STRIDE: Learnable Stepwise Language Feedback for LLM Reasoning

Junjie Zhang[1]*, Guozheng Ma[1], Shunyu Liu[1], Zetian Hu[1], Yongcheng Jing[1], Ting-En Lin[2], Yongbin Li[2]†, Dacheng Tao[1]†

[1]Generative AI Lab, College of Computing and Data Science, Nanyang Technological University, Singapore 639798
[2]Alibaba Token Hub, Alibaba Group

Abstract

Recent advances in Reinforcement Learning (RL) have underscored its potential for incentivizing reasoning capabilities of Large Language Models (LLMs). However, existing step-level efforts suffer from costly annotations that limit domain coverage, while scalar scores further impose an information bottleneck, offering insufficient semantic bandwidth to improve intermediate decisions. Alternative language-critique approaches, which rely on frozen or external critics, provide richer textual feedback but lack the scalability needed for sustained policy improvement. In this work, we propose language-driven stepwise trajectory redirection, termed as **STRIDE**, a novel training framework that shifts process supervision from scalar rewards to *learnable* stepwise language feedback. Specifically, we co-train a generator and a generative verifier using only outcome-based rewards, eliminating external annotations, while delivering sustained policy improvement through jointly aligned verifier training. The verifier's stepwise language critiques explicitly localize and explain failures, enabling the generator to redirect reasoning trajectories at intermediate steps toward alternative decisions. The trajectory redirection design guarantees harmless policy improvement, even under noisy or suboptimal verifier feedback. Experiments on diverse reasoning benchmarks show that STRIDE significantly outperforms state-of-the-art baselines, as well as achieving breakthroughs on zero-pass-rate problems where scalar methods yield no learning signal in our ablation studies, demonstrating the effectiveness of learnable stepwise language feedback for enhancing LLM reasoning.

1 Introduction

The recent surge in reasoning capabilities of LLMs across diverse domains has been largely driven by Reinforcement Learning from Verifiable Rewards (RLVR) [1, 2, 3, 4]. However, these methods rely on sparse, outcome-based rewards [3, 5, 6, 7, 8], which offer no feedback on individual reasoning steps, leaving credit assignment a fundamental unsolved challenge in multi-step reasoning.

Process Reward Models (PRMs) [9, 10, 11] advance credit assignment through step-level supervision, yet suffer from two compounding limitations. First, reliable step-level annotations are prohibitively expensive to obtain [9], confining PRMs to narrow domains. Automated labeling alleviates the cost but introduces inaccurate labels that actively mislead training with harmful gradient noise [12, 11]. Second, scalar scores impose a fundamental representational constraint: compressing high-dimensional reasoning into a single numerical value creates an information bottleneck (see §3.2),

*Email: junjie.zhang@ntu.edu.sg
†Corresponding authors: shuide.lyb@alibaba-inc.com, dacheng.tao@ntu.edu.sg

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

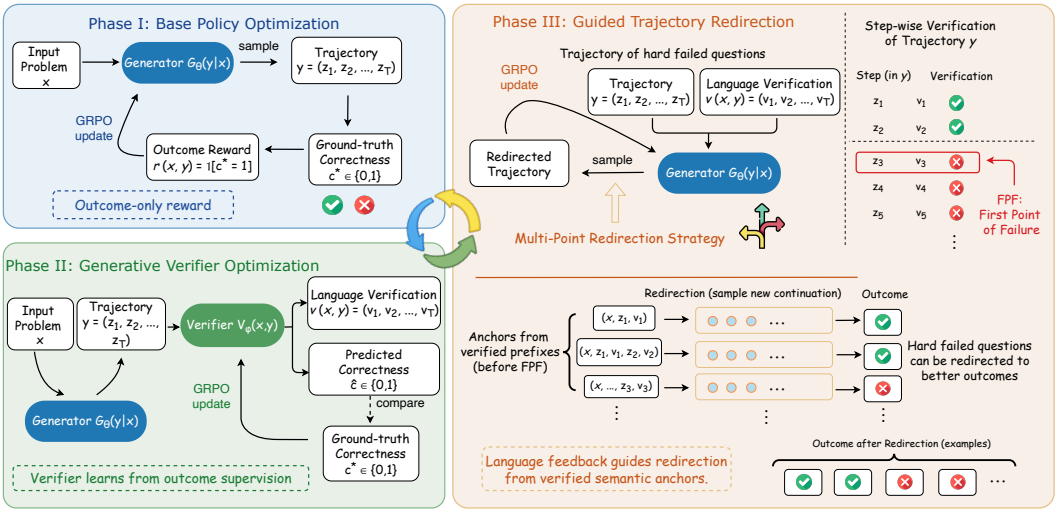


Figure 1: **Overview of the STRIDE framework. STRIDE** shifts the process supervision paradigm from unidimensional scalar rewards to high-bandwidth in-context guidance. **Phase I** builds basic reasoning capabilities through outcome-based GRPO. **Phase II** optimizes a generative verifier to decompose terminal rewards into step-level linguistic feedback v_t. **Phase III** leverages the verifier to localize the First Point of Failure (FPF) and triggers Multi-Point Redirection. By initiating reconstruction from multiple verified anchors with actionable semantic guidance, the framework effectively constrains the vast exploration space to break through reasoning stagnation.

providing insufficient semantic bandwidth to distinguish or correct qualitatively different error modes [10]. To overcome the representational constraint, critique-based methods [13, 14] and SFT-based error-corrective approaches [15, 16, 17] shift supervision from scalar to language, recovering the semantic richness that scalars discard. However, their reliance on frozen or external critics and on supervised fine-tuning limits adaptability, preventing sustained improvement as the policy evolves. Most recently, TANGO [18] co-trains a generative verifier alongside the generator, yet converts its language output back into step-level scalar rewards, reintroducing the same information bottleneck. A natural question then arises: can a feedback mechanism that is simultaneously stepwise, language-informative, and learnable resolve all of the above limitations?

In this paper, we propose **STRIDE** to answer this question. STRIDE co-trains a generator and a generative verifier using only outcome-based rewards, requiring no step-level annotations. The core insight is the shift of process supervision paradigm from scalar reward signals to *learnable* in-context language feedback: language critiques from the co-trained verifier carry the semantic direction needed to localize and rectify specific reasoning errors, and generate productive training signal even on hard problems where scalar methods yield identically zero advantage [2, 19]. Specifically, the framework operates through an interleaved three-phase schedule: Base Policy Optimization (Phase I), Generative Verifier Optimization (Phase II), and Guided Trajectory Redirection (Phase III). For challenging problems where the generator fails, STRIDE localizes the First Point of Failure (FPF) and employs a **Multi-Point Redirection Strategy**, efficiently constraining the search space by redirecting from verified prefix steps. To ensure training stability, STRIDE maintains **outcome-only reward grounding**, where learning signals are strictly tied to final correctness, shielding the model from the harmful gradient noise that unreliable step-level signals introduce [13]. By unlocking the information bandwidth of process supervision, STRIDE enables LLMs to overcome reasoning plateaus through guided self-correction rather than exhaustive sampling.

The main contributions of this work are as follows:

- **Paradigm Shift to High-Bandwidth Feedback:** We first propose shifting process supervision toward high-bandwidth with learnable stepwise language feedback to unlock richer training signals.
- **The STRIDE Framework:** We introduce an interleaved three-phase co-training schedule that incorporates a *Multi-Point Redirection* strategy. This approach utilizes verified semantic anchors to efficiently constrain the reasoning exploration space at verified prefix steps.

Table 1: Comparison of STRIDE and closely related method categories across four key design dimensions. Policy update: whether the method updates the base policy (vs. inference-only). Language feedback: whether language-form feedback is directly utilized in the training loop. Learnable verifier: whether a dedicated verifier is jointly optimized with the generator via RL. RL-based training: whether RL is used for policy optimization.

Method	Policy update	Language feedback	Learnable verifier	RL-based training
Outcome-based RL [2, 20]	✓	✗	✗	✓
Inference-time Search [21, 17]	✗	✗	✗	N/A
Inference-time Verifier Search [22, 12]	✗	✓ (inference)	✗	N/A
RL w/ Language Scaffold [23]	✓	✗ (distilled)	✗	✓
Critique-based RL [13, 14, 24]	✓	✓ (frozen/self)	✗	✓
SFT Error-Corrective [16, 15]	✓	✓ (SFT)	✗	✗ (SFT)
Co-training w/ Scalar [18]	✓	✗ (scalar)	✓	✓
STRIDE	✓	✓ (co-trained)	✓	✓

- **Superior Empirical Performance:** We demonstrate that STRIDE significantly outperforms reward-based baselines across diverse reasoning benchmarks, achieving consistent gains at minimal additional training cost: Phase III accounts for only 1/13 of the total training schedule yet delivers the decisive capability to learn from previously unsolvable problems.

2 Related Work

RLVR and Process Supervision. RLVR has demonstrated significant efficacy in enhancing the reasoning performance of LLMs. Early paradigms primarily rely on Outcome-based Reward Models (ORMs) [3], where the model is optimized using a terminal reward signal derived from ground-truth correctness [1, 5]. While ORMs provide an unbiased supervision signal, they suffer from severe credit assignment challenges, as the model receives no feedback on which specific steps in a multi-step trajectory led to the final success or failure. To mitigate this, recent research shifted toward Process Supervision, introducing Process Reward Models (PRMs) that assign scalar scores to individual reasoning steps [9, 25, 11, 10, 26, 27]. However, despite their density, these PRMs remain confined to a unidimensional scalar space. As discussed in Section 3.2, compressing high-dimensional logical reasoning into a single numerical value creates an information bottleneck, leading to representational collapse where distinct error modes become indistinguishable [10]. Consequently, while PRMs improve credit assignment, they lack the semantic bandwidth necessary to guide the model through complex logical redirections, a gap that **STRIDE** aims to fill.

