SCOPE-RL: Optimizing Reasoning Paths Before and After Success

Anonymous ACL submission

Abstract

Reinforcement learning with verifiable rewards (RLVR) optimizes LLMs using sparse verifiable final-answer rewards. This *sparse anchor* reliably verifies whether a trajectory succeeds but provides no direct feedback on the reasoning path that produced it. **Before success**, prerequisite progress on hard problems receives no reward signal; **after success**, outcome rewards cannot distinguish well-organized correct trajectories from redundant or locally flawed ones. We introduce **SCOPE-RL** (Scaffolded Chain Optimization with Process Efficiency), a two-stage framework that densifies this anchor while retaining the GRPO update: **Adaptive Scaffolded RL** adds prefix-decomposed verifiable rewards on answer-hidden sub-question chains before success, and **Quality-Aware Process RL** applies correctness-gated process-shape rewards to refine correct trajectories after success. An expert-validated Step-Quality Evaluation Protocol evaluates useful-step density, error localization, and token efficiency beyond final-answer accuracy. On Qwen3-8B-Instruct trained on DAPO-Math and Big-Math, SCOPE-RL improves average accuracy by up to 11.2 pp and reduces reasoning tokens by up to 27.1% over outcome-only GRPO; the gains hold under GSPO and on Qwen3-0.6B-Instruct, indicating that reward-signal densification is complementary to policy-update-level RLVR advances.

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) optimizes LLMs using sparse terminal rewards, typically final-answer correctness (Guo et al., 2025; Shao et al., 2024; Yu et al., 2025; Zheng et al., 2025; Liu et al., 2025). This terminal anchor reliably certifies success but provides no direct feedback on the reasoning path, leading to two failure modes usually treated separately: training stalls on hard problems where outcome-only rollouts rarely succeed (Pang et al., 2025; Shi et al., 2025; Ma et al., 2025), and models produce correct but inefficient reasoning on problems they can already solve (Zhou et al., 2026; Rakotonirina et al., 2026; He et al., 2025). **Before success**, prerequisite progress is invisible unless it reaches the final answer, so near-misses provide little signal; **after success**, endpoint rewards cannot distinguish well-organized correct trajectories from redundant or locally flawed ones. Two diagnostic probes provide evidence for this interpretation (Figure 1; details in Appendix C): scaffolded prompts reveal prerequisite signal that endpoint rewards leave unexposed, and among final-answer-correct rollouts outcome-only GRPO increases response length while retaining low-value content.

This sparse-anchor view reframes prior work as addressing different consequences of the same anchor: exploration-oriented methods reshape inputs, rollouts, or updates on hard problems (Parashar et al., 2025; Pang et al., 2025; Bae et al.; Shi et al., 2025; Chih-Yao Chen et al., 2026; Simonds and Yoshiyama, 2025; Qu et al., 2026; Liao et al., 2026; Zhang et al., 2025; Liu et al.; Yan et al., 2026), efficiency methods reduce overthinking (Hou et al., 2025; He et al., 2025; Rakotonirina et al., 2026), and process-supervision methods train PRMs, derive implicit rewards, or reassign dense feedback (Lightman et al., 2024; Wang et al., 2024; Cui et al., 2025; Rezaei et al., 2026; Yao et al., 2026; Cheng et al., 2026). Less explored is how the rule-verifiable signal itself can be densified through independently verified sub-answer targets before success and correctness-gated process composition after success.

We instantiate this view as **SCOPE-RL** (Scaffolded Chain Optimization with Process Efficiency), a two-stage RLVR framework that densifies the sparse anchor while retaining the GRPO update. **Adaptive Scaffolded RL** (ASR) targets the before-success phase: for problems where outcome-only rollouts yield low success rates, it

converts answer-hidden scaffolded sub-question chains into prefix-decomposed verifiable rewards, crediting partial progress without leaking answers. **Quality-Aware Process RL** (QPR) targets the after-success phase: on trajectories that reach the correct answer, it applies correctness-gated process-shape rewards that prefer useful, well-organized reasoning over redundant or locally flawed reasoning. The stages are naturally sequential—ASR first raises the supply of verified trajectories, after which QPR refines their process quality—and by design neither allows answer leakage nor brevity to substitute for verified correctness. To evaluate reasoning processes beyond final-answer accuracy, we introduce a **Step-Quality Evaluation Protocol** that preserves rule-based answer verification while adding post-hoc LLM-judged diagnostics of useful-step density, low-value step ratios, error localization, and token efficiency.

