STRUCTURED REASONING FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long chains of thought interleave multiple reasoning abilities, yet common training objectives supervise complete trajectories or rely on final-answer correctness, making it difficult to identify and improve individual abilities. We propose **S**tru**C**tured **R**easoning (SCR), which organizes reasoning into explicit units with distinct functional roles. Our theoretical analysis characterizes the limitations of final-answer supervision and shows how structured trajectories support local evaluation and training. An analysis of four reasoning models further reveals that verification and revision are frequent but often ineffective. Motivated by this observation, we instantiate SCR as Generate-Verify-Revise and develop a training procedure that combines selective supervised fine-tuning with staged reinforcement learning for generation, verification, and revision. Across three backbone models, SCR consistently improves average reasoning accuracy over outcome-only GRPO, while producing shorter outputs. Code is available at https://anonymous.4open.science/r/SCR-8228.

1 INTRODUCTION

Large reasoning models (LRMs) achieve strong performance on complex tasks by generating long chains of thought (CoT) (OpenAI et al., 2024; Guo et al., 2025; Yang et al., 2025). These trajectories involve multiple atomic reasoning abilities, as shown in Figure 1. However, common training objectives provide supervision mainly at the trajectory level rather than the ability level. Supervised fine-tuning imitates complete reasoning traces without distinguishing the roles of different reasoning steps (Min et al., 2024; Chu et al., 2025), while reinforcement learning typically assigns rewards based on final-answer correctness (Guo et al., 2025; Kimi Team et al., 2025; Uesato et al., 2022). As a result, the same final outcome can arise from different successes and failures within the reasoning process. For example, a model may correctly detect an error but fail to revise it (Tyen et al., 2024). The incorrect final answer alone cannot determine whether verification or revision should be improved.

We propose **S**tru**C**tured **R**easoning (SCR), a general framework that organizes reasoning into explicit units corresponding to distinct reasoning abilities. Instead of treating a complete reasoning trajectory as a single training unit, SCR separates it into units with distinct functional roles. Each unit exposes the input and output of a particular ability, making its behavior directly observable when suitable checkers are available. This structure supports targeted supervision for different reasoning abilities rather than relying on a single signal over the entire trajectory. Our analysis in Section 2 shows that final-answer supervision does not necessarily reveal local reasoning quality and that the value of a local improvement depends on how it is used by subsequent reasoning.

The specific units in structured reasoning depend on which abilities are important to the target task. To instantiate SCR for mathematical reasoning, we analyze the reasoning trajectories of four representative LRMs in Section 3.1. We find that verification and revision occur frequently across models, while most revisions do not change answer correctness. Their high prevalence and limited effectiveness suggest that these abilities form an important bottleneck in long reasoning. Based on this observation, we instantiate SCR as Generate-Verify-Revise (GVR). The model first generates an initial solution, verifies its correctness, and revises it only when an error is identified. This structure exposes the outcome of each ability and makes their roles in the reasoning process explicit.

We use the GVR structure to determine which trajectory segments to imitate and which behaviors to optimize at each stage of reinforcement learning. During supervised fine-tuning, Dynamic

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

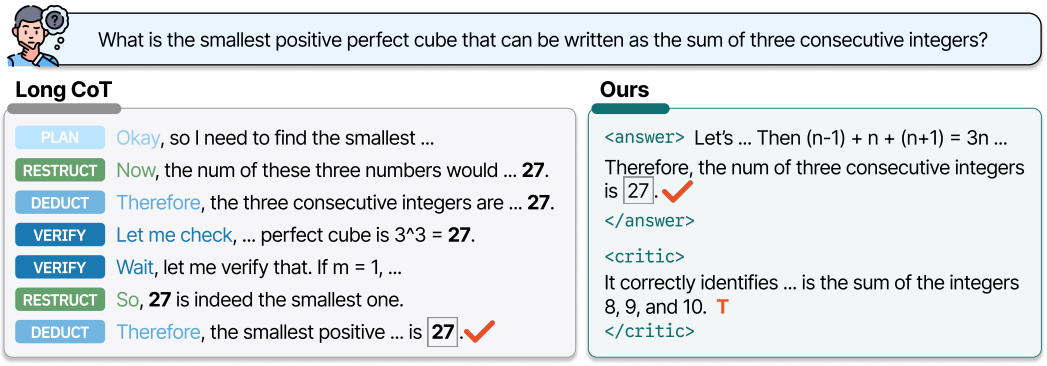

Figure 1: **Comparison of reasoning trajectories.** Long CoT entangles different atomic abilities within a single trajectory, while structured reasoning separates them into clearly bounded units.

Termination Supervision teaches the model whether to stop after verification or continue to revision. Selective Loss Masking keeps incorrect initial solutions as context for verification and revision while excluding them from imitation targets. We then apply GRPO (Shao et al., 2024) in two stages. Stage I optimizes initial generation and verification, while Stage II further optimizes revision with final-answer and revision-specific rewards. This design provides different training signals for different reasoning roles while optimizing a single autoregressive policy.

Across three backbone models, the GVR instantiation of SCR improves average reasoning accuracy over outcome-only GRPO and long-CoT SFT followed by GRPO. The gains over outcome-only optimization show the benefit of assigning supervision according to distinct reasoning roles rather than relying only on final-answer correctness. Despite introducing explicit verification and revision, SCR also produces shorter outputs than the long-CoT baseline on Qwen2.5-3B. This suggests that verification can regulate unnecessary continuation rather than simply increase inference cost.

Our contributions are as follows.

- We introduce SCR, a structured reasoning framework that decomposes reasoning trajectories into explicit and evaluable units. We analyze the limits of final-answer supervision and the conditions under which local improvements contribute to task accuracy.
- We instantiate SCR as Generate-Verify-Revise based on an empirical analysis of reasoning behaviors and develop an ability-aligned training procedure that combines selective imitation, termination supervision, and staged reinforcement learning.
- We validate the GVR instantiation across three backbone models and show improved average reasoning accuracy over outcome-only GRPO and long-CoT training baselines, together with shorter outputs on Qwen2.5-3B.

2 Theoretical Analysis

We study two questions about local reasoning abilities: whether their quality can be inferred from final-answer supervision, and when improving them leads to higher task accuracy. Proofs and additional analyses are provided in Appendix A.

Fix a task distribution and consider a reasoning trajectory $\boldsymbol{y} = (Y_1, \ldots, Y_K)$. At unit k, the policy generates Y_k from the history $H_k = (x, Y_1, \ldots, Y_{k-1})$. Let $R \in \{0, 1\}$ denote final-answer correctness and $A(\pi) = \mathbb{E}_\pi[R]$ the task accuracy. Given a fixed checker c_k, we define the correctness of unit k and its expected local quality as

$$Z_k = c_k(H_k, Y_k) \in \{0, 1\}, \qquad q_k(\pi) = \mathbb{E}_\pi[Z_k]. \tag{1}$$

Final outcomes need not determine local reasoning quality. Final-answer supervision observes whether the overall trajectory succeeds, but need not reveal which intermediate abilities succeed or fail. In particular, two policies may induce the same distribution over final outcomes while differing in the quality of a specific reasoning unit.

Theorem 1 (Non-identifiability from final outcomes)**.** *There exist a fixed task distribution and two policies that differ only in their segment-k conditional such that*

$$\mathrm{Law}_{\pi_1}(x, R) = \mathrm{Law}_{\pi_2}(x, R), \qquad q_k(\pi_1) \neq q_k(\pi_2). \tag{2}$$

This remains possible even when downstream success depends on the output of the unit.

Therefore, an objective determined only by this outcome distribution can assign the same value to policies with different local reasoning quality. Appendix A.1 gives a constructive proof and a finite-output characterization of identifiability.

Structured units make local supervision possible. Local quality becomes directly assessable when two conditions hold. R1 is *addressability*, which requires the unit (H_k, Y_k) to be explicitly identifiable in the trajectory. R2 is *decidability*, which requires a checker to determine Z_k from the unit and task-provided references. Together, these conditions expose a well-defined local training objective,

$$J_k(\pi) = \mathbb{E}_\pi[Z_k]. \tag{3}$$

The structure specifies which reasoning behavior is being evaluated, while the checker provides the corresponding supervision. Appendix A.2 gives the estimator, and Appendix A.4 analyzes the fixed-history gradient.

Local improvement does not automatically translate into task improvement. Even when local quality can be directly optimized, its effect on final accuracy depends on how useful that improvement is to subsequent reasoning. To isolate this effect, consider two policies with the same history distribution d_k, downstream policy, and output distributions within each correctness class. They differ only in

$$q_k(h; \pi) = \Pr_\pi(Z_k = 1 \mid H_k = h).$$

Let $r_z(h)$ denote the final success probability when the unit has correctness z at history h, averaged over the shared output distribution within that class.

Proposition 2 (Local quality and task accuracy)**.** *Under this controlled comparison,*

$$A(\pi') - A(\pi) = \mathbb{E}_{H_k \sim d_k}\left[\big(q_k(H_k; \pi') - q_k(H_k; \pi)\big)\big(r_1(H_k) - r_0(H_k)\big)\right]. \tag{4}$$

Higher average local quality alone does not imply higher task accuracy, even when $r_1(h) \geq r_0(h)$ for every history.

The proof and a counterexample are given in Appendix A.3. The value of a local improvement depends on where it occurs and how subsequent reasoning uses it. This motivates combining local supervision with optimization of the resulting task outcomes. Appendices A.5 and A.7 examine shared-parameter GRPO updates and single-round GVR accuracy.

3 Instantiating Structured Reasoning

We instantiate structured reasoning through Generation-Verification-Revision (GVR). We first motivate this design empirically, then present the trajectory representation and training procedure.

3.1 Empirical Motivation

We analyze the reasoning trajectories of four LRMs on MATH500 to examine the frequency and effectiveness of these atomic abilities. The ability taxonomy, experimental setup, and detailed annotation procedure are provided in Appendix D. The results are shown in Figure 2.

We observe that verification and revision are prevalent across all four LRMs, together accounting for over half of the annotated operations in each model. Yet most revision steps leave answer correctness unchanged, either preserving an already correct answer or failing to correct an incorrect one. Successful corrections account for only a small fraction of revision steps.