Generative Verification and Language Feedback. In the context of LLMs, a growing body of work explores the use of generative verification and language feedback. Unlike discriminative models, generative verifiers provide feedback in the form of natural language critiques, which offer higher informational density [28, 29, 30]. This paradigm shift is motivated by the observation that LLMs often possess latent knowledge of their errors that cannot be fully expressed through a single numerical score [31]. Existing approaches generally fall into two categories. First, Inference-time Refinement methods [32, 33], leverage linguistic feedback to iteratively correct reasoning paths during the decoding stage. However, these methods primarily focus on improving a single instance at test time rather than updating the underlying policy. Second, Alignment via Feedback methods [34, 35, 14, 24, 36, 37, 38], attempt to internalize feedback during the training phase. However, a major challenge in this area is the instability of linguistic signals: without a robust grounding mechanism, generative feedback can lead to *hallucinated gradients* where the model optimizes toward incorrect critiques [13]. **STRIDE** distinguishes itself by integrating generative verification directly into an interleaved RL training loop and employing an outcome-only reward, ensuring that high-bandwidth language feedback leads to stable and verifiable policy improvements.

Refining, Rethinking and Trajectory Redirection. The concept of refining or rethinking a trajectory after an initial attempt is a well-established strategy for solving complex reasoning tasks. Conventional methods typically employ Inference-time Search [39, 40], such as Tree-of-Thought (ToT) [21], Backtracking Search [17], and SWE-Search [41], which explore multiple reasoning branches or

solution paths to identify valid outcomes. While effective during decoding, these approaches do not inherently improve the model's base policy. In the training context, methods like STaR [42], Quiet-STaR [43], and DOTS [44] focus on self-taught reasoning by fine-tuning on successful *rationale* trajectories. However, these frameworks often ignore the valuable signal present in failed attempts, treating them as simple negative samples rather than opportunities for learning error correction [45]. STRIDE draws inspiration from this lineage but introduces a fundamental shift through Guided Trajectory Redirection. Unlike Re-sampling, Re-Reading [46], or Re-solving [47] strategies that often restart the reasoning process from scratch, our Multi-Point Redirection leverages the verifier's localization of the First Point of Failure (FPF) to pinpoint where the logic diverged. This allows the model to re-explore only the necessary sub-trees of the reasoning space, significantly constraining the exploration effort compared to unguided trial-and-error [31]. Furthermore, by providing explicit in-context language feedback at the redirection anchors, **STRIDE** transforms the *refine* process from a stochastic search into a directed evolution of the reasoning policy.

Positioning STRIDE in the Landscape. Table 1 provides a structured comparison of **STRIDE** against closely related methods across four dimensions. STRIDE is the only approach that simultaneously satisfies all four properties, unified by a single design principle: shifting process supervision from scalar rewards to in-context language feedback produced by a co-trained verifier.

3 Preliminaries

3.1 The Generator-Verifier Framework

The Generator-Verifier (GV) framework [18] uses RL to concurrently train a generative *Generator* G_θ and a generative *Verifier* V_ϕ. Given a query x, the generator produces a reasoning path $y = (z_1, \ldots, z_T)$ of sequential thought steps; the verifier assesses each step and produces a language verification sequence $v = (v_1, \ldots, v_T) = V_\phi(x, y)$, from which step-level correctness labels are parsed. Both models are trained via RLVR using only the outcome signal (whether the final judgment $\hat{c}_O$ matches the ground truth c^*_O), with no access to intermediate step annotations.

To update the generator, prior work combines step-level and outcome-based advantages via a decaying coefficient α. However, this design carries a critical instability: step-level rewards derived from the verifier are difficult to align with the ground-truth outcome signal [18], leading to unreliable gradient updates. Formal definitions and the full optimization objective are provided in Appendix B.

3.2 The Information Bottleneck of Scalar Rewards

In the GV framework, the verifier compresses a high-dimensional reasoning path $y = (z_1, \ldots, z_T) \in \mathcal{Y}$ into a unidimensional scalar reward sequence $r \in \mathbb{R}$. Since $\dim(\mathcal{Y}) \gg \dim(\mathbb{R})$, this mapping is inherently **many-to-one**: paths with fundamentally different logical errors may receive identical scalar values, providing the generator no semantic direction to identify *where* or *why* a mistake occurred. A formal Rate-Distortion analysis showing that scalar rewards are fundamentally limited in information bandwidth is provided in Appendix B.2.

The fundamental limitation of scalar rewards lies in this *information-theoretic disparity* between high-dimensional reasoning and unidimensional rewards, rendering the error-correction process ill-posed. As illustrated in Appendix B.3, two semantically distinct errors collapse to the same scalar value, while language feedback restores the missing semantic dimension. To handle this challenge, we introduce the STRIDE framework in the following section.

4 Methodology

In this section, we present **STRIDE**, a novel three-phase training framework that effectively trains, generates, and leverages verification for stepwise redirection, overcoming the performance plateau of LLM with purely scalar reward training. The core idea of STRIDE is to shift the process supervision paradigm from scalar rewards to stepwise language feedback, enabling the generator to improve with informative stepwise language feedback during the training process.

4.1 Overview of STRIDE Framework

The STRIDE framework is a unified, three-phase training system designed to evolve the reasoning capabilities of LLMs from simple outcome matching to active, guided redirection. Unlike traditional co-training paradigms, STRIDE executes these phases in an interleaved manner with a scheduled cadence (e.g., a $9 : 3 : 1$ ratio for G training, V training, and G redirection), ensuring a stable progression from base policy optimization to complex error rectification.

As depicted in Figure 1, the framework orchestrates the interaction between the Generator G_θ and the Verifier V_ϕ through the following stages:

Phase I: Base Policy Optimization. In this foundational stage, the generator G_θ is trained using Group Relative Policy Optimization (GRPO) based on outcome rewards c_O^*. This phase occupies the largest portion of the training cycle (9/13 of the schedule), focusing on building the fundamental ability of the model to generate coherent reasoning trajectories y that reach the correct final answer.

Phase II: Generative Verifier Optimization. Following the base generator updates, the verifier V_ϕ is trained in an outcome-based RLVR manner to provide generative verification $v = (v_1, \ldots, v_T)$. By approximating the outcome-based supervision of overall correctness prediction (whether $\hat{c}_O = c_O^*$), the verifier learns to decompose the terminal signal into stepwise language verification $v_t \in \mathcal{V}^*$.

Phase III: Guided Trajectory Redirection. This stage drives Generator G_θ to overcome reasoning stagnation by leveraging V_ϕ as a contextual navigator. For samples where G_θ fails and V_ϕ correctly identifies the error, we construct a redirection pipeline. In this phase, the generator is trained specifically to rectify its reasoning path y before the first point of failure t^*, using the verifier's guidance v_{t^*} as an in-context trigger. Crucially, this phase uses a pure redirection distribution without mixing Phase I samples to maximize the gradient focus on error correction and avoid the off-policy issues [48, 13].

4.2 Generative Verification and Stepwise Error Localization

This section formalizes how the verifier V_ϕ transitions from a passive reward model to an active error-localization tool.

Structured Verification Generation. For each step z_t in a trajectory y, the verifier V_ϕ decodes a sequence of language verification v_t. This generative process is modeled as:

$$(v_1, v_2, \ldots, v_T) = V_\phi(x, y)$$

The resulting sequence $v = (v_1, \ldots, v_T)$ provides a high-fidelity audit trail of the generator's reasoning process.

The Triggering Function for Redirection. To automate the redirection process, we define a Triggering Function $\tau(v_t)$ that parses the semantic content of each verification step:

$$\tau(v_t) = \begin{cases} 0, & \text{if } v_t \text{ identifies a logical or arithmetic fallacy} \\ 1, & \text{otherwise} \end{cases}$$

The system then identifies the First Point of Failure (FPF), denoted as t^*:

$$t^* = \min\{t \mid \tau(v_t) = 0\}$$

This t^* serves as the temporal anchor for redirection, ensuring that the generator's redirection starts at where the logic deviated, enabling precise and contextually relevant corrections.

4.3 Guided Trajectory Redirection

In Phase III, STRIDE transforms flawed reasoning paths into high-value training signals by Guided Trajectory Redirection. Instead of treating the First Point of Failure (FPF) as a terminal error, we leverage it as a sign for parallel path reconstruction.

Multi-Point Redirection Strategy. Given an initial reasoning trajectory $y = (z_1, \ldots, z_T)$ where the verifier V_ϕ localizes the first point of failure at index t^*, we do not merely rectify the specific step z_{t^*}. Instead, we define a set of anchor points encompassing the entire prefix up to the failure:

$\mathcal{A} = \{t \mid 1 \leq t \leq t^*\}$. For each (x, y), we simultaneously construct t^* distinct redirection samples. This dense sampling strategy addresses three critical challenges in reasoning alignment: (i) Deep Error Attribution: It accounts for *latent drift*, where the terminal fallacy at t^* is a downstream consequence of a suboptimal (though not yet incorrect) choice at $t < t^*$. (ii) Verification Noise Tolerance: It mitigates the inherent uncertainty of V_ϕ. By re-sampling from steps preceding the detected error, the system remains robust even if the verifier fails to pinpoint the exact step of deviation. (iii) Exploration Density: It encourages the generator to explore alternative valid reasoning paths, effectively balancing error correction with path diversification.