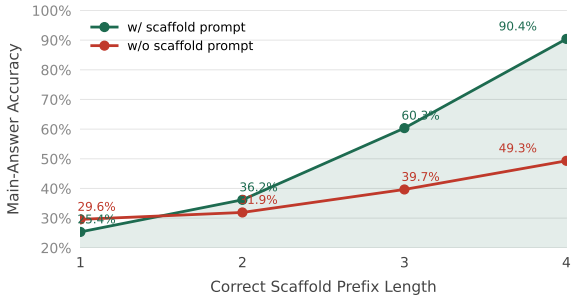
Across mathematical and scientific benchmarks and two training corpora (DAPO-Math (Yu et al., 2025) and Big-Math (Albalak et al., 2025)), SCOPE-RL improves accuracy, useful-step density, and token efficiency over outcome-only GRPO on **Qwen3-8B-Instruct** (Team, 2025) . Average accuracy rises from 55.80% to 66.35% on DAPO-Math and from 53.58% to 64.79% on Big-Math, with 16.2% and 27.1% token reductions, respectively. The gains hold under a GSPO backend (61.60%→66.93%) and on Qwen3-0.6B-Instruct (26.06%→32.06%; both verified on DAPO-Math), indicating that reward-signal densification is complementary to policy-update-level RLVR advances.

Contributions. (i) A unifying **sparse-anchor** view of RLVR: outcome-only verification leaves the reasoning path under-specified before and after success, calling for phase-specific densification around the same verified endpoint. (ii) **SCOPE-RL**, instantiating this view with prefix-decomposed verifiable rewards (ASR) and correctness-gated process-shape rewards (QPR) in a naturally sequential procedure. (iii) An expert-validated Step-Quality Evaluation Protocol, with consistent gains across two training sources and multiple benchmarks.

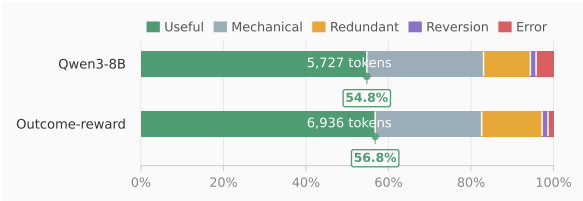
2 Preliminary

Given a prompt-answer pair (q, a), outcome-only RLVR samples $y \sim \pi_\theta(\cdot \mid q)$ and assigns the rule-verified reward

$$r_{\text{out}}(y, a) = \mathbf{1}[\hat{a}(y) = a]. \tag{1}$$



(a) Prerequisite scaffold progress vs. main-answer accuracy.



(b) Step-quality composition of correct reasoning trajectories.

Figure 1: Two probes of the same sparse anchor, one on each side of success. **(a) Before success**: the same problems are evaluated under a *scaffolded prompt* (sub-questions followed by the main problem) and the *original prompt*, both verified against the same ground-truth answer. Scaffold-prompt accuracy exceeds original-prompt accuracy and rises monotonically with the length of the correct sub-question prefix. **(b) After success**: among final-answer-correct responses, outcome-only GRPO produces longer responses that retain substantial low-value content.

More generally, GRPO admits any scalar reward $R(q, y)$ in place of r_{out}. For G sampled responses $\{y_g\}_{g=1}^{G}$, we use

$$A_g = \frac{R(q, y_g) - \text{mean}_j R(q, y_j)}{\text{std}_j R(q, y_j)}. \tag{2}$$

When the within-group reward is constant ($\sigma_R^2(q) = 0$), Eq. (2) is undefined; following standard GRPO implementations (Guo et al., 2025), we set $A_g \equiv 0$, so the group produces no reward-driven policy-gradient signal.

To quantify where GRPO receives usable reward variation, define

$$\sigma_R^2(q) = \text{Var}_g\, R(q, y_g), \tag{3}$$

$$\eta_R = \mathbb{P}_{q \sim \mathcal{D}}\big[\sigma_R^2(q) > 0\big]. \tag{4}$$

η_R measures the fraction of prompts that can produce non-zero group-relative advantages under reward R; for $r_{\text{out}} \in \{0, 1\}$, $\sigma_{r_{\text{out}}}^2(q) = 0$ exactly when the group is uniformly correct or uniformly wrong.

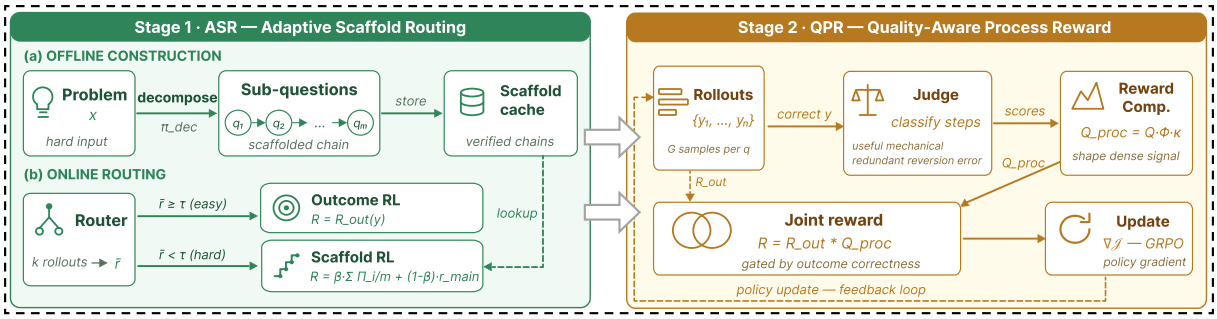


Figure 2: Overview of SCOPE-RL. Stage 1 routes hard problems to answer-hidden scaffolds for denser verifiable rewards, while Stage 2 applies correctness-gated process rewards to refine verified-correct trajectories.