These observations motivate explicit supervision of verification and revision. We adopt GVR as a concrete instantiation of structured reasoning. Generation produces an initial solution, verification assesses whether its answer is correct, and revision attempts to correct detected errors. Explicit

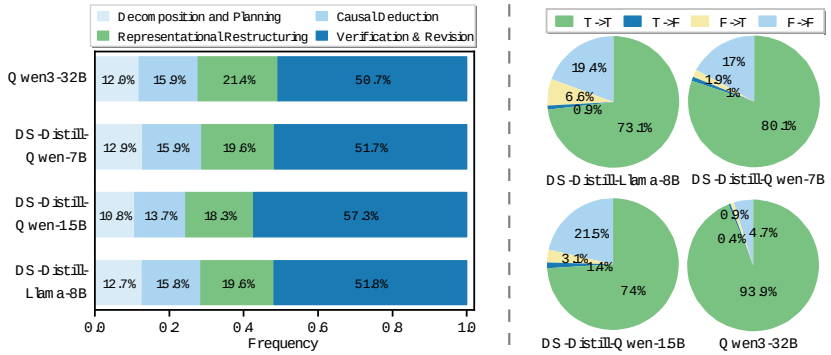

Figure 2: **Behavioral analysis of long CoT.** The left panel shows the frequency of different atomic abilities. The right panel shows the distribution of correctness changes induced by revision.

segment boundaries make each component addressable, while reference answers allow us to evaluate initial answer correctness, verification verdict accuracy, and revision outcomes separately. These evaluations provide targeted supervision for each component.

3.2 GVR Framework

Behavior cloning establishes the GVR trajectory structure, and staged reinforcement learning further optimizes its components. Figure 3 shows the overall training framework.

3.2.1 Structured Trajectory

A single policy π_θ produces the GVR segments as a single autoregressive sequence. For input x, let $y^{(0)}$ be the initial solution and $c^{(0)}$ its critique. The verdict $v^{(0)} = \mathrm{crit}(c^{(0)}) \in \{\mathrm{T}, \mathrm{F}\}$ is extracted from the critique and indicates whether the model judges the initial answer to be correct. A verdict of T accepts the solution and terminates the trajectory, while F triggers a revision $y^{(1)}$. A single round is

$$\boldsymbol{y} = \begin{cases} \left(y^{(0)}, c^{(0)}\right), & v^{(0)} = \mathrm{T}, \\ \left(y^{(0)}, c^{(0)}, y^{(1)}\right), & v^{(0)} = \mathrm{F}. \end{cases} \tag{5}$$

We write $y^{\mathrm{init}} = y^{(0)}$ and use y^{final} to denote the last solution produced. For a single round, $y^{\mathrm{final}} = y^{(1)}$ if revision occurs and $y^{\mathrm{final}} = y^{\mathrm{init}}$ otherwise. Additional rounds repeat verification and conditional revision on the latest solution. At inference time, the model's verdict determines whether to accept the current solution or attempt a revision. Reference answers provide training supervision rather than inference-time stopping signals. The trajectory probability factorization is given in Appendix C.3.

3.2.2 Behavior Cloning on Structured Trajectories

Behavior cloning uses two supervision mechanisms to govern when a trajectory terminates and which segments the model learns to imitate.

Data synthesis. For each problem, we sample candidate solutions from models of different scales and use the task checker to label their answers against the reference. This produces a mixture of correct and incorrect initial answers. A teacher model then produces a critique of each candidate, either confirming the answer or identifying an error and suggesting a revision. For an incorrect answer, the teacher first attempts a local revision based on the original reasoning. If the revised answer remains incorrect, the teacher generates a new solution. We retain only trajectories whose final answers pass the task checker and whose critique verdicts match the initial answers' correctness labels.

The resulting dataset contains two types of trajectories. Trajectories with correct initial answers terminate after a confirming critique, while correction trajectories continue with a corrected solution produced by revision or regeneration. We use opening and closing `<answer>`, `<critic>`, and `<revised>` tags to mark the initial solution, critique, and corrected solution. Sampling details and examples are provided in Appendix C.1.

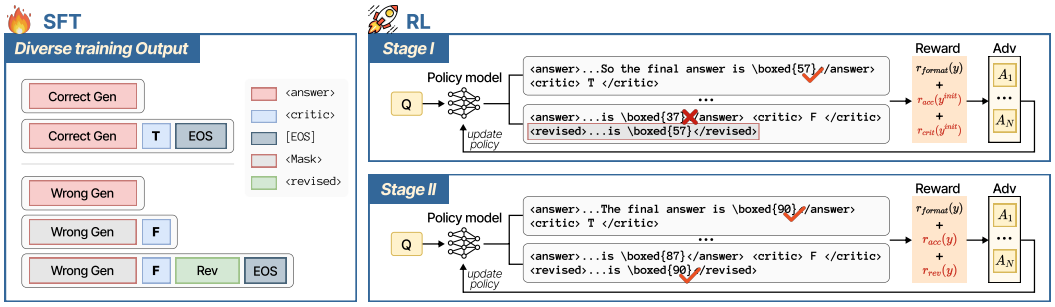

Figure 3: **Structured reasoning framework.** The left panel illustrates supervised fine-tuning with DTS and SLM. The right panel presents staged reinforcement learning, where Stage I jointly optimizes generation and verification, and Stage II focuses on revision.

Dynamic Termination Supervision (DTS). To teach the model when to stop reasoning, we place the `[EOS]` token according to the critique verdict. If the verdict is T, `[EOS]` immediately follows the critique's closing tag, indicating acceptance of the current solution. If the verdict is F, the trajectory continues to revision without an intervening `[EOS]`. In the single-round training trajectories, `[EOS]` follows the revised solution's closing tag. This supervision couples verification with the decision to stop or revise, discouraging unnecessary continuation after acceptance and premature termination after rejection.

Selective Loss Masking (SLM). Supervising an entire correction trajectory would also train the model to imitate its incorrect initial solution. This may encourage avoidable errors and subsequent revisions. We therefore mask the incorrect initial solution from the loss while retaining it as context for verification and revision. Let y_i denote the i-th token of trajectory $\boldsymbol{y}$, $|\boldsymbol{y}|$ its length, and $m_i \in \{0, 1\}$ the loss mask. We minimize

$$\mathcal{L} = -\sum_{i=1}^{|\boldsymbol{y}|} m_i \log \pi_\theta(y_i \mid x, y_{<i}). \tag{6}$$

For correction trajectories, we set $m_i = 0$ for tokens in the incorrect initial solution and $m_i = 1$ for tokens in the critique and corrected solution. For trajectories with correct initial answers, all tokens remain supervised. Termination tokens remain supervised in both cases. Appendix C.2 provides the serialization and masking details. Proposition 9 analyzes the resulting imitation objective under the assumptions in Appendix A.6.

3.2.3 Staged Reinforcement Learning

We further optimize the policy with Group Relative Policy Optimization (GRPO). Since effective revision relies on both a meaningful initial solution and a reliable verification signal, we adopt a two-stage training strategy. Stage I optimizes generation and verification, while Stage II focuses on revision starting from the Stage I policy. For each problem, the rollout policy π_{old} samples G trajectories, and we compute advantages using group-normalized rewards.

Stage I. Stage I uses three rule-based rewards to jointly optimize generation and verification while preserving the GVR structure. The format reward checks five structural requirements, including segment boundaries, verdict-dependent branching, and answer formatting. The complete list appears in Appendix C.3. Let $C_i(\boldsymbol{y}) \in \{0, 1\}$ indicate whether the i-th requirement is satisfied. We define:

$$r_{\text{format}}(\boldsymbol{y}) = \prod_{i=1}^{5} C_i(\boldsymbol{y}). \tag{7}$$

The initial-accuracy reward encourages the model to solve the problem correctly before revision. Using the task-specific checker, we define $r_{\text{acc}}(y^{\text{init}}) = \mathbb{I}\big[y^{\text{init}} \text{ is correct}\big]$. The self-verification reward measures whether the critique verdict matches the initial answer's correctness. It is given by

$$r_{\text{crit}}(\boldsymbol{y}) = \mathbb{I}\Big[\Big(r_{\text{acc}} = 1 \wedge v^{(0)} = \text{T}\Big) \vee \Big(r_{\text{acc}} = 0 \wedge v^{(0)} = \text{F}\Big)\Big]. \tag{8}$$

Both correct acceptance and correct rejection receive a reward of one. The Stage I reward is

$$r^{\text{I}}_{\text{total}}(\boldsymbol{y}) = \alpha r_{\text{format}}(\boldsymbol{y}) + \beta r_{\text{acc}}(y^{\text{init}}) + \gamma r_{\text{crit}}(\boldsymbol{y}). \tag{9}$$

To focus optimization on generation and verification, we mask the <revised> segment from the policy loss and optimize only tokens in the <answer> and <critic> blocks. The revision is still generated and constrained by the format reward, but its tokens do not contribute directly to the policy loss. Let $\mathcal{I}_i$ denote the unmasked token positions in trajectory $\boldsymbol{y}_i$. We maximize

$$J_{\text{Stage-I}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{I}_i|}\sum_{t\in\mathcal{I}_i}\min\left(r_{i,t}A_i, \text{clip}(r_{i,t}, 1-\epsilon, 1+\epsilon)A_i\right)\right], \tag{10}$$

where $r_{i,t}$ is the probability ratio between π_θ and π_{old} for token t in trajectory i, conditioned on the problem and all preceding tokens. The advantage A_i is derived from $r^{\text{I}}_{\text{total}}$ and shared across the unmasked tokens of the trajectory. Clipping is applied separately to each token ratio, with threshold ϵ.

Stage II. Stage II starts from the Stage I policy and focuses on effective revision. We retain the format reward and replace the Stage I accuracy and verification rewards with final-answer accuracy and a revision reward. The final-answer accuracy reward evaluates the correctness of the final solution and is defined as $r^{\text{f}}_{\text{acc}}(y^{\text{final}}) = \mathbb{I}[y^{\text{final}} \text{ is correct}]$.