Context Construction. For each anchor $t \in \mathcal{A}$, we construct a unique redirection context $S^{(t)}_{redirect}$. The semantics of the guidance are conditioned on the anchor's position relative to the failure point:

$$S^{(t)}_{redirect} = (x, z_1, v_1, \ldots, z_{t-1}, v_{t-1}, \text{Instr}^{(t)}) \tag{1}$$

where the redirection instruction $\text{Instr}^{(t)}$ is defined with subtle but crucial differences: (i) Rectification Prompt (if $t = t^*$): The verifier provides v_{t^*} identifying the specific error. The generator is prompted to rectify the fallacy and resume reasoning. (ii) Exploration Prompt (if $t < t^*$): Since step z_t was deemed correct but still led to a failed outcome, the generator is prompted to continue the reasoning from this valid prefix step, implicitly encouraging the discovery of more robust or efficient paths. Detailed prompt templates are provided in Appendix C.

Training on Pure Redirection Distributions. In Phase III, we update G_θ solely on redirected samples $\{y^k_{redirect}\}_{k=1}^K$ by GRPO with rollout batch size K. By maintaining a non-mixed distribution from Phase I, we specialize the policy in *listening* to contextual guidance for redirection. Following our robust training principle, the advantage $\hat{A}_k$ is calculated strictly based on the outcome correctness c^* of each redirected trajectory:

$$\hat{A}_k = \frac{c^*_k - \text{mean}(c^*_1, \ldots, c^*_K)}{\text{std}(c^*_1, \ldots, c^*_K) + \epsilon} \tag{2}$$

If the verifier's guidance is hallucinated or logically wrong, the resulting batch trajectories $\{y^k_{redirect}\}_{k=1}^K$ are incorrect still, all yielding $\hat{A}_k = 0$ for these samples. This ensures that while we push the reasoning ceiling via correct guidance, we do not pollute the policy with verifier-induced noise, which is unavoidable if directly using scalar rewards from V_ϕ for advantage computation [18].

In summary, STRIDE establishes an interleaved three-phase training paradigm, as illustrated in Algorithm 1, to harmonize reasoning and verification. The core idea lies in shifting the process supervision paradigm from unidimensional scalar rewards to high-bandwidth stepwise language feedback, effectively breaking the information bottleneck in complex reasoning tasks. Benefiting from the outcome-only reward, STRIDE maintains high robustness against verifier hallucinations by ensuring only successful redirections contribute to policy updates. Although Phase III represents a small fraction of the training cycle, it serves as the decisive engine for transcending reasoning ceilings by fostering sparse but vital *breakthrough* samples that enable the model to overcome the performance plateau, as demonstrated in our experiments.

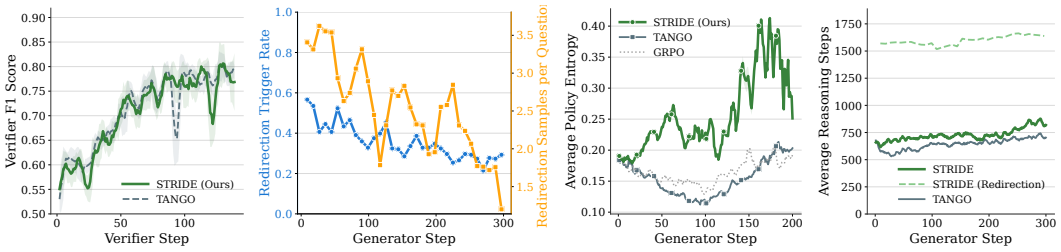
5 Experiments

Models and Baselines. To evaluate the efficacy of STRIDE, we employ two series of generator-verifier pairs: (1) Qwen, using Qwen2.5-Math-7B [49] as the generator and Qwen2.5-7B [50] as the verifier; (2) Llama, using Llama-3.1-8B [51] for both roles. We compare our method against two primary classes of baselines: (1) Outcome-based RL: A standard RLVR approach using only terminal ground-truth rewards via GRPO [2]. (2) TANGO [18]: The state-of-the-art co-training framework that utilizes the verifier to provide scalar step-level rewards alongside outcome rewards. This setup allows us to directly measure the gain from shifting from scalar-based process supervision to our proposed stepwise language feedback.

Datasets and Benchmarks. We conduct evaluation on five competition-level mathematical benchmarks: AIME 2024/2025 [52, 53], AMC 2023 [54], MATH-500 [9], and OlympiadBench [55]. To assess general reasoning and robustness across domains, we further include BoardgameQA (logic) [56], CRUXEval (code) [57], StrategyQA (commonsense) [58], and TableBench (tabular reasoning) [59]. These benchmarks represent a comprehensive testbed for complex, multi-step logical deduction.

Table 2: **Comprehensive performance comparison with prior methods** on mathematical and general reasoning benchmarks. **STRIDE** achieves state-of-the-art performance among 7B/8B-scale models across both domains. For mathematical reasoning, results for most baseline models are sourced from their respective original papers or the prior works [60, 61]. We adopt the performance of reproduction of PRIME [11] reported in [18].

Model	Mathematical Reasoning						Out-of-Domain Reasoning				
	MATH 500	AIME 2024	AIME 2025	AMC 2023	Olympiad Bench	Avg.	BGQA	CRUX Eval	Strategy QA	Table Bench	Avg.
Frontier LLMs											
GPT-4o [62]	76.6	9.3	-	47.5	43.3	-	-	-	-	-	-
Claude3.5-Sonnet [63]	78.3	16.0	-	-	-	-	-	-	-	-	-
o1-preview [64]	85.5	44.6	-	90.0	-	-	-	-	-	-	-
o1-mini [64]	90.0	56.7	-	95.0	65.3	-	-	-	-	-	-
Open-sourced reasoning LLMs (large)											
Llama-3.1-70B-Instruct [51]	68.0	13.3	-	42.5	29.4	-	58.3	59.6	88.8	34.2	-
OpenMath2-Llama3.1-70B [65]	71.8	13.3	-	45.0	30.1	-	68.7	35.1	95.6	46.8	-
NuminaMath-72B-CoT [66]	64.0	3.3	-	70.0	32.6	-	-	-	-	-	-
Qwen2.5-Math-72B-Instruct [49]	82.6	23.3	-	70.0	49.0	-	-	-	-	-	-
QwQ-32B-Preview [67]	90.6	50.0	33.3	77.5	61.2	62.5	71.1	65.2	88.2	51.5	69.0
Open-sourced reasoning LLMs (small)											
Llama-3.1-8B-Instruct [51]	51.9	3.3	3.3	22.5	15.1	19.2	50.3	38.5	**92.2**	32.4	53.4
OpenMath2-Llama3.1-8B [65]	67.8	6.7	3.3	37.5	28.9	28.8	49.0	11.1	84.4	34.2	44.7
Qwen2.5-7B-Instruct [50]	75.5	10.0	6.7	52.5	35.5	36.0	53.0	**58.1**	91.3	43.2	61.4
Qwen2.5-Math-7B-Instruct [49]	83.6	16.7	10.0	62.5	41.6	42.9	51.3	28.0	85.3	36.2	50.2
rStar-Math-7B [60]	78.4	**26.7**	-	47.5	**47.1**	-	-	-	-	-	-
Eurus-2-7B-PRIME [11]	80.4	**26.7**	13.3	60.0	43.7	44.8	-	-	-	-	-
Ours											
STRIDE-Llama-8B	70.4	13.3	10.0	50.3	36.0	36.0	50.2	46.0	88.2	32.3	54.2
STRIDE-Qwen-7B	**84.6**	**26.7**	**23.3**	**75.0**	46.1	**51.1**	**66.8**	57.0	92.0	**43.8**	**64.9**



(a) Verifier F_1 (b) Guidance Efficiency (c) Policy Entropy (d) Reasoning Depth

Figure 2: **STRIDE training dynamics.** (a) Fair Comparison Validated: STRIDE and TANGO share near-identical verifier F_1 trajectories, confirming the performance gap originates from *how* feedback is utilized (language guidance vs. scalar reward). (b) Continuous Breakthrough on Hard Problems: The declining redirection error rate shows the generator progressively conquers previously unsolvable instances, with the verifier pinpointing failures at ever-earlier steps as training matures. (c) Sustained Exploration: STRIDE maintains consistently higher policy entropy than TANGO, demonstrating that language guidance prevents premature convergence and sustains a richer exploration landscape throughout training. (d) Emergence of Deep Reasoning: Redirected trajectories grow substantially longer during Phase III, reflecting qualitatively richer reasoning chains that the generator could not produce through independent sampling alone.

Evaluation Metrics. We employ the zero-shot Pass@1 accuracy as our primary metric, using greedy decoding for all models. Furthermore, to specifically isolate the impact of our Phase III mechanism, we introduce the Correction Success Rate (CSR), which measures the probability of a generator successfully reaching the correct outcome after receiving a redirection trigger compared to its initial failed attempt. For the verifier, we measure its F1 score on verification accuracy to ensure its reliability. More implementation details are provided in the Appendix D.

Table 3: **Comparison of STRIDE with vanilla RLVR and co-training baselines**. By shifting process supervision from unidimensional rewards to stepwise redirection, **STRIDE** significantly outperforms GRPO and TANGO on both model series across mathematical reasoning and out-of-domain reasoning benchmarks. The baseline results of Qwen2.5 series are adopted from [18]. All models are trained for 200 generator steps with identical settings.