3 The SCOPE-RL Framework

SCOPE-RL (**S**caffolded **C**hain **O**ptimization with **P**rocess **E**fficiency) instantiates a single design principle on GRPO: the policy update is left unchanged, and the sparse anchor is densified only by modifying *what enters the scalar reward* on each side of the success boundary.

Figure 2 illustrates the complete pipeline. Both stages follow the same reward-densification template—**where** new signal is placed, **when** it is triggered, and **how** it is composed. Stage 1, **Adaptive Scaffolded RL** (ASR), places verifiable signal at scaffolded sub-endpoints, triggers it through on-policy routing, and composes prefix-consistent rewards for weak-signal problems. Stage 2, **Quality-Aware Process RL** (QPR), places signal on already-correct trajectories, triggers it through a correctness gate, and composes bounded process-shape rewards.

3.1 Two-Stage Coupling

ASR and QPR are coupled by a sequential dependency. Because QPR's correctness gate (§3.3) returns zero on incorrect rollouts, it provides little signal on hard problems before the policy can reliably reach verified answers. ASR first lifts success rates through prefix-decomposed verifiable rewards, after which QPR has a denser set of correct trajectories to shape. Jointly mixing the two rewards would weaken this gate by applying QPR penalties before enough correct rollouts exist to provide useful contrast among them. We therefore train sequentially: Stage 1 produces a checkpoint θ_1 from the routed mixed reward R_1, and Stage 2 initializes from θ_1 and optimizes R_{QPR} on the original undecomposed problems. The only state passed across stages is the policy parameters.

3.2 Stage 1: Adaptive Scaffolded RL

Stage 1 addresses the before-success regime, where few rollouts reach the final answer and endpoint rewards provide little information about prerequisite progress. Since scaffold-prefix correctness is predictive of main-answer success (Figure 1a), ASR treats ordered scaffold prefixes as verifiable proxies for prerequisite progress, increasing supervision density through answer-hidden scaffold construction, on-policy scaffold routing, and prefix-consistent scaffold rewards (Figure 2, Stage 1).

3.2.1 Answer-Hidden Scaffold Construction

Scaffold construction turns hidden prerequisite progress into verifiable sub-endpoints, each checkable by the same rule-based verifier as the final answer. Construction is performed *offline*, with answer-hidden non-leakage enforced as a validity predicate (Eq. (6) below). The complete decomposition prompt template is provided in Appendix F.

Before RL training, ASR generates candidate scaffolded sub-question chains offline for each training problem. Given a problem q with answer a, a valid scaffold has the form

$$\mathcal{S}(q) = \{(q_1, a_1), \ldots, (q_m, a_m), (q_{\text{main}}, a)\}, \tag{5}$$

where q_{main} is the final-target sub-problem whose verifiable answer is the original problem's answer a, and each q_i targets a prerequisite concept needed for the final solution. During RL training, the policy observes only the ordered sub-questions and the main problem q_{main}; the sub-answers a_i are verifier-only and used solely for reward computation.

We cache a scaffold only when it satisfies

$$\text{Valid}(\mathcal{S}) = V(\mathcal{S}) \wedge P(\mathcal{S}) \wedge D(\mathcal{S}) \wedge H(\mathcal{S}), \tag{6}$$

where V denotes automatic verifiability, P prerequisite relevance, D dependency awareness, and H

answer-hidden non-leakage. These predicates filter out scaffolds that are unverifiable, irrelevant, dependency-inconsistent, or answer-leaking, preserving the verifiable-answer invariant of RLVR (the policy is rewarded only for answers it produces itself, not for answers that appear in the prompt) by construction. A human audit of 200 random scaffolds confirms strict prerequisite-chain integrity (Appendix N); invalid decompositions are discarded before RL training.

3.2.2 On-Policy Scaffold Routing

Applying scaffolds to every problem would dilute supervision and waste rollout cost on already-solvable problems. ASR routes groups by on-policy outcome reward, a self-adapting alternative to static difficulty labels or a learnable router.

During RL, each problem is first rolled out on the original prompt and scored with the outcome reward. The group mean outcome reward

$$\bar{r}(q) = \frac{1}{G}\sum_{g=1}^{G} r_{\text{out}}(y_g, a) \tag{7}$$

serves as an on-policy estimate of whether endpoint-only training provides sufficient positive signal for q. A problem is routed to the scaffolded path if

$$\bar{r}(q) < \tau, \tag{8}$$

where τ is a fixed routing threshold; otherwise, the original rollouts and outcome rewards are retained. We use $G = 8$ and $\tau = 0.5$ in all experiments, with sensitivity analysis in Appendix K.