Final-answer accuracy alone does not indicate whether a revision is beneficial. We therefore evaluate revision quality by comparing the correctness of the initial and final solutions. For trajectories that contain a revision, we define:

$$r_{\text{rev}}(\boldsymbol{y}) = \begin{cases} \mu_1, & r_{\text{acc}}(y^{\text{init}}) = 0,\ r^{\text{f}}_{\text{acc}}(y^{\text{final}}) = 1, \\ \mu_2, & r_{\text{acc}}(y^{\text{init}}) = 0,\ r^{\text{f}}_{\text{acc}}(y^{\text{final}}) = 0, \\ \mu_3, & r_{\text{acc}}(y^{\text{init}}) = 1,\ r^{\text{f}}_{\text{acc}}(y^{\text{final}}) = 1, \\ \mu_4, & r_{\text{acc}}(y^{\text{init}}) = 1,\ r^{\text{f}}_{\text{acc}}(y^{\text{final}}) = 0. \end{cases} \tag{11}$$

We set $r_{\text{rev}} = 0$ when no revision occurs and use negative coefficients satisfying $\mu_1 > \mu_2 > \mu_3 \geq \mu_4$. All revisions incur a penalty, with the smallest assigned to successful correction and larger penalties to ineffective, unnecessary, and harmful revisions. Successful corrections also earn the final-answer reward. With the chosen coefficients, correcting an incorrect answer yields a higher total reward than leaving it unchanged, while a correct initial answer receives the highest reward when accepted without revision. The total Stage II reward is

$$r^{\text{II}}_{\text{total}}(\boldsymbol{y}) = \alpha' r_{\text{format}}(\boldsymbol{y}) + \beta' r^{\text{f}}_{\text{acc}}(y^{\text{final}}) + \gamma' r_{\text{rev}}(\boldsymbol{y}). \tag{12}$$

Stage II removes the revision mask and applies the GRPO objective in Eq. (10) to the full trajectory. The active set $\mathcal{I}_i$ now includes all token positions, and advantages are derived from $r^{\text{II}}_{\text{total}}$. This jointly adapts generation, verification, and revision to the final correction outcome. Reward coefficients and optimization details are provided in Appendix C.3.

4 Experiments

4.1 Experimental Setup

Models and datasets. We conduct experiments on Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct (Qwen Team, 2024), and Llama3.1-8B-Instruct (Grattafiori et al., 2024). For SFT, we use MATH (Hendrycks et al., 2021) problems at difficulty levels 3 to 5 and remove overlap with the test set. For RL, we use DAPO-MATH (Yu et al., 2025) after filtering out Chinese problems.

Evaluation. We evaluate in-domain mathematical reasoning on AIME24, AIME25, MATH500, AMC (Li et al., 2024), and Olympiad (He et al., 2024). Out-of-domain evaluation uses GPQA-Diamond(GPQA) (Rein et al., 2024), ARC (Clark et al., 2018), and a subset of MMLU-Pro (MMLU) (Wang et al., 2024b). For AIME24, AIME25, and AMC, we report *avg@10* at temperature 0.6. For the other benchmarks, we report *pass@1* at temperature 0.

Baselines. We compare SCR with four baselines. **Base** is the original model without fine-tuning. **On-policy RL** applies GRPO directly to Base. **Self-Refine** (Madaan et al., 2023) iteratively critiques and revises outputs through prompting. **SFT + GRPO** first trains on long-CoT demonstrations and then applies GRPO. Appendix B provides experimental settings and hyperparameters.

4.2 Main Results

Table 1 compares SCR with prompting-based and training-based baselines. **Training atomic reasoning abilities can improve final performance.** SCR outperforms both on-policy GRPO and SFT +

Method	In-Domain					Out-of-Domain			AVG
	MATH	AIME24	AIME25	Olympiad	AMC	ARC	MMLU	GPQA	
Qwen2.5-3B-Instruct									
Base	66.60	7.00	2.33	29.93	27.95	69.20	39.60	29.80	34.05
Self-Refine	65.60	6.67	3.00	29.78	29.52	63.57	43.50	32.32	34.25
GRPO	70.20	6.00	5.00	33.04	35.90	81.14	44.60	36.36	39.03
SFT + GRPO	70.00	7.33	3.67	35.11	35.06	79.01	45.30	36.36	38.98
SCR-SFT	63.80	2.33	2.33	24.59	23.86	29.69	38.40	27.27	26.53
SCR-Stage I	65.20	4.33	2.00	29.04	29.16	67.32	41.30	39.39	34.72
SCR (Ours)	70.40 $_{+3.80}$	7.33 $_{+0.33}$	4.00 $_{+1.67}$	37.03 $_{+7.10}$	36.02 $_{+8.07}$	79.35 $_{+10.15}$	44.50 $_{+4.90}$	39.39 $_{+9.59}$	**39.75** $_{+5.70}$
Qwen2.5-7B-Instruct									
Base	75.60	11.00	6.00	39.70	42.05	84.39	55.80	33.84	43.55
Self-Refine	68.40	9.33	5.67	36.44	38.67	57.51	55.90	34.85	38.35
GRPO	77.40	11.33	10.33	42.81	50.60	85.49	57.00	37.88	46.60
SFT + GRPO	78.80	15.33	11.67	42.81	50.84	80.63	57.80	35.86	46.72
SCR-SFT	74.20	9.33	4.00	34.67	35.18	72.95	49.10	33.84	39.16
SCR-Stage I	77.00	11.33	9.00	42.07	44.46	80.63	51.50	35.35	43.92
SCR (Ours)	80.00 $_{+4.40}$	13.67 $_{+2.67}$	13.67 $_{+7.67}$	45.63 $_{+5.93}$	50.12 $_{+8.07}$	86.43 $_{+2.04}$	56.30 $_{+0.50}$	38.38 $_{+4.54}$	**48.03** $_{+4.48}$
Llama3.1-8B-Instruct									
Base	42.80	2.00	0.00	15.85	12.77	57.08	42.40	24.75	24.71
Self-Refine	13.40	2.00	0.00	5.04	7.23	8.79	9.90	22.73	8.64
GRPO	52.20	8.33	1.00	20.15	24.22	70.31	45.50	31.31	31.63
SFT + GRPO	57.60	6.00	0.67	22.52	28.67	84.73	43.00	24.24	33.43
SCR-SFT	53.40	3.00	1.33	19.11	19.04	74.06	39.10	24.75	29.22
SCR-Stage I	55.80	4.00	1.67	23.41	26.51	82.17	42.20	31.82	33.45
SCR (Ours)	60.80 $_{+18.00}$	5.33 $_{+3.33}$	4.67 $_{+4.67}$	23.70 $_{+7.85}$	24.46 $_{+11.69}$	84.56 $_{+27.48}$	43.20 $_{+0.80}$	27.27 $_{+2.52}$	**34.25** $_{+9.54}$

Table 1: **SCR vs. other baselines.** SCR consistently outperforms the Base across all three models. The best results are highlighted in bold, and subscripts denote the absolute change of SCR relative to the Base (green for improvement, red for degradation).

GRPO in average performance across backbones. By explicitly supervising generation, verification, and revision, SCR improves final performance beyond objectives that rely primarily on final-answer rewards. These results provide empirical support for the ability-specific training motivated by our theoretical analysis in Section 2.

In contrast, prompting and imitation alone do not reliably improve reasoning. Self-Refine and SCR-SFT yield inconsistent gains, suggesting that eliciting self-correction or imitating correction trajectories is insufficient for consistent improvement. Staged RL further improves average performance over SCR-SFT across backbones by optimizing the model on its own answers and correction outcomes. This indicates that demonstrations provide a useful initialization, while RL is needed to adapt the correction procedure to the model's own reasoning behavior.

Training atomic reasoning abilities yields benefits beyond mathematics. Although trained only on mathematical data, SCR consistently outperforms other dataset across all evaluated backbones. This suggests that the learned improvements are not limited to indomain-task, but transfer to broader reasoning tasks. Besides, verification and revision are general reasoning abilities that are useful whenever a model must assess intermediate conclusions and correct flawed reasoning.

4.3 Analysis of Verification Capability

Table 2 compares the self-verification performance of Base, SCR-SFT, and SCR-Stage I. **Stage I improves verification reliability beyond SFT.** It achieves the highest average accuracy, precision, and F1 across all backbones, with consistently higher precision across benchmarks. This is particularly important for GVR because verification acts as the decision point between termination and revision. A false acceptance prematurely terminates reasoning and eliminates the opportunity to correct an incorrect solution. The higher precision of Stage I therefore indicates more reliable control over when revision should be triggered. Its consistent gains over SFT also reveal a distinction between learning the structure of verification and optimizing its decisions. SFT provides the verification behavior through imitation, while Stage I further improves how this behavior is applied to the model's own sampled solutions. These results further show the value of explicitly separating reasoning abilities and directly optimizing each ability for its role in the reasoning process.

Method	MATH500				Olympiad				AMC				AVG			
	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1
Qwen2.5-3B-Instruct																
Base	71.80	71.92	94.59	81.71	50.37	36.51	89.11	51.80	53.37	35.62	82.76	49.81	58.51	48.02	88.82	61.11
SCR-SFT	78.57	78.16	100.00	87.74	58.70	55.12	100.00	71.07	57.08	37.60	71.21	49.21	64.78	56.96	90.40	69.34
SCR-Stage I	80.63	82.55	90.85	86.50	82.76	73.63	75.51	74.56	83.15	72.37	68.75	70.51	82.18	76.18	78.37	77.19
Qwen2.5-7B-Instruct																
Base	76.95	79.91	92.84	85.89	56.15	47.41	95.52	63.37	59.64	51.10	93.12	65.99	64.25	59.47	93.83	71.75
SCR-SFT	87.35	87.28	99.43	92.96	62.29	57.72	99.07	72.95	69.51	61.35	78.28	68.79	73.05	68.78	92.26	78.23
SCR-Stage I	87.02	90.82	93.13	91.96	84.14	81.31	83.04	82.17	82.40	80.81	80.16	80.48	84.52	84.31	85.44	84.87
Llama3.1-8B-Instruct																
Base	68.75	59.86	82.63	69.43	71.13	32.95	80.19	46.70	57.19	20.00	79.05	31.92	65.69	37.60	80.62	49.35
SCR-SFT	69.83	68.32	98.86	80.81	48.03	38.41	96.80	55.00	59.66	28.71	76.92	41.81	59.17	45.15	90.86	59.21
SCR-Stage I	78.36	79.93	85.66	82.70	81.87	69.01	62.42	65.55	81.39	85.43	93.75	89.40	80.54	78.12	80.61	79.22

Table 2: **Self-verification across training stages.** Correct answers are the positive class for precision, recall, and F1. AVG reports the unweighted mean of each metric across MATH500, Olympiad, and AMC. Blue cells indicate the best result within each backbone.

