baseline is initialized from Qwen3-4B-Instruct-2507 and trained on DAPO-Math-17K.

Model	AIME 2026	HMMT Feb. 2026	ProcessBench
Qwen3-4B-Instruct-2507	53.3	**36.4**	63.8
+ Solver RLVR	55.8	33.3	62.8
+ TIPS	53.3	31.8	**77.7**
GooseReason-4B-Instruct	**62.5**	33.3	72.7
Qwen3-4B-Thinking-2507	80.0	53.0	60.3
+ TIPS	80.8	54.6	**85.2**
Qwen3-30B-A3B-Thinking-2507	**86.7**	**74.2**	80.3

Comparison with solver and verification baselines. We train a same-backbone control using standard solver-format reinforcement learning with verifiable rewards (RLVR) on DAPO-Math-17K, with questions as inputs and final-answer correctness as the reward. This baseline scores higher than TIPS-Qwen3-4B-Instruct-2507 on both AIME 2026 and HMMT February 2026, but trails it by 14.9 F1 points on ProcessBench. Therefore, the solver-format RLVR baseline does not reproduce the process-verification gains achieved by TIPS, despite its higher direct problem-solving scores.

The additional comparisons show a similar pattern. GooseReason-4B-Instruct outperforms TIPS-Qwen3-4B-Instruct-2507 on both solver benchmarks but trails it by 5.0 F1 points on ProcessBench. Likewise, Qwen3-30B-A3B-Thinking-2507 achieves higher solver scores than TIPS-Qwen3-4B-Thinking-2507 while scoring 4.9 F1 points lower on ProcessBench. These comparisons demonstrate that, among the evaluated models, **higher solver benchmark scores do not necessarily translate into better step-error localization**. Together, these findings support the interpretation that TIPS strengthens process verification beyond the improvements observed in direct problem solving.