3.2.3 Prefix-Consistent Scaffold Rewards

Once routing has selected a scaffold, the question is how to convert sub-answer matches into reward. We use a prefix product so that a sub-question contributes reward only when itself and all earlier sub-questions in its dependency chain are correct—this stays inside GRPO while preventing the policy from harvesting credit on later sub-answers it can guess without solving the prerequisites.

For routed problems, the scaffolded rollout asks the model to answer all sub-questions and then the original problem in a single generation, using explicit answer tags for extraction. Let $\hat{a}_i$ be the model's answer to q_i, and define the prefix-consistency indicator

$$\Pi_i = \prod_{j=1}^{i} \mathbf{1}[\hat{a}_j = a_j]. \tag{9}$$

The scaffold reward is

$$R_{\text{ASR}} = \beta \sum_{i=1}^{m} \frac{1}{m}\Pi_i + (1-\beta)\Pi_m \mathbf{1}[\hat{a}_{\text{main}} = a], \tag{10}$$

where β controls the weight assigned to scaffold checkpoints; we set $\beta = 0.5$ in our experiments. Since $\Pi_m = \prod_{j=1}^{m} \mathbf{1}[\hat{a}_j = a_j]$ requires all m sub-answers to be correct, valid prerequisite prefixes receive partial credit and the main-answer match contributes only when the entire scaffold prefix is satisfied.

Stage 1 therefore uses the mixed scalar reward

$$R_1(q, y) = \begin{cases} R_{\text{ASR}}(y, \mathcal{S}(q)), & \bar{r}(q) < \tau, \\ r_{\text{out}}(y, a), & \bar{r}(q) \geq \tau. \end{cases} \tag{11}$$

Here, y denotes the rollout used in the selected branch: scaffolded rollouts for routed groups and original-prompt rollouts otherwise.

Optimizing ASR with GRPO. Routed groups are re-rolled out on cached scaffolded prompts and scored with R_{ASR}; non-routed groups keep their original rollouts and outcome rewards.

ASR enlarges the effective gradient support. The mixed reward R_1 can introduce effective gradient support in routed groups that are degenerate under r_{out} (Eq. (4)). In particular, routed scaffold rewards produce non-zero advantages whenever an outcome-uniform group contains prefix-level disagreement, consistent with the gap between GRPO and SCOPE-RL Stage 1 in Figure 3a.

3.3 Stage 2: Quality-Aware Process RL

Stage 2 addresses the after-success regime, where final-answer rewards cannot distinguish concise, useful, and well-organized correct trajectories from redundant or locally flawed ones. We use *process shape* to denote the compositional structure of an outcome-correct trajectory, including useful-step density, low-value-step ratios, local flaws, and soft conciseness. QPR returns to the original undecomposed problems and applies these preferences only after correctness is established (Figure 2, Stage 2).

3.3.1 Correctness-Gated Process-Shape Reward

Any process-shape signal that fires on incorrect rollouts could encourage short or fluent wrong answers, reproducing the failure mode of naive length penalties (Figure 1b). QPR prevents this with a

multiplicative correctness gate: the rule-based verifier decides final-answer correctness, and the LLM Judge only annotates how a verified-correct trajectory was reached. QPR is initialized from θ_1 and trains on the original, undecomposed problems. Concretely, for each rollout y,

$$R_{\mathrm{QPR}}(y, a) = \begin{cases} 0, & \hat{a}(y) \neq a, \\ Q_{\mathrm{process}}(y), & \hat{a}(y) = a. \end{cases} \quad (12)$$

The judge model, prompt, and reward formula are fixed throughout QPR training.

3.3.2 Step-Level Category Annotation

Conditioned on the gate in Eq. (12), QPR uses a fixed LLM Judge to annotate step-level role among correct trajectories. Unlike a learned PRM, the Judge is not trained against the policy and never decides final-answer correctness, so its annotations cannot override the verifier.

For each correct-answer rollout, the judge parses the reasoning trajectory into $N \geq 1$ atomic steps $\{s_1, \ldots, s_N\}$. Each step receives a category label c_i from five mutually exclusive classes—*useful*, *mechanical*, *redundant*, *reversion*, and *error*. Only *useful* contributes positive reward, while the remaining four labels carry penalties of different severities. Label definitions, the judging prompt, and parsing details are provided in Appendix I.

From these annotations, we compute the useful-step ratio S_u:

$$S_u = \frac{1}{N} \sum_i \mathbf{1}[c_i = u], \quad (13)$$

where u denotes the *useful* label. S_u rewards trajectories with a high density of substantive reasoning steps.