4.4 Analysis of Output Token Length

To examine whether verification and multi-round correction increase inference cost, we compare SCR with the long-CoT baseline SFT + GRPO on Qwen2.5-3B-Instruct. **SCR improves performance with shorter outputs.** As shown in Figure 4 and Table 1, SCR produces shorter outputs while achieving comparable or higher accuracy across the evaluated math benchmarks. This behavior aligns with its training design, which encourages concise reasoning within each round and uses verification to determine whether further revision is necessary. The Stage II reward reinforces this design by penalizing unnecessary and ineffective revisions.

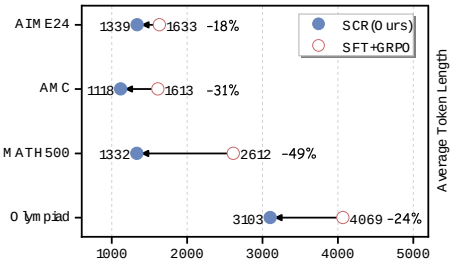

Figure 4: **Output token length comparison.** On Qwen2.5-3B, SCR consistently uses fewer tokens than SFT + GRPO.

As a result, SCR avoids unnecessarily extending the reasoning trajectory when the current solution is already sufficient. This suggests that explicit verification does not necessarily introduce additional inference overhead. Instead, it can help allocate computation only when further reasoning is useful. Verification therefore serves both to assess the current solution and to regulate subsequent reasoning.

4.5 Ablation Analysis

We examine how SFT supervision and staged RL contribute to self-correction. **SFT benefits from selective imitation and stopping supervision.** Figure 5 ablates DTS and SLM on Qwen2.5-3B-Instruct. Removing DTS degrades performance on all evaluated tasks, while removing SLM hurts most tasks. DTS connects verification with the decision to stop or revise. SLM preserves incorrect solutions as context for subsequent verification and revision while excluding them from the imitation loss. These mechanisms exploit information that final correctness alone cannot provide. Even when a trajectory ends successfully, its earlier content and continuation decisions need not all be desirable training targets.

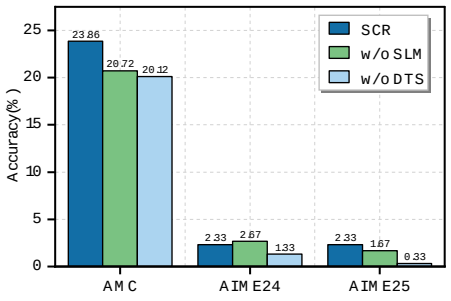

Figure 5: **SFT ablations.** Ablating DTS or SLM generally degrades performance.

Stage II converts verification ability into effective correction. Figure 6 compares staged RL with outcome-only RL based solely on final-answer correctness. Stage I does not consistently outperform this baseline, while the full two-stage procedure does across all evaluated tasks. Stage I optimizes initial generation and verification, whereas Stage II further rewards whether revision improves the final solution. The results indicate that recognizing an error alone is insufficient. The model must also learn to correct it effectively. This pattern is consistent with Proposition 2, which links the value

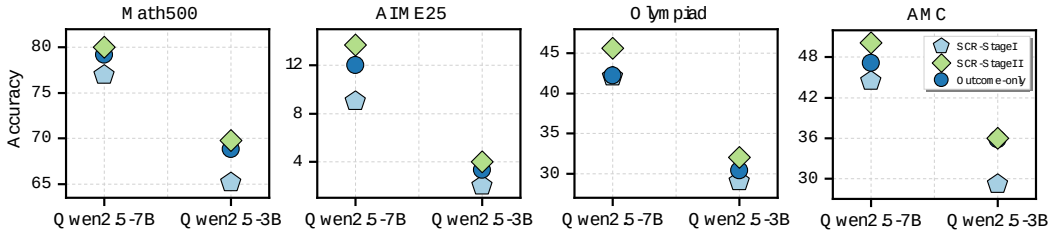

Figure 6: **Staged RL versus outcome-only RL on two Qwen backbones.** The full two-stage procedure outperforms both Stage I and outcome-only RL on all four tasks. Stage I alone does not consistently outperform outcome-only RL.

of an atomic improvement to downstream behavior. It highlights the need to train the interaction between verification and revision rather than treating them as independently sufficient abilities.

5 Related Work

Enhancing Deep Thinking Capability in LLMs. Test-time scaling improves LLM reasoning by allocating more computation to longer reasoning trajectories or multiple candidate solutions (Snell et al., 2025; Muennighoff et al., 2025). Reinforcement learning with verifiable rewards further strengthens reasoning through automatically checkable task feedback (Shao et al., 2024; Guo et al., 2025; Kimi Team et al., 2025). Most RLVR methods assign rewards based on final-answer correctness, typically verified by answer matching or code execution, without directly evaluating intermediate reasoning (Guo et al., 2025; Kimi Team et al., 2025). Process-supervised methods provide more localized signals. Math-Shepherd introduces step-level process rewards (Wang et al., 2024a), while ATTNPO derives step-level supervision from model attention to reduce redundant reasoning (Nie et al., 2026). Our work takes a complementary view. Instead of supervising individual derivation steps, we organize reasoning into explicit units with distinct functional roles. When these units can be locally evaluated, this structure supports ability-specific supervision.

Self-Correction in LLMs. Self-correction methods improve an initial response through self-generated or external feedback (Madaan et al., 2023; Shinn et al., 2023). Prompting-based approaches can support iterative refinement, but intrinsic correction remains unreliable when models fail to identify their own errors (Huang et al., 2024; Tyen et al., 2024; Kamoi et al., 2024). Recent post-training methods therefore learn correction behavior more directly. RISE trains models to improve over multiple turns (Qu et al., 2024), while SCoRe uses two-stage on-policy RL with reward shaping for improvements in answer correctness (Kumar et al., 2025). S^2R combines masked supervised initialization and verification-guided stopping with outcome-level or action-level RL (Ma et al., 2025), and SFS-DPO optimizes step-level correction preferences (Anh et al., 2026). CURE jointly trains solving, critiquing, and hint-guided re-exploration within a single policy (Chen et al., 2026). SCR differs from these approaches by organizing reasoning into explicit functional units and optimizing them with targeted supervision. Using GVR as a concrete instantiation, we separately train generation, verification, and revision rather than treating self-correction as a single behavior.

6 Conclusion

We present SCR, which structures reasoning into explicit and evaluable units to provide supervision for individual reasoning abilities. Our analysis explains the limitations of final-answer supervision and motivates local evaluation and training. We instantiate SCR as Generate-Verify-Revise and develop a training procedure that combines structured supervision with staged reinforcement learning. Experiments across three backbones show consistent improvements in average reasoning accuracy over outcome-based training baselines. Our results show that improving individual reasoning abilities requires not only identifying local errors but also optimizing how these abilities contribute to subsequent reasoning. Future work can extend SCR to other reasoning abilities with checkable intermediate outputs. More broadly, we hope this work motivates further study of structured reasoning as a general framework for building reasoning models whose internal problem-solving processes are more explicit, measurable, and trainable.

7 Ethics Statement

This work adheres to the ICLR Code of Ethics. In this study, no human subjects or animal experimentation was involved. All datasets used were sourced in compliance with relevant usage guidelines, ensuring no violation of privacy. We have taken care to avoid any bias or discriminatory outcomes in our research process. No personally identifiable information was used, and no experiments were conducted that could raise privacy or security concerns. We are committed to maintaining transparency and integrity throughout the research process.

8 Reproducibility Statement

We have made every effort to ensure that the results presented in this paper are reproducible. All code and datasets have been made publicly available in a repository to facilitate replication and verification. The experimental setup, including training steps, model configurations, and hardware details, is described in detail in the paper. Additionally, all datasets are publicly available, ensuring consistent and reproducible evaluation results. We believe these measures will enable other researchers to reproduce our work and further advance the field.

9 AI Usage Statement

During manuscript preparation, we used large language models only to polish wording and check English grammar, spelling, and word choice. The research ideas, core methodology, theoretical analysis, and experimental design were developed by the authors. All suggested edits were reviewed by the authors, who take full responsibility for the paper's content. This statement concerns writing assistance only. The use of language models for data construction and experimental analysis is described separately in the paper.

References

Vu Duc Anh, Nhat M. Hoang, Do Xuan Long, Cong-Duy Nguyen, Ponhvoan Srey, and Luu Anh Tuan. Reinforcing step-level reasoning for effective self-correction in LLMs. arXiv preprint arXiv:2608.11573, 2026. URL https://arxiv.org/abs/2608.11573.

Guirong Chen, Shuqi Ye, Wenkai Yang, Shiqi Shen, Guangyao Shen, and Yankai Lin. CURE: Critique-driven unified reinforcement learning for test-time self-improvement. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28632–28653, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1321. URL https://aclanthology.org/2026.acl-long.1321/.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10818–10838. PMLR, 2025. URL https://proceedings.mlr.press/v267/chu25c.html.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. arXiv preprint arXiv:1803.05457, 2018. URL https://arxiv.org/abs/1803.05457.

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. URL https://arxiv.org/abs/2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, pp. 32808–32824, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8b4add8b0aa8749d80a34ca5d941c355-Abstract-Conference.html.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. doi: 10.1162/tacl_a_00713. URL https://aclanthology.org/2024.tacl-1.78/.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, et al. Kimi k1.5: Scaling reinforcement learning with LLMs, 2025.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, JD Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, pp. 54523–54549, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/871ac99fdc5282d0301934d23945ebaa-Abstract-Conference.html.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. S^2R: Teaching LLMs to self-verify and self-correct via reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22632–22654, 2025. doi: 10.18653/v1/2025.acl-long.1104. URL https://aclanthology.org/2025.acl-long.1104/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, et al. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. arXiv preprint arXiv:2412.09413, 2024. URL https://arxiv.org/abs/2412.09413.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Shuaiyi Nie, Siyu Ding, Wenyuan Zhang, Linhao Yu, Tianmeng Yang, Yao Chen, Weichong Yin, Yu Sun, Hua Wu, and Tingwen Liu. ATTNPO: Attention-guided process supervision for efficient reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 39728–39748, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1845. URL https://aclanthology.org/2026.acl-long.1845/.