	Mathematical Reasoning						Out-of-Domain Reasoning				
Model	**MATH500**	**AIME2024**	**AIME2025**	**AMC2023**	**OlympiadBench**	**Avg.**	**BGQA**	**CRUXEval**	**StrategyQA**	**TableBench**	**Avg.**
Qwen2.5-7B-SFT	66.6	3.3	3.3	27.5	28.1	25.8	46.6	44.3	85.9	34.4	52.8
+ GRPO [2]	74.6	13.3	10.0	50.0	36.9	37.0	55.3	48.8	88.1	38.2	57.6
+ TANGO [18]	81.4	20.0	20.0	65.0	43.9	46.1	60.5	51.4	90.0	42.3	61.1
+ **STRIDE** (Ours)	**84.6**	**26.7**	**23.3**	**75.0**	**46.1**	**51.1**	**66.8**	**57.0**	**92.0**	**43.8**	**64.9**
Llama3.1-8B-SFT	57.2	6.6	3.3	42.5	28.0	27.5	46.5	43.5	83.7	27.8	50.4
+ GRPO [2]	66.8	6.6	6.6	47.5	34.2	32.3	48.2	45.5	84.6	30.8	52.3
+ TANGO [18]	69.2	6.6	6.6	50.0	35.0	33.5	49.0	45.3	86.0	28.9	52.3
+ **STRIDE** (Ours)	**70.4**	**13.3**	**10.0**	**50.3**	**36.0**	**36.0**	**50.2**	**46.0**	**88.2**	**32.3**	**54.2**

5.1 Main Results

Overall performance across domains. Table 2 reports zero-shot Pass@1 results on both mathematics and out-of-domain reasoning benchmarks. Across the 7B/8B scale models of various families, STRIDE achieves the strongest overall performance, with consistent gains on nearly all tasks. Crucially, these gains extend beyond math to logic, code, commonsense, and tabular reasoning, indicating that the generalization of STRIDE is not confined to math domain but rather reflects consistent improvements in reasoning capabilities.

Comparison with RLVR and co-training baselines. To isolate the effect of replacing scalar step rewards with stepwise redirection, we compare STRIDE against vanilla outcome-based RLVR (GRPO) and the scalar-reward co-training baseline TANGO under identical settings. As shown in Table 3, STRIDE consistently outperforms both baselines on the two series across the two benchmark groups. These results validate our claim: high-bandwidth linguistic guidance provides more actionable supervision than unidimensional scalar rewards, especially for multi-step reasoning where credit assignment is challenging.

Training dynamics and the role of guidance. Figure 2 provides a closer look into the training process across four dimensions. First, both STRIDE and TANGO exhibit near-identical verifier F_1 growth curves (a), establishing that the two systems operate with comparable verifier quality throughout co-training. This isolates the performance advantage of STRIDE to the *paradigm difference*: language guidance versus scalar step rewards, rather than a superior verifier. Second, the simultaneous decline in redirection trigger rate and per-question sample yield (b) reveals maturing guidance efficiency: as training progresses, the generator fails on fewer problems, and the verifier localizes errors at increasingly earlier steps, reducing the cost and expanding the scope of each redirection cycle. Third, this high-bandwidth supervision sustains consistently higher policy entropy in STRIDE than in TANGO (c), indicating that stepwise language feedback preserves a broader exploration landscape and actively prevents the representational collapse observed in scalar-reward baselines. Finally, the redirection phase fosters a qualitative shift in reasoning depth (d): trajectories generated under verifier guidance are substantially longer and more structurally complex than those produced by independent sampling, reflecting the emergence of deliberate, multi-step reasoning patterns that scalar supervision cannot elicit. These dynamics confirm the benefits of learnable language feedback.

5.2 Ablation Study

Core findings. Table 4 decomposes Phase III into two orthogonal choices: (i) *anchor selection* (single-point t^* vs. multi-point $[1, t^*]$) and (ii) *supervision bandwidth* (no guidance vs. stepwise linguistic guidance). Two conclusions stand out. First, both factors are independently beneficial: multi-point anchors improve robustness to imperfect failure localization and capture earlier suboptimal decisions, while linguistic guidance provides actionable directions that scalar feedback cannot convey. Second, their combination yields the largest gain, indicating that effective trajectory redirection requires *both* reliable restart states and high-bandwidth corrective signals. Finally, we find that injecting scalar step rewards into STRIDE can hurt performance compared to our outcome-only design, suggesting that misaligned step rewards may introduce noise and destabilize learning. Critically, STRIDE achieves a

CSR of 6.8% on zero-pass-rate problems, where all scalar-reward baselines yield identically zero gradient signal, directly quantifying Phase III's role as a breakthrough mechanism that converts otherwise unsolvable problems into productive training signals.

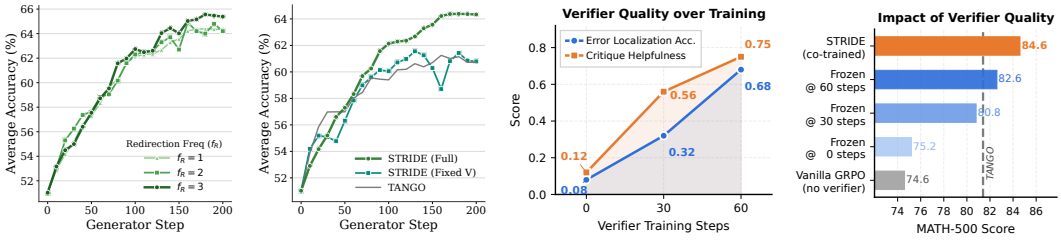
Sensitivity and Verifier Quality Assessment. Figure 3a shows that higher redirection frequency f_R yields marginal but consistent Pass@1 gains, suggesting Phase III acts as a "rare-but-high-value" correction operator whose returns saturate once easy-to-correct failures are exhausted. Figure 3b further confirms that co-training the verifier with the generator is crucial: the fixed-verifier variant underperforms co-trained STRIDE, as a frozen verifier cannot adapt its localization to the generator's evolving error distribution.

Table 4: **Ablation of Redirection Strategies.** This table evaluates the contribution of anchor selection and linguistic guidance on MATH-500 with performance and CSR. *Single-Point denotes no linguistic guidance with only a single-point anchor at t^*. *STRIDE indicates including step-level reward for training. Results confirm that combining multi-point anchors with high-bandwidth guidance yields the most significant performance breakthrough.

Configuration	Anchor	Guidance	MATH	CSR
(i) Standard RLVR	None	None	74.6	-
(ii) *Single-Point	t^* only	None	74.8	1.2
(iii) Single-Point	t^* only	Ling.	78.5	3.4
(iv) Multi-Point	$[1, t^*]$	None	79.1	3.8
(v) *STRIDE	$[1, t^*]$	Ling.	82.6	5.2
(vi) **STRIDE**	$[1, t^*]$	**Ling.**	**84.6**	**6.8**

To directly characterize verifier reliability, Figure 3c tracks step-level quality over training using GPT-5 as an automatic judge, measuring error localization accuracy and critique helpfulness independently. Both metrics improve substantially (localization: $0.08 \rightarrow 0.68$; helpfulness: $0.12 \rightarrow 0.75$), confirming that outcome-level RL supervision is sufficient to develop reliable process-level capabilities, analogous to how chain-of-thought reasoning emerges from outcome reward alone.

Figure 3d further isolates the impact of verifier quality by comparing STRIDE against variants that freeze the verifier at different training stages. Performance degrades monotonically as verifier quality drops, yet even the weakest frozen verifier (0 steps) still outperforms vanilla GRPO (75.2 vs. 74.6). This confirms that STRIDE's Multi-Point Redirection Strategy provides built-in robustness to imperfect verification: even when localization is unreliable, the outcome-only reward ensures that only successful redirections contribute gradient updates, guaranteeing non-negative improvement by design. This robustness also demonstrates that paradigm shifts toward language feedback can yield benefits from imperfect verifiers, while noisy of scalar rewards can degrade performance, alleviating concerns about verifier reliability and effectively leverage verifier feedback for policy improvement, even when the verifier is still learning in early training stages.



(a) Redirection Frequency (f_R)

(b) Impact of Co-training

(c) Verifier Quality over Training

(d) Impact of Verifier Quality

Figure 3: **Sensitivity Analysis and Verifier Quality Assessment.** (a) Higher redirection frequency ($f_R = 1, 2, 3$) yields marginal but consistent Pass@1 gains. (b) Co-training the verifier and generator is important for maintaining high-quality guidance signals. (c) Verifier step-level localization accuracy and critique helpfulness both improve steadily over training, confirming that outcome-level supervision yields reliable process-level capabilities. (d) Higher verifier quality at the time of freezing yields consistently better MATH-500 performance, while co-training (STRIDE) achieves the strongest result, validating the importance of joint optimization.

6 Conclusion

We propose STRIDE, an interleaved three-phase training framework that shifts process supervision in RLVR from sparse scalar rewards to high-bandwidth stepwise language feedback. By co-training a generative verifier with outcome-only rewards and using its language critiques to trigger guided trajectory redirection, STRIDE alleviates the information bottleneck of scalar supervision and turns failed trajectories into efficient learning signals. Extensive experiments across mathematical and out-of-domain reasoning benchmarks demonstrate consistent improvements over outcome-only RLVR and scalar-reward co-training baselines, while ablations confirm the importance of multi-point anchors and linguistic guidance. A promising direction for future work is to extend the redirection mechanism to more general agentic settings, which involve multi-step tool calls.