3.3.3 Quality-Aware Reward Composition

The annotations from the previous section give us a positive useful-step signal and four kinds of low-value step. We compose them with bounded multiplication, which avoids the failure modes of additive composition (length canceling quality) and pure-product composition (a single category zeroing the reward). Concretely, for each undesirable category $k \in \mathcal{K} = \{\mathrm{mec}, \mathrm{red}, \mathrm{rev}, \mathrm{err}\}$, let r_k denote the fraction of steps assigned label k, and define

$$\Phi = \prod_{k \in \mathcal{K}} (1 - \lambda_k r_k), \quad (14)$$

$$\kappa = \frac{1}{1 + \alpha \ln(N + 1)}. \quad (15)$$

The overall process-shape reward is

$$Q_{\mathrm{process}} = S_u \cdot \Phi \cdot \kappa, \quad (16)$$

where S_u rewards useful reasoning density, Φ suppresses low-value or locally flawed steps, and κ softly encourages conciseness with a logarithmic shape that avoids over-penalizing already-short solutions while still discouraging unbounded length. We set all $\lambda_k < 1$, so each penalty factor lies in $(0, 1]$ and the product Φ is strictly positive. Since κ multiplies the positive term, conciseness alone cannot raise the reward; shortening helps only when S_u is preserved. Hyperparameter values are reported in Appendix I.

Optimizing QPR with GRPO. Stage 2 uses R_{QPR} as the scalar reward in the same GRPO backend. Because QPR keeps all training prompts in their original form, the model must internalize the process-shape preference rather than rely on explicit sub-question structure at inference time.

The complete two-stage training procedure is summarized in Algorithm 1 (Appendix A).

4 Step-Quality Evaluation Protocol

We propose a general **Step-Quality Evaluation Protocol** for diagnosing reasoning processes beyond final-answer accuracy. Accuracy alone cannot reveal whether a training method changes the reasoning trajectories that lead to an answer. The protocol preserves rule-based answer verification while adding post-hoc LLM-judged step annotations, and reports decomposed measures of useful reasoning, low-value steps, error localization, and token efficiency. These metrics are diagnostic-only and never feed back into training. The protocol shares QPR's five-class step taxonomy (§3.3) but uses independent evaluation judges and prompts (Appendix H). We instantiate it on standard scientific and mathematical reasoning benchmarks; a scaling validation across five model sizes confirms the protocol's discriminative power (Appendix E).

5 Experiments

We evaluate SCOPE-RL in a math/science-oriented RLVR setting with Qwen-family backbones, testing whether its stage-specific supervision—ASR before success and QPR after success—improves **learnability**, **trace quality**, and **efficiency**, and whether these gains hold across two training sources, four evaluation domains, and independent expert preferences.

Method	Accuracy (%) ↑					Reasoning Quality ↑		Avg. Tokens ↓
	$GPQA_{@1}$	$MATH500_{@1}$	$AIME24_{@8}$	$AIME25_{@8}$	Avg	Useful%	1st-Err-Pos	
Qwen3-8B-Instruct	46.97	69.92	26.25	18.75	40.47	54.76	0.665	5,727
Trained on DAPO-Math								
GRPO	48.48	79.70	51.67	43.33	55.80	56.79	0.606	6,936
ASR only	51.52	90.23	65.00	**56.67**	65.86	60.73	0.638	6,948
SCOPE-RL	**53.03**	**90.30**	**65.83**	56.25	**66.35**	**64.52**	**0.703**	**5,815**
Trained on Big-Math (data-source generalisation)								
GRPO	43.43	84.21	46.67	40.00	53.58	59.73	0.656	8,228
ASR only	53.03	**87.22**	59.17	50.00	62.36	61.92	0.666	6,245
SCOPE-RL	**55.05**	84.96	**64.58**	**54.58**	**64.79**	**66.47**	**0.738**	**5,994**

Table 1: Main results on accuracy, reasoning quality, and efficiency. All methods use GRPO; ASR denotes SCOPE-RL after Stage 1, before QPR. **Accuracy**: subset accuracy and arithmetic mean (Avg), with truncated responses counted incorrect. **Useful%**: fraction of LLM-judge-labeled useful steps. **1st-Err-Pos**: relative position of the first error step (↑ = later errors). **Avg. Tokens**: mean tokens per sample. Bold marks the best result within each training-data group.

Setup. We use Qwen3-8B-Instruct (Team, 2025) as the base model and outcome-only GRPO (Shao et al., 2024) as the primary baseline, trained for the same total optimizer budget as the two SCOPE-RL stages combined. SCOPE-RL is trained sequentially on 2,400 problems from each of DAPO-Math (Yu et al., 2025) and Big-Math (Albalak et al., 2025): Stage 1 applies ASR; its best checkpoint initializes Stage 2 QPR. We evaluate GPQA (Rein et al., 2023) and MATH500 (Lightman et al., 2024) with single-sample decoding, and report 8-sample average accuracy on AIME 2024 (Zhang and Math-AI, 2024) and AIME 2025 (Zhang and Math-AI, 2025). Full training, fairness, and evaluation-protocol details are in Appendix J.

5.1 Main Results and Reasoning Efficiency

Across both training sources, ASR raises final-answer accuracy over outcome-only GRPO; QPR then improves correct-trace quality and efficiency without sacrificing correctness (Table 1).