OpenAI. Introducing GPT-4.1 in the API. OpenAI, April 2025. URL https://openai.com/index/gpt-4-1/.

OpenAI, 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:2412.16720v1, 2024. URL https://arxiv.org/abs/2412.16720v1.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *Advances in Neural Information Processing Systems*, volume 37, pp. 55249–55285. Curran Associates, Inc., 2024. doi: 10.52202/079017-1754. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/639d992f819c2b40387d4d5170b8ffd7-Abstract-Conference.html.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297. Association for Computing Machinery, 2025. doi: 10.1145/3689031.3696075. URL https://dl.acm.org/doi/10.1145/3689031.3696075.

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*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Herbert A. Simon. Information-processing theory of human problem solving. In W. K. Estes (ed.), *Handbook of Learning and Cognitive Processes: Human Information Processing*, volume 5, chapter 7, pp. 271–295. Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, pp. 10131–10165, 2025.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13894–13908, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.826. URL https://aclanthology.org/2024.findings-acl.826/.

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. URL https://arxiv.org/abs/2211.14275.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024b. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244. Curran Associates, Inc., 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 400–410, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-demos.38. URL https://aclanthology.org/2024.acl-demos.38/.

APPENDIX

A ADDITIONAL ANALYSIS AND PROOFS

We give the proofs for Section 2, characterize local identifiability, and examine the supervision used in GVR. The task distribution and checkers are fixed. All policy kernels and checkers are measurable. A local policy comparison varies the segment-k conditional while fixing the history law and downstream policy. Additional assumptions are stated where needed.

A.1 NON-IDENTIFIABILITY FROM FINAL OUTCOMES

Proof of Theorem 1. Consider a single input x_0 and a fixed history before segment k. The segment has three possible outputs y_1, y_2, y_3. Their checker labels and success probabilities under a fixed downstream policy are

$$\big(c_k(h,y_1), c_k(h,y_2), c_k(h,y_3)\big) = (1,0,0), \qquad \big(v_k(h,y_1), v_k(h,y_2), v_k(h,y_3)\big) = \left(1, \frac{1}{2}, 0\right).$$

Here $v_k(h,y) = \Pr(R = 1 \mid H_k = h, Y_k = y)$ is determined by the shared continuation. Let the two segment policies assign probabilities

$$\mu_1 = \left(\frac{3}{8}, \frac{1}{4}, \frac{3}{8}\right), \qquad \mu_2 = \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right).$$

Both policies have final success probability $1/2$, but their local qualities are $3/8$ and $1/4$. Since the input is fixed and R is binary, the policies induce the same law of (x, R). They differ only in the segment conditional, and the continuation success probability is not constant across segment outputs. This proves the claim. □

A local characterization. The construction is an instance of a more general finite-output comparison. Fix a history h, a downstream policy, and candidate outputs $y_1, \dots, y_m$. Write $z_i = c_k(h, y_i)$ and $v_i = v_k(h, y_i)$. For a probability vector μ over these outputs, define

$$a(\mu) = \sum_{i=1}^{m} \mu_i v_i, \qquad q(\mu) = \sum_{i=1}^{m} \mu_i z_i.$$

The policy class in the following result contains every probability vector over the candidate outputs. Local quality is identifiable from the final success probability when $a(\mu) = a(\nu)$ implies $q(\mu) = q(\nu)$ for every pair μ, ν.

Proposition 3 (Identifiability in a finite local comparison)**.** *Local quality is identifiable from the final success probability if and only if there exist constants $\alpha, \beta \in \mathbb{R}$ such that*

$$z_i = \alpha + \beta v_i \qquad \textit{for every } i.$$

Proof. If $z_i = \alpha + \beta v_i$, then $q(\mu) = \alpha + \beta a(\mu)$, so equal final success probabilities imply equal local qualities.

For the converse, let $\boldsymbol{v} = (v_1, \dots, v_m)^\top$ and $\boldsymbol{z} = (z_1, \dots, z_m)^\top$. If $\boldsymbol{z}$ is not in the span of $\mathbf{1}$ and $\boldsymbol{v}$, there is a vector u satisfying

$$\mathbf{1}^\top u = 0, \qquad \boldsymbol{v}^\top u = 0, \qquad \boldsymbol{z}^\top u \neq 0.$$

Take the uniform probability vector μ_0. For sufficiently small $t > 0$, both $\mu_+ = \mu_0 + tu$ and $\mu_- = \mu_0 - tu$ have positive entries and sum to one. They have the same final success probability, while

$$q(\mu_+) - q(\mu_-) = 2t\boldsymbol{z}^\top u \neq 0.$$

Thus local quality is not identifiable. □

This characterization concerns mixtures over a fixed set of outputs at one history. It does not assert that an arbitrary pair of such mixtures is realizable by changing the parameters of a given neural model. Its role is to show that a dependence of final success on segment output is not sufficient for identifying local quality.

Outcome-insensitive continuations. A simpler special case arises when $v_k(h, y) = m(h)$ for every supported output at almost every history. Let d_k be the shared history law and let π_μ use segment conditional μ. Then

$$\Pr_{\pi_\mu}(R = 1 \mid x) = \int d_k(dh \mid x) \int m(h)\, \mu(dy \mid h) = \int m(h)\, d_k(dh \mid x).$$

The outcome law is independent of μ. Any two admissible segment policies with different local qualities therefore give another non-identifiable pair. Unlike this special case, the construction in Theorem 1 allows segment outputs to have different effects on downstream success.

A.2 Local evaluation and training targets

R1 exposes each unit (H_k, Y_k), and R2 supplies the checker label Z_k. Suppose evaluation uses n independent trajectories from a fixed policy π, each containing segment k.

Proposition 4 (Estimation of local quality)**.** *The observable labels give an unbiased estimator and define a local training objective as*

$$\widehat{q}_k = \frac{1}{n} \sum_{i=1}^{n} Z_k^{(i)}, \qquad J_k(\pi) = \mathbb{E}_\pi[Z_k] = q_k(\pi), \qquad \mathbb{E}_\pi[\widehat{q}_k] = q_k(\pi).$$

Proof. Each label has expectation $q_k(\pi)$, so linearity of expectation gives the estimator identity. The objective identity follows from the definition of local quality. □

The checker determines what this objective measures. In GVR, the verification label evaluates agreement between the verdict and initial-answer correctness. It does not certify the validity of every statement in the critique. Making a unit explicit therefore supports evaluation only to the extent allowed by its checker.

For an optional segment, let $B_k = 1$ indicate that it is reached and assume $\Pr_\pi(B_k = 1) > 0$. The mean over n independent trajectories drawn conditional on $B_k = 1$ is unbiased for

$$q_k^{\mathrm{act}}(\pi) = \mathbb{E}_\pi[Z_k \mid B_k = 1].$$

Assigning zero reward to unreached segments instead gives the objective $\Pr_\pi(B_k = 1) q_k^{\mathrm{act}}(\pi)$, which also depends on the probability of reaching the segment.

A.3 Local quality and task accuracy

Assumption 5 (Controlled local comparison)**.** The policies π and π' share the history law d_k and downstream kernels. For each $z \in \{0, 1\}$, they also share a class-conditional output kernel $\nu_z(dy \mid h)$ supported on $\{y : c_k(h, y) = z\}$. These kernels are specified even when a class has zero probability under one of the policies.

Write $q_k(h; \pi) = \Pr_\pi(Z_k = 1 \mid H_k = h)$ and

$$r_z(h) = \int v_k(h, y)\, \nu_z(dy \mid h).$$

The comparison changes only the mixture weight of the two correctness classes at each history. It holds fixed both the continuation and the distributions of outputs within each class.

Proof of Proposition 2. For either policy $\widetilde{\pi} \in \{\pi, \pi'\}$, conditioning on the correctness class gives

$$\mathbb{E}_{\widetilde{\pi}}[R \mid H_k = h] = q_k(h; \widetilde{\pi}) r_1(h) + \big(1 - q_k(h; \widetilde{\pi})\big) r_0(h).$$

Subtracting the two expressions and averaging over the shared history law proves Eq. (4).

For the second claim, consider two equally likely histories. Let $r_0(h_1) = r_0(h_2) = 0$, $r_1(h_1) = 1$, and $r_1(h_2) = 1/2$. Set $q_k(h; \pi) = 1/2$ at both histories and

$$q_k(h_1; \pi') = \frac{1}{4}, \qquad q_k(h_2; \pi') = \frac{7}{8}.$$

Then

$$q_k(\pi') - q_k(\pi) = \frac{1}{16} > 0, \qquad A(\pi') - A(\pi) = -\frac{1}{32} < 0.$$

The improvement at h_2 does not offset the loss at h_1, where local correctness has greater value for final success. □

A sufficient condition for nonnegative task gain is $q_k(h;\pi') \geq q_k(h;\pi)$ and $r_1(h) \geq r_0(h)$ at almost every history. The gain is strictly positive when both inequalities are strict on a set of positive d_k-measure. An increase in average quality alone is weaker than this pointwise condition.

A.4 Local sensitivity and gradients

We next vary a segment conditional while fixing the history law and downstream policy. Unlike the preceding comparison, output distributions within correctness classes may change.

Assumption 6 (Regularity for local gradients)**.** Only the segment conditional μ_λ varies with λ. The parameter ranges over an open interval, and the conditionals have positive densities on a common support with respect to a fixed reference measure. The densities are differentiable with locally integrably dominated derivatives, allowing differentiation under the conditional and outer integrals. For discrete outputs, densities are probability masses.