Acknowledgments

This research is supported by the RIE2025 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) (Award I2301E0026), administered by A*STAR, as well as supported by Alibaba Group and NTU Singapore through Alibaba-NTU Global e-Sustainability CorpLab (ANGEL).

References

[1] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

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

[3] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

[4] Wenkai Fang, Shunyu Liu, Yang Zhou, Kongcheng Zhang, Tongya Zheng, Kaixuan Chen, Mingli Song, and Dacheng Tao. Serl: Self-play reinforcement learning for large language models with limited data. In *Advances in Neural Information Processing Systems*, 2025.

[5] 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.

[6] Shunyu Liu, Wenkai Fang, Zetian Hu, Junjie Zhang, Yang Zhou, Kongcheng Zhang, Rongcheng Tu, Ting-En Lin, Fei Huang, Mingli Song, Yongbin Li, and Dacheng Tao. A survey of direct preference optimization. *arXiv preprint arXiv:2503.11701*, 2025.

[7] Guanyi Qin, Junjie Zhang, Chunming He, Yibing Fu, Jie Liang, Tianhe Wu, and Lei Zhang. Tool-IQA: Augmenting image quality assessment with simple tools. *arXiv preprint arXiv:2606.16082*, 2026.

[8] Guanyi Qin, Xiaozhen Wang, Zhu Zhuo, Chang Han Low, Yuancan Xiao, Yibing Fu, Haofeng Liu, Kai Wang, Chunjiang Li, and Yueming Jin. SurGo-R1: Benchmarking and modeling contextual reasoning for operative zone in surgical video. *arXiv preprint arXiv:2602.21706*, 2026.

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

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

[11] 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.

[12] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

[13] Xiaoying Zhang, Hao Sun, Yipeng Zhang, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. Critique-grpo: Advancing llm reasoning with natural language and numerical feedback. *arXiv preprint arXiv:2506.03106*, 2025.

[14] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*, 2022.

[15] Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, et al. Enhancing llm reasoning via critique models with test-time and training-time supervision. *arXiv preprint arXiv:2411.16579*, 2024.

[16] Zhuoshi Pan, Yu Li, Honglin Lin, Qizhi Pei, Zinan Tang, Wei Wu, Chenlin Ming, H Vicky Zhao, Conghui He, and Lijun Wu. Lemma: Learning from errors for mathematical advancement in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11615–11639, 2025.

[17] Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Ding-Chu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. Step back to leap forward: Self-backtracking for boosting reasoning of language models. *arXiv preprint arXiv:2502.04404*, 2025.

[18] Kaiwen Zha, Zhengqi Gao, Maohao Shen, Zhang-Wei Hong, Duane S. Boning, and Dina Katabi. Rl tango: Reinforcing generator and verifier together for language reasoning. In *Advances in Neural Information Processing Systems*, 2025.

[19] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[20] Xiangyuan Xue, Yifan Zhou, Guibin Zhang, Zaibin Zhang, Yijiang Li, Chen Zhang, Zhenfei Yin, Philip Torr, Wanli Ouyang, and Lei Bai. Comas: Co-evolving multi-agent systems via interaction rewards. *arXiv preprint arXiv:2510.08529*, 2025.

[21] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

[22] Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

[23] Weijie Shi, Yanxi Chen, Zexi Li, Xuchen Pan, Yuchang Sun, Jiajie Xu, Xiaofang Zhou, and Yaliang Li. R^3L: Reflect-then-retry reinforcement learning with language-guided exploration, pivotal credit, and positive amplification. *arXiv preprint arXiv:2601.03715*, 2026.

[24] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, 2025.

[25] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Step-level value preference optimization for mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP*, 2024.

[26] 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 *Annual Meeting of the Association for Computational Linguistics*, pages 9426–9439, 2024.

[27] Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, et al. Versaprm: Multi-domain process reward model via synthetic reasoning data. *arXiv preprint arXiv:2502.06737*, 2025.

[28] Junjie Zhang, Guozheng Ma, Shunyu Liu, Haoyu Wang, Jiaxing Huang, Ting-En Lin, Fei Huang, Yongbin Li, and Dacheng Tao. A simple" motivation" can enhance reinforcement finetuning of large reasoning models. In *International Conference on Learning Representations*, 2026.

[29] Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D Chang, and Prithviraj Ammanabrolu. Critique-out-loud reward models. *arXiv preprint arXiv:2408.11791*, 2024.

[30] Rong-Cheng Tu, Wenhao Sun, Zhao Jin, Jingyi Liao, Jiaxing Huang, and Dacheng Tao. Spagent: Adaptive task decomposition and model selection for general video generation and editing. *IEEE Transactions on Image Processing*, 2026.

[31] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024.

[32] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

[33] Kechi Zhang, Zhuo Li, Jia Li, Ge Li, and Zhi Jin. Self-edit: Fault-aware code editor for code generation. *arXiv preprint arXiv:2305.04087*, 2023.

[34] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

[35] Yuhua Jiang, Yuwen Xiong, Yufeng Yuan, Chao Xin, Wenyuan Xu, Yu Yue, Qianchuan Zhao, and Lin Yan. Pag: Multi-turn reinforced llm self-correction with policy as generative verifier. *arXiv preprint arXiv:2506.10406*, 2025.

[36] Zhihui Xie, Jie Chen, Liyu Chen, Weichao Mao, Jingjing Xu, and Lingpeng Kong. Teaching language models to critique via reinforcement learning. In *International Conference on Machine Learning*, 2025.

[37] Xiaoyuan Liu, Tian Liang, Zhiwei He, Jiahao Xu, Wenxuan Wang, Pinjia He, Zhaopeng Tu, Haitao Mi, and Dong Yu. Trust, but verify: A self-verification approach to reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2505.13445*, 2025.

[38] Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, et al. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for open-ended llm reasoning. In *International Conference on Machine Learning*, 2026.

[39] Junjie Zhang, Rushuai Yang, Shunyu Liu, Ting-En Lin, Fei Huang, Yi Chen, Yongbin Li, and Dacheng Tao. Supervised optimism correction: Be confident when llms are sure. *Annual Meeting of the Association for Computational Linguistics Findings*, 2025.

[40] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters for reasoning. In *International Conference on Learning Representations*, 2025.

[41] Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. *arXiv preprint arXiv:2410.20285*, 2024.

[42] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

[43] Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

[44] Murong Yue, Wenlin Yao, Haitao Mi, Dian Yu, Ziyu Yao, and Dong Yu. DOTS: Learning to reason dynamically in LLMs via optimal reasoning trajectories search. In *International Conference on Learning Representations*, 2025.

[45] Kongcheng Zhang, Qi Yao, Shunyu Liu, Wenjian Zhang, Min Cen, Yang Zhou, Wenkai Fang, Yiru Zhao, Baisheng Lai, and Mingli Song. Replay failures as successes: Sample-efficient reinforcement learning for instruction following. In *International Conference on Machine Learning*, 2026.

[46] Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, Jian-Guang Lou, and Shuai Ma. Re-reading improves reasoning in large language models. In *Conference on Empirical Methods in Natural Language Processing*, 2024.

[47] Pinzheng Wang, Shuli Xu, Juntao Li, Yu Luo, Dong Li, Jianye Hao, and Min Zhang. Re^2: Unlocking LLM reasoning via reinforcement learning with re-solving. In *International Conference on Learning Representations*, 2026.

[48] Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025.

[49] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[50] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[51] 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.

[52] AI-MO. Aime 2024, 2024.

[53] OpenCompass. Aime 2025, 2025.

[54] AI-MO. Amc 2023, 2024.

[55] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng 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 *Annual Meeting of the Association for Computational Linguistics*, 2024.

[56] Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. Boardgameqa: A dataset for natural language reasoning with contradictory information. In *Advances in Neural Information Processing Systems*, 2023.

[57] Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I Wang. Cruxeval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*, 2024.

[58] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. In *Transactions of the Association for Computational Linguistics*. MIT Press, 2021.

[59] Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, et al. Tablebench: A comprehensive and complex benchmark for table question answering. In *AAAI Conference on Artificial Intelligence*, 2025.

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

[61] Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search. *arXiv preprint arXiv:2502.02508*, 2025.

[62] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[63] Anthropic. Claude 3.5 sonnet, 2024.

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

[65] Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024.

[66] Edward Beeching, Shengyi Costa Huang, Albert Jiang, Jia Li, Benjamin Lipkin, Zihan Qina, Kashif Rasul, Ziju Shen, Roman Soletskyi, and Lewis Tunstall. Numinamath 72b cot, 2024.

[67] Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown, 2024.

[68] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2 edition, 2006.