Accuracy. ASR raises average accuracy on both sources (DAPO-Math: 55.80→65.86%; Big-Math: 53.58→62.36%), with the largest gains on AIME24/25 where endpoint rewards are sparsest. QPR adds further headroom (66.35% / 64.79%) and improves process diagnostics: Useful% rises monotonically across the two stages, and 1st-Err-Pos shifts later. Pass@128 analysis (Appendix B) rules out a capacity ceiling from ASR scaffolding; QPR largely preserves this ceiling, with a minor coverage–quality trade on one AIME subset.

Efficiency. Total token reduction is 16.2% (DAPO-Math) and 27.1% (Big-Math) without sacrificing accuracy or useful-step density. The split is source-dependent: on DAPO-Math, Stage 1 preserves length (6,936→6,948) and Stage 2 carries the full reduction; on Big-Math, Stage 1 already cuts 24.1% (8,228→6,245) and Stage 2 refines a further 4.0%. QPR thus shifts from primary compressor to fine-grained refiner with Stage 1's residual redundancy.

5.2 Ablation Studies

Our ablations test whether the gains depend on prefix-decomposed verifiable rewards, correctness-gated process-shape rewards, and sequential training rather than auxiliary prompting, static routing, or simple brevity pressure.

ASR needs both scaffolded rewards and adaptive routing. Removing scaffolded decomposition (No Decomp.), applying it indiscriminately (All Decomp.), or replacing on-policy routing with offline difficulty labels each costs 7–9 pp of average accuracy (Table 2); both prefix-decomposed rewards and capability-tracking routing matter. A reward-signal ablation (Appendix L) isolates the prefix-consistency gate as the largest contributor.

Step-level decomposition, all three terms, and sequential staging are each load-bearing. Table 3 reports three coupled ablations on the shared ASR backbone. Panel (a): step-level decomposition is the load-bearing choice—*Holistic Judge*, which reuses the same correctness gate and LLM judge but collapses per-step labels into a single

Method	GPQA$_{@1}$	M500$_{@1}$	A24$_{@8}$	A25$_{@8}$	Avg
GRPO	48.48	79.70	51.67	43.33	55.80
No Decomp.	45.45	82.09	54.58	45.42	56.89
All Decomp.	46.97	77.61	59.58	50.00	58.54
Offline Route	42.42	84.33	57.50	51.25	58.88
ASR only	**51.52**	**90.23**	**65.00**	**56.67**	**65.86**

Table 2: ASR (Stage 1) ablation. *No Decomp.*: drop scaffolded rewards, keep dynamic sampler; *All Decomp.*: scaffold all problems, no routing; *Offline Route*: fixed difficulty labels instead of on-policy routing; *ASR only*: full Stage 1 model. M500: MATH500; A24/A25: AIME 2024/2025.

Method	Avg	Useful%	1st-Err	Tokens
(a) Simple alternatives				
Holistic Judge	63.98	64.14	0.707	7,277
Concise Prompt	62.53	58.69	0.662	**4,115**
Overlong Penalty	59.81	58.97	0.708	6,424
(b) Reward-component ablations				
No Positive	58.81	58.84	0.658	4,281
No Penalty	60.86	61.00	**0.751**	5,328
No Length	63.33	62.37	0.700	7,882
(c) Training-schedule ablation				
Mix-Stage	60.98	**65.86**	0.727	6,518
SCOPE-RL	**66.35**	64.52	0.703	5,815

Table 3: QPR (Stage 2) ablations on the shared ASR Stage 1 checkpoint; panel (c) *Mix-Stage* is the no-warm-up control with mixed ASR/QPR rewards. *Concise Prompt*: inference-time instruction on the same checkpoint, no QPR training (Appendix G).

trace score, nearly matches SCOPE-RL on Useful% (64.14%) yet trails 2.37 pp on Avg and uses 25% more tokens, while *Concise Prompt* sacrifices useful-step density (58.69%) and *Overlong Penalty* weakens accuracy without reliably controlling length. Panel (b): the three multiplicative terms in Q_{process} (Eq. (16)) target disjoint failure modes—removing the positive term collapses Avg (58.81%), removing the penalty leaves redundancy (Useful% 61.00%) despite the latest 1st-Err position (0.751), and removing the length factor preserves accuracy but inflates tokens to 7,882. Panel (c): *Mix-Stage* reaches only 60.98% Avg, confirming that prefix-decomposed verifiable rewards must precede process-shape refinement. Although individual ablations lead on isolated metrics (No Penalty on 1st-Err, Mix-Stage on Useful%), only SCOPE-RL jointly maximizes accuracy and overall trace quality.

5.3 Training Dynamics

Figure 3 corroborates the two-stage mechanism: ASR sustains a higher *effective gradient ratio*—the fraction of groups with non-degenerate advantages—by routing low-pass-rate problems to scaffolded prompts; response length grows under ASR then drops once QPR engages, while validation reward rises, indicating that QPR removes low-value reasoning rather than necessary steps.