Define the conditional continuation advantage as

$$\Delta_{k,\lambda}(h,y) = v_k(h,y) - \int v_k(h,u)\,\mu_\lambda(du \mid h).$$

Lemma 7 (Segment sensitivity)**.** *Under Assumption 6,*

$$\frac{\partial}{\partial\lambda}\mathbb{E}_{\pi_{\mu_\lambda}}[R] = \mathbb{E}_{\pi_{\mu_\lambda}}\left[\Delta_{k,\lambda}(H_k,Y_k)\frac{\partial}{\partial\lambda}\log\mu_\lambda(Y_k \mid H_k)\right]. \tag{13}$$

Proof. Differentiating the conditional expectation and using the score identity gives

$$\frac{\partial}{\partial\lambda}\mathbb{E}[R \mid H_k] = \mathbb{E}\left[v_k(H_k,Y_k)\frac{\partial}{\partial\lambda}\log\mu_\lambda(Y_k \mid H_k)\,\middle|\,H_k\right].$$

The conditional score has mean zero. Subtracting the mean continuation value at the same history therefore leaves the expectation unchanged. Averaging over d_k proves the result. □

Under outcome insensitivity, $\Delta_{k,\lambda} = 0$ almost surely, so the local derivative vanishes. The converse need not hold for a single perturbation direction. Applying the score identity to the checker label instead gives

$$\nabla_\theta J_k(\mu_\theta; d_k) = \mathbb{E}_{d_k\mu_\theta}[Z_k\nabla_\theta\log\mu_\theta(Y_k \mid H_k)],$$

where $J_k(\mu_\theta; d_k)$ denotes the local objective with the history law held fixed. These identities compare the signals supplied by final correctness and local correctness. They are not full gradients of a shared policy whose earlier segments also change.

A.5 Shared parameters and the GRPO update

Full-policy gradients. For a fixed collection of K segments, let $s_j = \nabla_\theta\log\pi_\theta(Y_j \mid H_j)$ denote the segment score. Assume the trajectory distribution has a differentiable density on a common support and satisfies the domination conditions needed to interchange differentiation and expectation. The full-policy gradients are

$$\nabla_\theta A(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[R\sum_{j=1}^{K} s_j\right], \qquad \nabla_\theta q_k(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[Z_k\sum_{j=1}^{k} s_j\right]. \tag{14}$$

The first identity follows by differentiating the full trajectory likelihood. For the second, Z_k depends only on the history through segment k. For $j > k$, it is measurable with respect to H_j, while $\mathbb{E}[s_j \mid H_j] = 0$. All later score terms therefore vanish in expectation.

The local objective can change through both the distribution of earlier histories and the segment conditional itself. Because the scores share parameters, removing a segment's loss does not freeze the model's behavior on that segment.

The masked surrogate. Consider the empirical GRPO objective in Eq. (10). Hold the sampled trajectories, advantages, and active token sets fixed during differentiation. At $\theta = \theta_{\text{old}}$, every likelihood ratio equals one and clipping is locally inactive. Its gradient is

$$\widehat{g}_{\mathcal{I}} = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{I}_i|}\sum_{t\in\mathcal{I}_i} A_i \nabla_\theta \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})\Big|_{\theta=\theta_{\text{old}}}. \tag{15}$$

Here each active set is nonempty. This is the gradient of the sampled surrogate at the rollout policy, not a claim that it equals the full gradient of task accuracy or local quality. Group normalization, token averaging, masking, and subsequent clipped updates must be distinguished from the fixed-history score identities above.

If all rewards in a group are equal, their centered advantages vanish under the usual zero-variance convention. The group then contributes no reward-driven gradient to this surrogate. Trajectories with identical final correctness can nevertheless have different local labels. A local reward can provide an additional distinction when it makes their combined rewards different. Variation in a local label alone is insufficient if other reward terms cancel that variation.

In Stage I, the initial-answer and verification rewards are combined into a trajectory-level reward, and the resulting advantage is shared across unmasked tokens. This supplies role-related feedback but does not assign an independent advantage to each ability. Stage II changes the reward and includes revision tokens in the loss. Both stages update the shared policy.

A.6 Selective imitation targets

For Selective Loss Masking, consider a fixed demonstration distribution p over discrete segments. The weighted imitation loss is

$$\mathcal{L}_w(\pi) = -\mathbb{E}_p\left[\sum_{k=1}^{K} w_k \log \pi(Y_k \mid H_k)\right].$$

Uniform weights include incorrect segments even when the final answer is correct. Setting $w_k = Z_k$ removes those segments from the loss while retaining them in the history for later predictions. Masked terms contribute zero even when the model assigns zero probability to the segment.

Assumption 8 (Coverage and realizability of imitation targets)**.** For every k, $p(Z_k = 1 \mid H_k) > 0$ for p-almost every history. Both $p(\cdot \mid H_k)$ and $p(\cdot \mid H_k, Z_k = 1)$ have finite expected conditional entropies under the demonstration history law. Each target family can be matched simultaneously across all segments by a policy in the class. The policies realizing the two families may differ.

Proposition 9 (Population targets of selective imitation)**.** *Under Assumption 8, every population minimizer with uniform weights matches $p(\cdot \mid H_k)$, whereas every minimizer with $w_k = Z_k$ satisfies*

$$\pi^\star(\cdot \mid H_k) = p(\cdot \mid H_k, Z_k = 1)$$

for p-almost every history and every k.

Proof. The uniform loss decomposes into conditional entropy and a nonnegative KL term as

$$\mathcal{L}_{\mathbf{1}}(\pi) = \sum_{k=1}^{K} \mathbb{E}_{H_k\sim p}\big[\mathrm{H}\big(p(\cdot \mid H_k)\big) + \mathrm{KL}\big(p(\cdot \mid H_k) \,\|\, \pi(\cdot \mid H_k)\big)\big].$$

For selective weights, write $s_k(h) = p(Z_k = 1 \mid H_k = h)$ and $p_k^+(\cdot \mid h) = p(\cdot \mid H_k = h, Z_k = 1)$. Conditioning on the history gives

$$\mathcal{L}_Z(\pi) = \sum_{k=1}^{K} \mathbb{E}_{H_k\sim p}\big[s_k(H_k)\big\{\mathrm{H}\big(p_k^+(\cdot \mid H_k)\big) + \mathrm{KL}\big(p_k^+(\cdot \mid H_k) \,\|\, \pi(\cdot \mid H_k)\big)\big\}\big].$$

For each loss, the entropy contribution is finite and independent of π. Joint realizability allows all KL terms to vanish simultaneously. Every minimizer must therefore make them zero almost surely. Since $s_k(H_k) > 0$, this yields the stated targets. □

The selective loss gives no direct constraint where $s_k = 0$ or outside demonstration support. The result therefore does not determine quality under the learned policy's own history distribution. For an optional segment, the same argument applies conditional on reaching it in the demonstrations. Its positive, fixed reach probability only scales the loss. Correctness throughout this result is defined by the available checker, including verdict correctness for verification units.

A.7 Accuracy of a single GVR round

Let $C \in \{0, 1\}$ indicate initial-answer correctness and let $B = \mathbf{1}\{v^{(0)} = \mathrm{F}\}$ indicate whether revision is triggered. For a single GVR round, $R = C$ when $B = 0$. Define

$$p_g = \Pr(C = 1), \qquad \rho = \Pr(B = 1 \mid C = 0), \qquad \varepsilon = \Pr(B = 1 \mid C = 1),$$
$$p_r = \Pr(R = 1 \mid C = 0, B = 1), \quad q_h = \Pr(R = 0 \mid C = 1, B = 1).$$

Here ρ and ε measure error detection and false alarms. The rates p_r and q_h measure successful repair and corruption conditional on revision. Rates conditioned on events of zero probability may take any value in $[0, 1]$, since those branches contribute zero to accuracy.

Proposition 10 (Accuracy of a Generate, Verify, and Revise trajectory)**.** *For one verification and revision round,*

$$A - p_g = (1 - p_g)\rho p_r - p_g \varepsilon q_h, \qquad \omega_r = \Pr(C = 0, B = 1) = (1 - p_g)\rho. \tag{16}$$

For a second policy, denote its rates by primes. If $\rho > 0$ and $p'_g < 1$, the probability mass of incorrect answers reaching revision increases exactly when

$$\frac{\rho'}{\rho} > \frac{1 - p_g}{1 - p'_g}. \tag{17}$$

Proof. Partitioning trajectories by initial correctness and whether revision occurs gives

$$\begin{aligned} A &= p_g(1 - \varepsilon) + p_g \varepsilon(1 - q_h) + (1 - p_g)\rho p_r \\ &= p_g - p_g \varepsilon q_h + (1 - p_g)\rho p_r. \end{aligned}$$

These products use conditional probabilities and require no independence assumption. The identity for ω_r follows from its definition. Thus $\omega'_r > \omega_r$ exactly when $(1 - p'_g)\rho' > (1 - p_g)\rho$. Dividing by $\rho(1 - p'_g)$ under the stated positivity conditions gives Eq. (17). The product inequality also applies at the boundaries. □

The net gain consists of successful repairs minus corruption of initially correct answers. Detecting an error is useful only when the subsequent revision can repair it. Conversely, repair ability has no effect on an incorrect answer that is accepted without revision. Holding the other rates fixed gives $\partial A / \partial p_r = (1 - p_g)\rho$. Better generation can reduce this mass even when error detection improves.

The identity holds separately for each policy, so its terms can be measured at different training checkpoints without assuming that other abilities remain fixed. Differences between checkpoints are descriptive, however, and do not isolate the causal effect of any one training component. The identity supports evaluating generation, verification, and revision together. It does not select an optimal order for training them.

A.8 Scope of the analysis

The non-identifiability result establishes a possible limitation of final-answer supervision, not its prevalence or magnitude in a particular model. It does not imply that outcome-based training cannot improve local abilities. The local comparisons isolate dependencies under fixed policy components, while the imitation result concerns population targets on demonstrated histories. Neither guarantees improvement after a shared-parameter update. The GRPO analysis describes the information and token contributions in the sampled surrogate, rather than convergence or monotonic improvement. Whether the proposed stage schedule is preferable to joint training is an empirical question. The GVR identity concerns one verification and revision round and does not establish gains from additional rounds.

B EXPERIMENTAL DETAILS

Hardware and Software Platform. We conduct supervised fine-tuning (SFT) on servers with eight NVIDIA A800-SXM4 GPUs, each with 40GB of memory, using LLaMA-Factory (Zheng et al., 2024). Reinforcement learning (RL) uses servers with eight NVIDIA H20 GPUs and the Verl (Sheng et al., 2025) framework.

SFT Training Configuration. We fine-tune all model parameters for one epoch with a maximum sequence length of 14,000 tokens. Training uses a per-device batch size of 1 with gradient accumulation over 8 steps. We use Adam with a learning rate of 1×10^{-6}, a cosine learning rate schedule, and a warmup ratio of 0.1. Training uses bf16 precision and DeepSpeed ZeRO-3.