[69] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

Appendix

Table of Contents

A STRIDE Interleaved Training Algorithm **16**

B Formal Preliminaries **16**

B.1 Generator-Verifier Framework 16

B.2 Information Bottleneck: Rate-Distortion Analysis 16

B.3 Illustrative Example: Representational Collapse 17

C Prompt Templates & Redirection Instructions **17**

C.1 Phase I Generator Reasoning Template 17

C.2 Phase II Generative Verification Template 18

C.3 Phase III Redirection Instructions 18

D Implementation Details **19**

E Additional Experimental Results **20**

E.1 Compute Fairness Analysis 20

E.2 Orthogonality with Inference-Time Scaling 20

E.3 Results with Variance: Comparison with Baselines 20

F Limitations **21**

G Case Studies **21**

G.1 Case Study I: Breaking Representational Collapse 21

G.2 Case Study II: Multi-Point Redirection Produces Diverse Strategies 21

G.3 Case Study III: Latent Drift and Recovery from Pre-Failure Anchors 22

A STRIDE Interleaved Training Algorithm

Algorithm 1 STRIDE Interleaved Training

Initialize: Generator G_θ, Verifier V_ϕ, Iterations N
Set Frequencies: $f_G = 9, f_V = 3, f_R = 1$
for each training cycle $C = 1, \ldots, N$ **do**
  // Phase I: Base Policy Optimization
  repeat
    Sample $y \sim G_\theta(x)$; Update θ via GRPO with outcome c_O^*.
  until run for f_G steps, with Phase II injected every 3 G-steps
  // Phase II: Generative Verifier Optimization
  Sample $y \sim G_\theta(x), v \sim V_\phi(x, y)$; Update ϕ to produce stepwise verification v via GRPO with $r = \mathbb{I}(\hat{c_O} = c_O^*)$
  // Phase III: Guided Trajectory Redirection
  Execute once per cycle ($f_R = 1$):
  1. **Selection:** Filter queries x where G_θ failed all attempts ($c_O^* = 0$) and V_ϕ correctly identified the failure ($c_O = 0$).
  2. **Error Localization:** Identify first point of failure $t^* = \min\{t \mid \tau(v_t) = 0\}$.
  3. **Parallel Reconstruction:** Build $\{S_{redirect}^{(t)}\}_{t=1}^{t^*}$ anchors using prefix $(x, z_{<t}, v_{<t})$.
  4. **Redirection:** Sample $y_{red} \sim G_\theta(S_{redirect})$ and update θ via GRPO with outcome c_O^*.
end for

B Formal Preliminaries

B.1 Generator-Verifier Framework

Generator as Reasoner. Given an input query $x \in \mathcal{X}$, the generator G_θ aims to generate a reasoning path y consisting of a sequence of intermediate thought steps, denoted as $y = (z_1, z_2, \ldots, z_T) \in \mathcal{Y}$. The probability of generating a specific path is factorized as:

$$G_\theta(y|x) = \prod_{t=1}^{T} G_\theta(z_t|x, z_{<t}), \tag{3}$$

where $\mathcal{Y}$ represents the high-dimensional discrete space of natural language reasoning.

Verifier as Evaluator. The verifier V_ϕ assesses the correctness $c_t \in \{0, 1\}$ of each step z_t and the overall correctness $c_O \in \{0, 1\}$ of path y by producing a generative verification sequence $v = (v_1, v_2, \ldots, v_T) = V_\phi(x, y),\ v \in \mathcal{V}^*$, where $\mathcal{V}^*$ denotes the discrete space of natural language. Step-level scores $(\hat{c}_1, \ldots, \hat{c}_T)$ and an overall judgment $\hat{c}_O$ are parsed from this sequence. The verifier is trained via RLVR with outcome-based supervision: the reward is 1 if $\hat{c}_O = c_O^*$, and 0 otherwise.

Optimization with Step-level Reward. In the original GV framework, the generator is updated with a mixed advantage:

$$\hat{A}_t = (1 - \alpha)\hat{A}_{t,\text{out}} + \alpha\hat{A}_{t,\text{step}}, \tag{4}$$

where $\alpha \in (0, 1)$ decays exponentially to shift focus from step-level to outcome-based supervision. The resulting optimization objective is:

$$\mathcal{J}(\theta) = \mathbb{E}_{x,y\sim G_\theta}\left[\sum_{t=1}^{T} \nabla_\theta \log G_\theta(z_t|x, z_{<t}) \cdot \hat{A}_t\right]. \tag{5}$$

While this mitigates supervision sparsity, the step-level reward from the verifier is not easily aligned with the ground-truth outcome signal, creating a critical instability in training.

B.2 Information Bottleneck: Rate-Distortion Analysis

We formalize the information bottleneck of scalar rewards via Rate-Distortion theory. Let the mapping $f : \mathcal{Y} \to \mathbb{R}$ compress reasoning paths to scalar rewards, and let Z denote the latent oracle

guidance representing ground-truth logical steps. According to Rate-Distortion theory [68], the mutual information $I(R; Z)$, quantifying the effective guidance provided by the reward, is bounded by the entropy of the reward signal:

$$I(R; Z) \leq H(R). \tag{6}$$

For a binary scalar reward $r \in \{0, 1\}$, $H(R) \leq 1$ bit. In contrast, the complexity of $\mathcal{Y}$ requires a substantially higher bitrate to uniquely identify and rectify diverse logical fallacies.

Furthermore, since $\dim(\mathcal{Y}) \gg \dim(\mathbb{R})$, f is **heavily many-to-one (non-injective)**: for a given r, the pre-image $f^{-1}(r) = \{y \in \mathcal{Y} \mid f(y) = r\}$ contains a vast number of semantically distinct paths. Paths with fundamentally different logical errors may receive identical scalar values, and the resulting gradient signal is insufficient to distinguish *where* or *why* a mistake occurred.

B.3 Illustrative Example: Representational Collapse

Representational Collapse in Scalar Rewards

Question: Solve for x in $2x + 5 = 13$.
Previous Step: $2x + 5 = 13$ (Initial state).

*The mapping from reasoning paths to rewards is **heavily non-injective**. As shown below, disparate error modes are collapsed into identical scalar values, providing no discriminative guidance.*

Current Erroneous Step	Scalar Reward	Language Feedback
Case A *(Arithmetic Error):* $2x = 13 + 5$	$r = 0.2$ ✗ (Low)	**Incorrect:** Incorrect additive inverse applied during transposition of $+5$.
Case B *(Logical Leap):* $2x + 5 = 13 \implies x = 3$	$r = 0.2$ ✗ (Low)	**Incorrect:** Intermediate algebraic transformations are missing between the premise and result.

Conclusion: Scalar rewards suffer from *representational collapse*, providing a magnitude without direction. Language feedback breaks this bottleneck by restoring the semantic dimensions of feedback.

C Prompt Templates & Redirection Instructions

To ensure a high-bandwidth information flow and maintain structural consistency, STRIDE employs standardized prompt templates for reasoning, verification, and trajectory redirection. This section details the specific instructions provided to the Generator (G_θ) and the Verifier (V_ϕ) across the three training phases.

C.1 Phase I Generator Reasoning Template

In Phase I, the generator is tasked with basic stepwise reasoning. The template enforces a strict `<think>` and `<step>` structure to facilitate error localization in subsequent phases.

Generator Reasoning Template (G_θ)

You are a helpful Assistant that solves mathematical problems step-by-step.
You MUST follow this exact format:

1. Start with a `<think>` section containing your step-by-step reasoning.
2. Inside `<think>`, each distinct logical step MUST be enclosed in its own `<step> </step>` tags.
3. After `<think>`, provide the final answer within `<answer> </answer>` tags, using the \boxed{} format.

... [Detailed Example Omitted] ...
User: prompt
Assistant:

C.2 Phase II Generative Verification Template

The verifier V_ϕ is optimized in Phase II to act as a Contextual Navigator. It decomposes the terminal outcome signal into high-bandwidth linguistic feedback for each reasoning step.

Generative Verification Template (V_ϕ)

System Instruction:
You are a verification assistant specialized in mathematical reasoning. Your task is to carefully evaluate the provided solution step by step, checking for mathematical correctness and logical coherence... You MUST verify EACH `<step>` block found in the Assistant's solution.

Problem: {problem}
Assistant's Solution: {solution}
Step Count: The Assistant's solution contains {generator_step_count} steps.

Task Guidance:
For each of the {generator_step_count} steps, you MUST provide ONE corresponding verification analysis within a `<step>` tag inside the `<step_verification>` section.

Required Format:
`<step_verification>`
`<step>`Step 1 Analysis: [Your reasoning]. \boxed{CORRECT/INCORRECT}`</step>`
...
`</step_verification>`
`<final_verification>`\boxed{CORRECT/INCORRECT}`</final_verification>`

Constraint Checklist:

- Analyze EVERY step individually.
- Provide original reasoning, do not copy solution text.
- Each step analysis must end with exactly one \boxed judgment.
- Output ONLY the specified tags.

C.3 Phase III Redirection Instructions

In Phase III, STRIDE distinguishes between Rectification and Exploration to maximize the utility of failure cases.

- Rectification Prompt: Triggered at the First Point of Failure (FPF, $t = t^*$), providing explicit feedback to correct the detected logical fallacy.
- Exploration Prompt: Triggered at pre-failure anchors ($t < t^*$), encouraging the discovery of alternative robust paths from verified semantic anchors.

Redirection: Rectification (at FPF)

System Instruction:
You are a mathematical Assistant. You will be shown a problem, a previous attempt, and step-by-step verification feedback. Your task is to correct the last step of previous attempt and continue solving the problem.

Problem: {problem}
[Previous Attempt & Feedback]:
`{combined_body}`

Task Guidance:
The verification detected the error in the last step. Now you'll correct the error and continue answering.

Your Solution:

Redirection: Exploration (before FPF)

System Instruction:
You are a mathematical Assistant. You will be shown a problem, a previous attempt, and step-by-step verification feedback. Your task is to continue solving the problem from the last step of previous attempt.

Problem: {problem}
[Previous Attempt & Feedback]:
`{combined_body}`

Task Guidance:
The verification guided the previous steps. Now you'll continue answering.