5.4 Qualitative Analysis

Figure 4 decomposes correct traces into the five-category taxonomy. The progression matches Table 1: GRPO inflates length without raising useful-step density, ASR shifts steps toward useful reasoning, and QPR drops length while pushing Useful% to its peak.

A side-by-side trace comparison on the same Lagrange interpolation problem (Appendix R) makes the two-stage mechanism concrete: GRPO reaches an incorrect intermediate conclusion and recovers only via lengthy self-correction, ASR identifies the correct mathematical structure earlier though some redundant verification remains, and QPR distills the trace into a direct partition-of-unity derivation.

5.5 Robustness Across Optimizer Backend and Model Scale

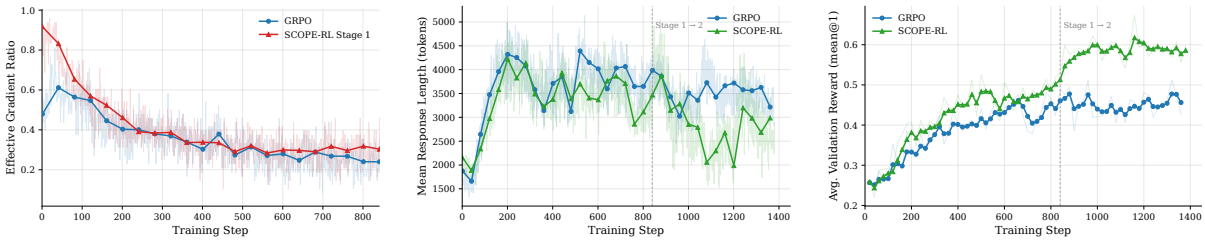
SCOPE-RL's gains hold under a GSPO backend (61.60→66.93%, 13.1% token reduction) and on Qwen3-0.6B-Instruct (26.06→32.06%, 22.9% token reduction), indicating that the before/after-success decomposition is not tied to a single optimizer or model scale. Full results are in Appendix Q.

5.6 Additional Process-Quality Validation

We further validate that QPR improves perceived trace quality rather than only automatic diagnostic scores. In a pairwise evaluation on 200 examples where both GRPO and SCOPE-RL are answer-correct, three domain experts prefer SCOPE-RL across all five dimensions: conciseness (72% vs. 28%), non-redundancy (69% vs. 31%), clarity (65.5% vs. 34.5%), logical coherence (59.5% vs. 40.5%), and overall quality (70.5% vs. 29.5%). Full results are reported in Appendix P.

6 Related Work

RLVR and hard-problem learning. Recent RLVR systems and GRPO-style optimizers (Guo



(a) Effective gradient ratio (b) Mean response length (c) Avg. validation reward

Figure 3: Training dynamics of GRPO vs. SCOPE-RL. (a) ASR maintains a higher effective gradient ratio via scaffolded routing. (b) Response length grows in Stage 1 then drops in Stage 2 as QPR encourages conciseness. (c) Validation reward improves across both stages. Dashed line marks the stage transition.

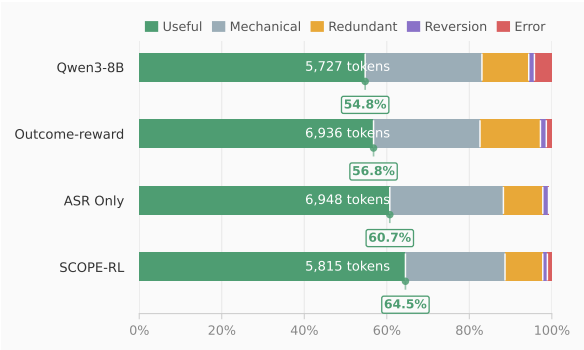


Figure 4: Step-quality composition of correct traces across training stages. Bars show proportions of useful, mechanical, redundant, reversion, and error steps, with mean token count and useful-step percentage.

et al., 2025; Shao et al., 2024; Yu et al., 2025; Zheng et al., 2025; Liu et al., 2025) improve LLM reasoning under verifiable final-answer rewards but face sparse signal on hard problems, motivating curricula (Parashar et al., 2025; Pang et al., 2025; Shi et al., 2025), difficulty filtering (Bae et al.), demonstrations (Yan et al., 2026), hint or prefix injection (Qu et al., 2026; Liao et al., 2026; Liu et al.; Zhang et al., 2025), reformulation or decomposition (Chih-Yao Chen et al., 2026; Simonds and Yoshiyama, 2025), and staged pipelines (Ma et al., 2025). These methods reshape inputs, rollouts, or GRPO updates (Zhang et al., 2025; Yan et al., 2026), but none install independently verifiable sub-answer rewards. ASR instead places *rule-based verifiers on each scaffolded sub-answer*, hidden from the policy. EvoCoT (Liu et al.) conditions rollouts on self-generated, verified CoT prefixes progressively shortened as a curriculum, not on independently verified sub-answer targets.