RL Training Configuration. We train with GRPO for 200 optimization steps, sampling $G = 8$ responses per problem at temperature $\tau = 1.0$ and enforcing end-of-sequence tokens. Rollout generation uses vLLM with paged attention and 128 rollout slots for batched decoding. The maximum prompt and response lengths are 1,024 and 8,000 tokens, respectively.

The actor model is optimized with Adam at a learning rate of 1×10^{-6} and a batch size of 64. The policy optimization clipping range is $[0.01, 0.99]$. Reward definitions and coefficients for both stages are given in Appendix C.3.

Inference Configuration. We use vLLM for decoding with a maximum generation length of 24,000 tokens. The temperature is 0.6 for AIME24, AIME25, and AMC, and 0 for all other benchmarks. The top-p parameter is set to 0.95.

Training Data. The *SFT + GRPO* baseline uses long chain-of-thought trajectories distilled from correct Qwen3-32B solutions to MATH problems at levels 3 to 5. To control for data scale, we use 7K SFT samples across all methods. All RL experiments use the same DAPO-MATH dataset for a fair comparison.

Stage	Settings	Math	AIME24	AIME25	Olympiad	AMC	AVG
SCR-Stage I	$\beta = 1.0, \gamma = 1.0$	65.20	4.33	2.00	29.04	29.16	25.95
	$\beta = 0.5, \gamma = 1.0$	40.20	1.33	0.33	13.48	10.24	13.12
	$\beta = 1.0, \gamma = 0.5$	65.00	6.33	4.00	30.81	32.53	27.73
SCR-Stage II	$\beta' = 1.0, \gamma' = 1.0$	70.40	7.33	4.00	37.03	36.02	30.96
	$\beta' = 1.0, \gamma' = 2.0$	68.20	7.67	4.00	32.30	37.11	29.86
	$\beta' = 2.0, \gamma' = 1.0$	68.40	7.00	5.33	32.89	34.94	29.71

Table 3: **Impact of reward weight coefficients on performance during Stage I and Stage II training.**

C TRAINING DETAILS FOR SCR

C.1 DATA CONSTRUCTION

Initial solution sampling. For each problem, we sample four solutions from each of Qwen2.5-3B, Qwen2.5-7B, and Qwen2.5-14B, yielding twelve candidates. We select equal numbers of candidates with correct and incorrect answers before applying the quality filters in Section 3.2.2. This balance applies to the candidate pool rather than the final dataset.

Teacher and retained data. GPT-4.1 with model identifier `gpt-4.1-2025-04-14` generates both critiques and revisions. The critique and refinement prompts appear in Prompts 4 and 5. The synthesis and filtering procedure in Section 3.2.2 yields approximately 7K trajectories. Trajectories with correct and incorrect initial answers appear in Examples 2 and 1, respectively.

C.2 Supervised Training Details

Serialization. The problem is the conditioning input x, and each trajectory is serialized as one response following the format in Prompt 3. Each solution block contains its final answer inside `\boxed{}`, and each critique ends with its verdict before `</critic>`. Termination follows the DTS rule in Section 3.2.2.

Mask application. Tag boundaries locate the incorrect initial solution to be masked in Eq. (6). The mask affects only the loss, not the input sequence or causal attention. The critique and revision therefore retain access to the full initial solution. The final-answer check validates the retained correction, not intermediate attempts during synthesis. Any additional answer segment used as an imitation target would require its own correctness check. SFT hyperparameters are listed in Appendix B.

C.3 Reward and Optimization Details

Trajectory probability. For the trajectory with one verification and revision round in Section 3.2.1, the probability under the shared policy factorizes as

$$\pi_\theta(\boldsymbol{y} \mid x) = \pi_\theta(y^{(0)} \mid x)\pi_\theta(c^{(0)} \mid x, y^{(0)}) \begin{cases} \pi_\theta(y^{(1)} \mid x, y^{(0)}, c^{(0)}), & v^{(0)} = \mathrm{F}, \\ 1, & v^{(0)} = \mathrm{T}. \end{cases} \tag{18}$$

Format checks. The format reward in Section 3.2.3 requires all five binary checks C_i below to pass.

- C_1 The response contains complete `<answer>` and `<critic>` tag pairs.
- C_2 The critique ends with a verdict of T or F.
- C_3 A verdict of F is followed by a complete `<revised>` block, while T ends the response without revision.
- C_4 The final answer in each solution block is enclosed in `\boxed{}`.
- C_5 The blocks appear in the order of initial solution, critique, and any required revision.

Correctness checks. We use the same task checker to compare the extracted initial and final answers with the reference. The verification reward evaluates whether the verdict matches initial-answer correctness, not whether every statement in the critique is valid. For trajectories with additional revision rounds, the revision reward compares the initial and final answers without scoring intermediate attempts separately.

Reward coefficients. For the total rewards in Eqs. (9) and (12), we use $\alpha = \alpha' = 0.1$ and $\beta = \beta' = \gamma = \gamma' = 1.0$. The revision coefficients are $\mu_1 = -0.1$, $\mu_2 = -0.3$, and $\mu_3 = \mu_4 = -0.5$.

Optimization. Advantages in Eq. (10) are computed by subtracting the group mean reward and dividing by the group standard deviation. Groups with identical rewards provide no relative preference among trajectories. Unlike SLM, the Stage I mask excludes revision tokens rather than incorrect initial solutions. Both RL stages update the shared policy, so masking a segment does not freeze a separate component. Optimization and rollout hyperparameters are listed in Appendix B.

D Pilot Study Details

In §3.1, we analyze long-CoT trajectories from DeepSeek-R1-Distill-Qwen-7B[1], DeepSeek-R1-Distill-Qwen-1.5B[2], DeepSeek-R1-Distill-Llama-8B[3], and Qwen3-32B[4]. We evaluate all four models on MATH500 with deterministic decoding at temperature 0.

We characterize reasoning behavior using an operator taxonomy grounded in Information Processing Theory (Simon, 1978). GPT-4.1-2025-04-14 (OpenAI, 2025) assigns operator labels and counts occurrences according to the units specified in the annotation prompt. Figure 2 reports verification and revision jointly, alongside decomposition and planning, causal deduction, and representational

[1] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B
[2] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B
[3] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B
[4] https://huggingface.co/Qwen/Qwen3-32B

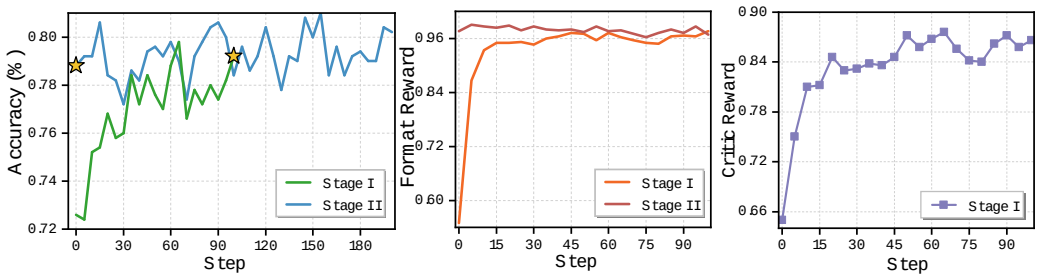

Figure 7: **Training dynamics of Qwen2.5-7B-Instruct.** From left to right, we report validation accuracy, format score, and self-evaluation reward over training.

restructuring. These counts describe behavioral frequency rather than token usage or computational cost. The operator definitions and annotation prompts appear in Prompts 6 and 7.

For the revision analysis, we extract the sequence of answers leading to the final prediction and compare the answers before and after each revision with the reference. Each event is assigned one of four correctness transitions, T→T, T→F, F→F, or F→T. Here T and F denote answer correctness rather than verification verdicts. The proportions in Figure 2 use all revision events as the denominator.

These proportions differ from conditional success and corruption rates. The success rate conditional on revising an incorrect answer is the number of F→T events divided by the total number of F→T and F→F events. The corruption rate is the number of T→F events divided by the total number of T→F and T→T events. These rates correspond to p_r and q_h in Proposition 10. The audit measures observed answer transitions rather than verification accuracy or the effect of a particular training objective.

E Further Analysis

E.1 Reward Weight Analysis

Table 3 compares reward-weight settings for Qwen2.5-3B-Instruct. We keep the format weight fixed and vary the remaining coefficients in Eqs. (9) and (12).

Weakening the generation reward is detrimental in Stage I. Reducing the initial-answer accuracy weight lowers performance on every benchmark. Reducing the verification weight instead improves average performance, with gains on all tasks except MATH500. The verification reward credits correct rejection even when the initial answer is wrong, so satisfying this objective does not necessarily require better generation. These results suggest that verdict supervision should complement, rather than substitute for, pressure to generate correct initial solutions.

Larger Stage II reward weights do not yield consistent gains. Increasing either the final-answer accuracy weight or the revision weight lowers average performance, although each benefits individual benchmarks. Both changes reduce accuracy on MATH500 and Olympiad, and the default setting gives the best average result among the tested configurations. Since all revision coefficients are negative, increasing the revision weight strengthens penalties even for successful corrections. It therefore changes the incentive to revise rather than simply placing more emphasis on successful correction.

E.2 Training Dynamics

Figure 7 tracks validation accuracy, format score, and verification reward during RL on Qwen2.5-7B-Instruct. **Early reward improvements are not matched by equally smooth accuracy gains.** During Stage I, format adherence and verification reward rise rapidly at the beginning of training and subsequently vary within a narrower range. Validation accuracy follows a less regular trajectory. Stage II starts with high format adherence but continues to show fluctuations in validation accuracy. Thus, stabilization of the local rewards should not be taken as evidence that reasoning performance has converged.

E.3 Failure Modes and Supervision Limits

Beyond the verdict errors discussed in the main text, the reward definitions leave two potential gaps in self-correction.

A correct verdict need not provide a useful diagnosis. Stage I rewards agreement between the verdict and initial-answer correctness, but does not check whether the critique accurately locates the error. A correct rejection can therefore receive full verification reward even when its explanation is incomplete or misleading. Such a critique may provide little useful guidance for revision.