Your Solution:

D Implementation Details

Our training framework is implemented using the **veRL** [69] distributed RL library and follows the interleaved schedule described in Section 4. We set the cadence ratio to $f_G : f_V : f_R = 9 : 3 : 1$, meaning the verifier updates once every three generator steps to compensate for the relative complexity of policy optimization, while the redirection phase (Phase III) is activated once per full cycle.

Data Preparation and SFT. Consistent with prior art [18], we perform initial Supervised Fine-Tuning (SFT) on the generator using 113K competition-level math prompts from the *Eurus-2-SFT-Data* [11]. To ensure the data quality, reasoning trajectories are generated by prompting `Llama-3.1-70B-Instruct` with a decoding temperature of 0.1 and top-p of 0.5, enforcing the step-by-step reasoning format within `<step>` tags. We use a full-parameter SFT with a learning rate of 5×10^{-6} and a cosine annealing schedule. For Qwen2.5 we use the model after 800 SFT steps, and for Llama-3.1 we use the model after 1,000 SFT steps as the base generator G_θ for subsequent RL training. Notably, the verifier is initialized directly from the base model without prior SFT to demonstrate the framework's capability to bootstrap from a weaker starting point through mutual reinforcement.

Reinforcement Learning Configurations. During the RL stage, we employ 455K question-answer pairs from *Eurus-2-RL-Data* [11]. All training is conducted using the **GRPO** algorithm with a group size of $M = 5$ rollouts per prompt. To prevent early training instability, we implement a **verifier warmup** of 40 steps, allowing the verifier to learn output formatting and basic correctness before providing redirection guidance to the generator. We use a constant learning rate of 1×10^{-6}, a total batch size of 256, and a KL-divergence penalty coefficient $\beta = 0.001$. For the ablation fixed-verifier setting, we freeze V_ϕ after the warmup phase to isolate the effect of joint training.

Stability and Robustness. A key advantage of STRIDE is its inherent stability without exhaustive hyperparameter tuning. Unlike discriminative PRMs that require complex reward mixing, STRIDE maintains **outcome-only reward grounding**. The advantage in Phase III is tied strictly to ground-truth outcome correctness, which shields the policy from potential verifier hallucinations or noisy step-level rewards. This design choice simplifies training dynamics and enhances robustness, as only successful redirections contribute to policy updates in GRPO.

E Additional Experimental Results

E.1 Compute Fairness Analysis

A natural concern is whether STRIDE's gains over GRPO stem from an increased computational budget rather than from the paradigm shift in process supervision. To address this, we compare STRIDE against compute-enhanced GRPO baselines that match STRIDE's total GPU hours (approximately 40 hours on 8×H20 GPUs) by either increasing training steps or enlarging the rollout batch size. As shown in Table 5, both compute-enhanced GRPO variants yield negligible gains over vanilla GRPO (74.4 and 75.2 vs. 74.6), while STRIDE achieves 84.6 under the same budget. This confirms that the improvement originates from the quality of supervision signals, not from additional compute.

Table 5: **Compute Fairness Analysis.** All methods are evaluated under matched GPU hours (≈40 hours on 8×H20 GPUs). Compute-enhanced GRPO baselines with more training steps or larger rollout batches yield negligible gains over vanilla GRPO, confirming that STRIDE's improvement stems from the paradigm shift in process supervision rather than increased computational budget.

Method	Training Time (hr)	RL Steps	MATH-500
Vanilla GRPO	32	200	74.6 ± 0.2
GRPO (more steps)	40	250	74.4 ± 0.2
GRPO (bigger batch)	40	200	75.2 ± 0.5
TANGO	40	200	81.4 ± 0.3
STRIDE	40	200	**84.6** ± 0.2

E.2 Orthogonality with Inference-Time Scaling

STRIDE is a training-time method and is by design orthogonal to inference-time scaling techniques. To empirically verify this, we combine STRIDE with Best-of-N (BoN) sampling and a Process Reward Model (PRM) and compare against GRPO under the same inference budget. Table 6 shows two key findings. First, STRIDE's pass@1 result (84.6) already surpasses GRPO with 16× more inference trajectories (79.6), demonstrating that training-time language guidance provides a fundamentally stronger policy than scalar-reward RL regardless of inference effort. Second, STRIDE further compounds with inference-time scaling: combining STRIDE with BoN(8)+PRM achieves 88.2, substantially outperforming the corresponding GRPO+BoN(8)+PRM baseline (78.2). These results confirm that training-time language guidance and inference-time search address orthogonal bottlenecks and are mutually beneficial.

Table 6: **Orthogonality with Inference-Time Scaling.** STRIDE pass@1 surpasses GRPO with 16× more inference trajectories, and further compounds with Best-of-N sampling and a PRM to achieve 88.2 on MATH-500, demonstrating that training-time language guidance and inference-time scaling are orthogonal and mutually beneficial.

Method	Training Time (hr)	Inference Traj.	MATH-500
GRPO (pass@1)	32	1	74.6 ± 0.2
GRPO + BoN(8) + PRM	32	8	78.2 ± 0.3
GRPO + BoN(16) + PRM	32	16	79.6 ± 0.4
STRIDE (pass@1)	40	1	**84.6** ± 0.2
STRIDE + BoN(8) + PRM	40	8	**88.2** ± 0.3

E.3 Results with Variance: Comparison with Baselines

Table 7 reproduces the main comparison table with standard deviations reported on the two Avg. columns, computed across three independent runs with different random seeds.

Table 7: **Comparison of STRIDE with vanilla RLVR and co-training baselines (with variance).** Standard deviations on Avg. columns are computed across three independent runs.

	Mathematical Reasoning						Out-of-Domain Reasoning				
Model	**MATH500**	**AIME2024**	**AIME2025**	**AMC2023**	**OlympiadBench**	**Avg.**	**BGQA**	**CRUXEval**	**StrategyQA**	**TableBench**	**Avg.**
Qwen2.5-7B-SFT	66.6	3.3	3.3	27.5	28.1	25.8 ± 2.1	46.6	44.3	85.9	34.4	52.8 ± 2.4
+ GRPO [2]	74.6	13.3	10.0	50.0	36.9	37.0 ± 1.8	55.3	48.8	88.1	38.2	57.6 ± 2.3
+ TANGO [18]	81.4	20.0	20.0	65.0	43.9	46.1 ± 2.0	60.5	51.4	90.0	42.3	61.1 ± 2.1
+ **STRIDE** (Ours)	**84.6**	**26.7**	**23.3**	**75.0**	**46.1**	**51.1** ± 2.2	**66.8**	**57.0**	**92.0**	**43.8**	**64.9** ± 2.3
Llama3.1-8B-SFT	57.2	6.6	3.3	42.5	28.0	27.5 ± 1.2	46.5	43.5	83.7	27.8	50.4 ± 1.4
+ GRPO [2]	66.8	6.6	6.6	47.5	34.2	32.3 ± 1.7	48.2	45.5	84.6	30.8	52.3 ± 2.1
+ TANGO [18]	69.2	6.6	6.6	50.0	35.0	33.5 ± 2.0	49.0	45.3	86.0	28.9	52.3 ± 1.8
+ **STRIDE** (Ours)	**70.4**	**13.3**	**10.0**	**50.3**	**36.0**	**36.0** ± 1.8	**50.2**	**46.0**	**88.2**	**32.3**	**54.2** ± 2.0

F Limitations

STRIDE introduces additional training complexity relative to vanilla RLVR, as it requires maintaining two separate models and executing an interleaved three-phase schedule; in our current implementation this amounts to approximately 40 hours on 8×H20 GPUs, compared to 32 hours for standard GRPO. The computational overhead could be reduced through selective redirection targeting only the most informative failure cases, parameter sharing between the generator and verifier, or parallelizing the three training phases. Additionally, while STRIDE is evaluated on mathematical and general reasoning benchmarks, its extension to open-ended tasks with non-verifiable outputs (e.g., creative writing or complex dialogue) would require rubric-based outcome signals such as LLM-as-a-judge, which introduces additional variance into the reward signal. Finally, the verifier's step-level localization, while improving over training, is not perfect; future work on dedicated process-level regularization could further strengthen its reliability.

G Case Studies

This section provides qualitative evidence demonstrating how STRIDE overcomes the inherent limitations of scalar rewards through high-bandwidth linguistic guidance.

G.1 Case Study I: Breaking Representational Collapse

As discussed in Section 3.2, a fundamental weakness of conventional PRMs is their unidimensional nature, which collapses semantically distinct errors into identical scalar values. Table 8 showcases two disparate error modes in an algebraic problem that are indistinguishable to a scalar verifier but are clearly resolved by STRIDE.

Table 8: Comparison of feedback bandwidth between Scalar Rewards and **STRIDE**.

Error Mode	**Scalar Reward**	**STRIDE Guidance (Informative Critique)**
Case A: Arithmetic Fallacy $2x + 5 = 13 \Rightarrow 2x = 18$	$r = 0.2$ (Low)	Incorrect additive inverse applied. The step incorrectly added 5 to the RHS instead of subtracting it.
Case B: Logical Leap $2x + 5 = 13 \Rightarrow x = 3$	$r = 0.2$ (Low)	Missing intermediate steps. While the result $x = 4$ is intended, the jump to $x = 3$ is both logically unsubstantiated and numerically wrong.