Answer-level rewards do not establish whether revision uses the critique. Stage II evaluates initial and final answer correctness without directly checking how the revised derivation responds to the diagnosis. A revision may fail to use a valid diagnosis or reach a correct answer through a different derivation. The reported metrics cannot distinguish faithful use of a critique from independent re-solving. Establishing this distinction would require trajectory-level evaluation of diagnosis quality and its use in revision.

Limitations

Our framework supports multi-round verification and revision, but our experiments are restricted to two correction rounds. This provides a minimal setting for evaluating critique and revision. Extending to more rounds requires high-quality correction trajectories, whose synthesis cost increases with the number of rounds. We leave more efficient data construction for future work.

Our pilot study uses an operator taxonomy grounded in Information Processing Theory. It captures the dominant roles in the sampled trajectories, but finer-grained behaviors such as analogy and abstraction are not modeled separately.

F Prompt Templates

We provide two training examples and five prompt templates used in our experiments.

- Example 1 illustrates a correction trajectory in which an incorrect initial answer is revised.
- Example 2 illustrates a correct-answer trajectory that terminates after verification.
- Prompt 3 specifies the structured output format used by GVR.
- Prompt 4 guides the teacher model to generate structured critiques.
- Prompt 5 guides the teacher model to refine incorrect solutions using critique feedback.
- Prompt 6 defines and counts cognitive operators in reasoning trajectories.
- Prompt 7 extracts the sequence of answer-level states for revision analysis.

Example 1. Correction Trajectory in SCR

/Input**/**

1. People numbered $1, 2, \cdots, 2015$ are arranged in a line, and a position-swapping game is played among them, with the rule that each swap can only occur between adjacent individuals. Now, the person numbered 100 and the person numbered 1000 are to swap positions, with the minimum number of swaps required being times.

/Output**/**

<answer>

Let's reason through this problem:

We need to find the minimum number of swaps needed for person 100 and person 1000 to swap their positions.

The key here is understanding that every swap moves two people forward or backward by one place. Specifically, we want to move person 100 and person 1000 relative to each other by 900 positions (from 100 to 1000). Since each swap affects two positions at once,

(... more reasoning tokens omitted for brevity)

Considering efficiency and the simplicity of the problem, we calculate:

$$\text{Minimum swaps required} = 900 + (\text{Efficiency savings})$$

Given our targeted statement on "minimum", we assume efficiency yields ideal movement, thus aligning our answer closely with Calculation 1:

$$\boxed{900}$$

</answer>

<critic>

The key mistake is that the answer incorrectly equates the number of swaps with the difference in the initial positions of the two individuals.

This is wrong because it only accounts for the movement of one person to the other's position, ignoring the movement required for the second person. To swap person A (at position 100) and person B (at position 1000), *(... more reasoning tokens omitted for brevity)*

F

</critic>

<revised>

To improve the answer, it's important to recognize the full process for swapping two individuals who are not adjacent.

(... more reasoning tokens omitted for brevity)

This is a distance of 899 positions, requiring 899 swaps. Each swap moves person 1000 one spot towards position 100.

Thus, the correct minimum number of adjacent swaps required is the sum:

$$900 \text{ (to move 100 to 1000)} + 899 \text{ (to move 1000 to 100)} = 1799$$

$$\boxed{1799}$$

</revised>

Example 2. Correct-Answer Trajectory in SCR

/Input**/**

The lengths in feet of three pieces of timber are 48, 72, and 40. The sawmill operator needs to cut the timber into logs of equal length with no waste. How many feet long is the greatest possible length she can cut?

/Output**/**

<answer>

To find the greatest possible length that can be used to cut the timber into equal pieces with no waste, we need to find the greatest common divisor (GCD) of the lengths of the three pieces of timber. The GCD is the largest number that divides all three lengths without leaving a remainder.
The lengths of the timber are 48, 72, and 40 feet.
First, we find the prime factorization of each length:

- $48 = 2^4 \cdot 3$
- $72 = 2^3 \cdot 3^2$
- $40 = 2^3 \cdot 5$

Next, we identify the common factors among the prime factorizations. The common factors are 2^3 (the smallest power of 2 present in all three factorizations) and 1 (since there are no other common factors).
Therefore, the GCD of 48, 72, and 40 is

$$2^3 \cdot 1 = 8$$

So, the greatest possible length she can cut is 8 feet.

$$\boxed{8}$$

</answer>

<critic>

The answer is correct.
The model correctly identifies that the problem requires finding the greatest common divisor (GCD) of the three lengths. The calculation of the GCD using prime factorization is accurate. **T**

</critic>

Prompt 3. Structured Output Format for SCR

You are a helpful AI assistant.

For each question, you must first solve the problem and put your complete solution inside <answer>...</answer>. The final result must be wrapped in *boxed{}*.

Then you must evaluate your own solution inside <critic>...</critic>. In this block, you must first give your evaluation and reasoning. At the end of the <critic>, you must give the final judgment using only one single symbol: T or F. T means the answer is correct. F means the answer is incorrect.

If the final judgment is F, you must give a corrected solution inside <revised>...</revised>, and the final result must also be wrapped in *boxed{}*.

If the final judgment is T, you must stop and give no further output.

Prompt 4. Teacher Prompt for Structured Critiques

/* System Prompt */

You are a careful and strict evaluator.
Given a question and a model-generated answer, judge whether the answer is correct.
Evaluation rules:
- Focus on correctness, not writing style.
- If the answer is correct, briefly explain why.
- If the answer is incorrect:
- Clearly state the key mistake.
- Explain why it is wrong.
- Give a short, actionable suggestion to fix it.
- Be concise and precise.
- Do NOT restate the full answer or redo the full solution.

End your response with a single letter on a new line:
T if the answer is correct.
F if the answer is incorrect.

/* User Prompt */

Question: {here is the question}
Answer: {here is the answer}

Prompt 5. Teacher Prompt for Solution Refinement

/* System Prompt */

You are a helpful AI assistant.
You will be given a question, an initial answer, and evaluation feedback on that answer.

Your task is to use the evaluation feedback to modify the initial answer so that the result becomes correct.
Describe the modification process in a natural and explanatory way.
Explain what needs to be corrected and how the correction is carried out, as part of the reasoning.
Do not use explicit section titles or meta descriptions such as “refinement process” or “final answer”.
Do not refer to the evaluation itself.
After completing the modification and reasoning, present the corrected final result.
Wrap the final answer in `\boxed{...}`.
/* User Prompt */
Question: {here is the question}
Answer: {here is the answer}
Evaluation: {here is the evaluation feedback}

Prompt 6. Cognitive Operator Counting

You are a **Cognitive Operator Analyzer & Counter**. The user will provide a QA pair. Your task has **TWO PHASES**:

Phase 1: Detailed Analysis - Analyze the Answer (A) in detail, identifying occurrences of different cognitive operators. - You may explain and reason freely. - Identify where and how each operator appears.

Phase 2: Final Operator Count (Strict Output) - After analysis, output a clean operator count summary in strict format for automated parsing.

Operators to Count:

1) **DECOMP_PLAN (Decomposition & Planning Operator)**: One coherent planning episode; sub-steps do not count separately. Multiple independent plans count separately.
2) **CAUSAL_INFER (Causal Inference Operator)**: One per causal chain; multiple connectors in the same chain count as one; new independent chains count separately.
3) **MONITOR (Monitoring Operator)**: Each evaluation or self-check counts as one instance.
4) **BACKTRACK (Backtracking / Step-Revision Operator)**: Each revision or correction of a reasoning step counts once.
5) **REPR_REFRAME (Representation Reframing Operator)**: Each distinct reframing or perspective shift counts once.

Counting Rules: - Analyze only the Answer (A), ignore the Question (Q). - Count each operator action once. - Multiple operators can appear in the same sentence if distinct. - Assign each instance to the most dominant operator. - Do not include explanations in the final output.

Final Output Format (Mandatory): Your output must contain only the following five lines:

```
COUNT_DECOMP_PLAN:<int>
COUNT_CAUSAL_INFER:<int>
COUNT_MONITOR:<int>
COUNT_BACKTRACK:<int>
COUNT_REPR_REFRAME:<int>
```

Prompt 7. Answer-Level Trajectory Extraction

You are an **"Answer-Level Trajectory Recorder"**.
The user will provide a QA pair. Your goal is to analyze **ONLY the Answer (A)** and extract the full trajectory of **ANSWER-LEVEL STATES** for the **TARGET** quantity asked by the question. This task analyzes how the model's ANSWER changes over time.
Phase 1: Analysis (Free-Form) - First, read the Question (Q) ONLY to identify the **TARGET** quantity (what the question ultimately asks the model to answer). - Then analyze the Answer (A) and reason about how the model treats different values or expressions as the ANSWER to that TARGET over time. - In this phase, distinguish answer-level commitments from intermediate computations, and identify moments where the model:

– proposes a candidate final answer

– temporarily commits to an answer

– revises or replaces a previously stated answer

– re-confirms or restates the same answer

Phase 2: Final Output (Strict) - After the analysis, output the **ANSWER-LEVEL TRAJECTORY** in chronological order.
What Counts as an Answer-Level State An answer-level state MUST satisfy BOTH conditions:

(1) **Target Alignment:** Refers to the TARGET quantity asked by Q (e.g., "the value of f(-2)+f(-1)+f(0)", "the final common fraction", "the correct option").

(2) **Answer Commitment:** The model presents it as an answer, conclusion, or result for the TARGET, including:

– "the answer is . . . "

– "the value is . . . "

– "thus . . . equals . . . "

– "final answer: . . . "

– an explicit restatement or confirmation of a previously given answer

Important: A statement counts even if it later turns out to be wrong.

What Must Be Filtered Out Do NOT extract:

– Sub-results used only to compute the TARGET (e.g., f(-2)=2, f(-1)=5/3, f(0)=1)

– Pure intermediate calculations or simplification steps

– Values not presented as the answer to the TARGET

Even if boxed or emphasized, they are NOT answer-level states unless treated as the answer.
Trajectory Rules

– Extract answer-level states strictly in the order they appear in A.

– Do NOT merge, deduplicate, or normalize answers.

– If the same answer is stated or confirmed multiple times, record EACH occurrence separately.

– If multiple different candidate final answers appear, record ALL of them.

– Do NOT invent answer states that do not explicitly appear in A.

Final Output Format (Mandatory) Your output must contain ONLY **boxed answer-level states**, one per line. No explanations, headers, numbering, or blank lines.

```
\boxed{}
```