In both cases, a scalar reward model provides only a magnitude of failure (e.g., 0.2), leaving the generator to explore blindly. In contrast, **STRIDE** restores the semantic dimension, providing a clear gradient direction in natural language for the generator to redirect its trajectory.

G.2 Case Study II: Multi-Point Redirection Produces Diverse Strategies

A key property of Multi-Point Redirection is that redirecting from *different* prefix anchors elicits qualitatively different solution strategies, even for the same problem. We trace two concurrent redirection paths on the same failed trajectory to illustrate this diversity.

Initial Attempt (Failed)

Problem: Solve for x: $2^{2x} - 5(2^x) + 4 = 0$.

- `<step 1>` Let $u = 2^x$. Rewrite as $u^2 - 5u + 4 = 0$. *Verified: CORRECT*
- `<step 2>` Factor: $(u-4)(u+1) = 0$, so $u = 4$ or $u = -1$. *Verified: INCORRECT (FPF, $t^* = 2$)*

Verifier Guidance at $t^* = 2$:
"The factors of $+4$ summing to -5 must both be negative: $(-4) \times (-1) = 4$ and $(-4) + (-1) = -5$. Using $+1$ is a sign error; the correct factoring is $(u-4)(u-1) = 0$."

Path A: Rectification from t^*=2 (same strategy corrected)

- `<step 2'>` $(u-4)(u-1) = 0 \Rightarrow u = 4$ or $u = 1$.
- `<step 3>` $2^x = 4 \Rightarrow x = 2$; $2^x = 1 \Rightarrow x = 0$.

Final Answer: $\boxed{x \in \{0, 2\}}$ *Status: SUCCESS*
Strategy: substitution with corrected factoring.

Path B: Exploration from t=1 (structurally different strategy)

- `<step 1'>` Recognise $2^{2x} = (2^x)^2$. Factor the original expression directly as a quadratic in 2^x: $(2^x - 4)(2^x - 1) = 0$.
- `<step 2'>` $2^x = 4 \Rightarrow x = 2$; $2^x = 1 \Rightarrow x = 0$.

Final Answer: $\boxed{x \in \{0, 2\}}$ *Status: SUCCESS*
Strategy: direct factoring without substitution — a structurally different path.

Path A corrects the arithmetic error within the substitution framework; Path B bypasses substitution entirely by recognising the product structure of the original expression. Both paths reach the same correct answer via qualitatively different algebraic strategies, validating that multi-point redirection broadens the generator's solution repertoire rather than merely patching individual mistakes.

G.3 Case Study III: Latent Drift and Recovery from Pre-Failure Anchors

Errors in multi-step reasoning are often not purely local: a step that *appears* correct may reflect a suboptimal earlier choice whose consequences only manifest steps later, a phenomenon we term *latent drift*. Single-point redirection from the FPF corrects the visible symptom but leaves the underlying fragility in place. Multi-Point Redirection, by also redirecting from pre-FPF anchors, allows the generator to discover cleaner paths that avoid the root cause entirely.

Initial Attempt (Failed)

Problem: Solve for real x: $(x-1)^2 + (x+3)^2 = 20$.

- `<step 1>` Expand: $(x^2 - 2x + 1) + (x^2 + 6x + 9) = 20 \Rightarrow 2x^2 + 4x + 10 = 20$. *Verified: CORRECT*
- `<step 2>` Simplify: $2x^2 + 4x - 10 = 0 \Rightarrow x^2 + 2x - 5 = 0$. *Verified: CORRECT*
- `<step 3>` Quadratic formula: $x = \frac{-2 \pm \sqrt{4+20}}{2} = \frac{-2 \pm \sqrt{24}}{2}$. Simplify $\sqrt{24} = 2\sqrt{5}$, giving $x = -1 \pm \sqrt{5}$. *Verified: INCORRECT (FPF, $t^* = 3$)*

Verifier Guidance at $t^* = 3$:
"$\sqrt{24} = \sqrt{4 \cdot 6} = 2\sqrt{6}$, not $2\sqrt{5}$. The correct simplification gives $x = -1 \pm \sqrt{6}$."

Path A: Rectification from t^*=3 (FPF correction)

- `<step 3'>` $x = \frac{-2\pm\sqrt{24}}{2} = \frac{-2\pm2\sqrt{6}}{2} = -1 \pm \sqrt{6}$.

Final Answer: $\boxed{x = -1 \pm \sqrt{6}}$ *Status: SUCCESS*

Corrects the radical simplification error, but retains the algebraically heavy expansion from Step 1.

Path B: Exploration from t=1 (avoiding the root cause)

- `<step 1'>` Let $m = x + 1$. Then $(x - 1) = m - 2$ and $(x + 3) = m + 2$, so $(m - 2)^2 + (m + 2)^2 = 20$.
- `<step 2'>` Expand: $2m^2 + 8 = 20 \Rightarrow m^2 = 6 \Rightarrow m = \pm\sqrt{6}$.
- `<step 3'>` $x = m - 1 = -1 \pm \sqrt{6}$.

Final Answer: $\boxed{x = -1 \pm \sqrt{6}}$ *Status: SUCCESS*

Exploits the symmetry of the expression via the substitution $m = x + 1$, bypassing the discriminant computation entirely.

Path A repairs the arithmetic error at the detected failure step. However, the brute-force expansion introduced in Step 1 remains a latent source of fragility: similar problems would again require heavy discriminant arithmetic. Path B, redirected from $t = 1$, introduces a symmetry-aware substitution $m = x + 1$ that recognises the geometric structure of the sum of squares, rendering the subsequent arithmetic trivial and eliminating the error-prone radical simplification altogether. The root cause of the failure was not the wrong simplification at Step 3, but the choice of brute-force expansion at Step 1 that created a more error-prone algebraic path. This case demonstrates that multi-point redirection does not only fix local errors: by exploring from pre-FPF anchors, it uncovers more robust reasoning strategies that single-point redirection, restricted to the FPF alone, cannot reach.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the three main contributions: (1) formulating the information bottleneck problem of scalar process supervision, (2) proposing the STRIDE three-phase interleaved training framework with generative verifier co-training and multi-point redirection, and (3) demonstrating consistent empirical improvements across mathematical and out-of-domain reasoning benchmarks. All claims are substantiated by the experimental results in Sections 4 and 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: A dedicated Limitations section is included in Appendix F, discussing the increased training complexity and GPU-hour overhead relative to vanilla RLVR, the restriction to verifiable-output tasks, and the imperfection of the verifier's step-level localization.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper does not include any formal theoretical results, lemmas, or proofs. The information-bottleneck motivation in Section 3.2 is presented as intuitive analysis rather than a formal theorem.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: Appendix D (Implementation Details) provides complete information on model architectures, training data (Eurus-2-SFT-Data and Eurus-2-RL-Data), SFT and RL hyperparameters (learning rates, batch sizes, group sizes, KL coefficients), the interleaved training schedule ($f_G : f_V : f_R = 9 : 3 : 1$), hardware setup (8×H20 GPUs), and the verifier prompt template in Appendix C. All baseline configurations match publicly reported settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All training and evaluation datasets used in this work (Eurus-2-SFT-Data, Eurus-2-RL-Data, MATH-500, AMC 2023, AIME 2024/2025, OlympiadBench, BoardgameQA, CRUXEval, StrategyQA, TableBench) are publicly available and properly cited. The implementation code has been publicly released.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 (Experiments) describes model families, benchmarks, and baselines. Appendix D (Implementation Details) provides complete training configurations including optimizer, learning rates, batch sizes, group sizes, KL penalty, SFT steps, and hardware. Evaluation metrics (pass@1, CSR) are defined in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Standard deviations across three independent runs are reported in Appendix E: (1) the main comparison table (Appendix E.3, Avg. columns for all methods on both model series), (2) the Compute Fairness Analysis table (Appendix E.1), and (3) the Orthogonality with Inference-Time Scaling table (Appendix E.2). All values reflect the mean over runs with identical hardware and data conditions; standard deviations capture variance due to random seed differences.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Training compute is reported in Appendix D (8×H20 GPUs) and the Limitations section (approximately 40 hours for STRIDE vs. 32 hours for GRPO). The compute fairness analysis in Appendix E.1 additionally documents GPU hours across all compared methods.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research focuses on improving mathematical reasoning in language models using publicly available datasets. No human subjects are involved, no personally identifiable information is used, and the work does not involve dual-use risks or fairness concerns.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper presents foundational research on reinforcement learning for language model reasoning. The method improves mathematical problem-solving capabilities and does not introduce novel capabilities that enable direct harmful applications. There is no direct path from improved mathematical reasoning to misuse beyond what already exists in deployed language models.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release any new models, datasets, or tools that pose a high risk for misuse. The training and evaluation datasets used are publicly available academic benchmarks. No fine-tuned model weights are released with this submission.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets, base models, and codebases are properly cited with original references. The training framework veRL is used in accordance with its open-source license.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not introduce new datasets. The training code implementing the STRIDE framework has been publicly released; no new model weights are released at this stage.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects. All experiments are conducted on automated language model training and evaluation using publicly available datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects are involved in this research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: GPT-5 is used as an automatic judge in Section 5.2 (Fig. 3c/d) to evaluate step-level verifier quality, specifically measuring error localization accuracy and critique helpfulness over training. This constitutes a non-standard LLM component in the evaluation pipeline. GPT-5 was queried via the OpenAI API with a structured evaluation prompt; results are reported as averaged scores over 200 sampled trajectories per checkpoint. The LLMs used as the subject of study (serving as generator and verifier) are trained models, not methodological tools.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.