Process supervision and reasoning efficiency. Step-level supervision spans PRMs trained on step labels (Lightman et al., 2024; Wang et al., 2024), implicit process rewards from policy logits (Cui et al., 2025), refined aggregation or credit assignment of PRM rewards (Cheng et al., 2026; Rezaei et al., 2026), and analytical decomposition of the outcome objective (Yao et al., 2026); PRIME (Wang et al., 2026) supports verifier evaluation and selection rather than step labels, and efficiency methods reduce overthinking via length-aware or compression objectives (Hou et al., 2025; He et al., 2025; Rakotonirina et al., 2026). QPR differs in how it gates and composes process signal: conditional on a correct answer, a fixed (not learned) LLM judge labels each step's *functional role* over a closed taxonomy (useful, mechanical, redundant, reversion, error) and shapes the trajectory reward accordingly, with zero process reward on incorrect rollouts. QPR operates on the reward-signal axis while ASR exposes additional verifiable targets through answer-hidden scaffolds; both leave the policy-update rule unchanged.

7 Conclusion

We framed outcome-only RLVR as resting on a *sparse anchor*—reliable at the endpoint but underspecified on the reasoning path before and after success. **SCOPE-RL** densifies this anchor while retaining the GRPO update: **Adaptive Scaffolded RL** adds prefix-decomposed verifiable rewards on answer-hidden scaffolded sub-question chains before success, and **Quality-Aware Process RL** applies correctness-gated process-shape rewards to refine verified-correct trajectories after success. Together with the Step-Quality Evaluation Protocol, experiments on Qwen3-8B-Instruct show gains in accuracy, token efficiency, and process quality across DAPO-Math and Big-Math, with robustness checks under GSPO and on Qwen3-0.6B-Instruct suggesting that reward-signal densification is complementary to policy-update-level RLVR advances.

Limitations

Our experiments are limited to Qwen-family backbones and math/science-oriented RLVR settings. The main results use Qwen3-8B-Instruct, with additional validation on Qwen3-0.6B-Instruct and a GSPO backend, but do not cover other model families, larger-scale models, or broader verifiable-reward domains.

We do not exhaustively compare against all hint-based exploration or length-aware reasoning-compression methods. This is a deliberate scope choice: our experiments isolate whether verifier-compatible reward densification improves RLVR when the policy optimizer and training interface are fixed. Combining SCOPE-RL with these orthogonal techniques is a promising direction for future work.

Finally, our hyperparameter analysis is limited to a sensitivity sweep for the routing threshold τ (Appendix K); broader tuning of β, α, and $\{\lambda_k\}$ remains future work.

References

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and 1 others. 2025. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *arXiv preprint arXiv:2502.17387*.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning, 2025. *URL https://arxiv.org/abs/2504.03380*.

Jie Cheng, Gang Xiong, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Yisheng Lv, and Fei-Yue Wang. 2026. Stop summation: Min-form credit assignment is all process reward model needs for reasoning. *Advances in Neural Information Processing Systems*, 38:131646–131671.

Justin Chih-Yao Chen, Archiki Prasad, Zaid Khan, Joykirat Singh, Runchu Tian, Elias Stengel-Eskin, and Mohit Bansal. 2026. Cog-drift: Exploration on adaptively reformulated instances enables learning from hard reasoning problems. *arXiv e-prints*, pages arXiv–2604.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xingyang He, Xiao Ling, and Jie Liu. 2025. Smartthinker: Learning to compress and preserve reasoning by step-level length control. *arXiv preprint arXiv:2507.04348*.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, and Jiang Bian. 2026. Self-hinting language models enhance reinforcement learning. *arXiv preprint arXiv:2602.03143*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.

Huanyu Liu, Jia Li, Chang Yu, Taozhi Chen, Yihong Dong, Lecheng Wang, Yongding Tao, Hu XiaoLong, and Ge Li. Evocot: Overcoming the exploration bottleneck in reinforcement learning for llms.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, Yanhao Li, and 1 others. 2025. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. *arXiv preprint arXiv:2506.07527*.

Bo Pang, Deqian Kong, Silvio Savarese, Caiming Xiong, and Yingbo Zhou. 2025. Reasoning curriculum: Bootstrapping broad llm reasoning from math. *arXiv preprint arXiv:2510.26143*.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and 1 others. 2025. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *arXiv preprint arXiv:2506.06632*.

Sundar Pichai, Demis Hassabis, and Koray Kavukcuoglu. 2025. A new era of intelligence with gemini 3. *Google. URL: https://blog. google/products-and-platforms/products/gemini/gemini*, 3.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. 2026. Pope:

Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Nathanaël Carraz Rakotonirina, Ren Pang, Neha Anna John, Michael Bohlke-Schneider, and Momchil Hardalov. 2026. Correct, concise and complete: Multi-stage training for adaptive reasoning. *arXiv preprint arXiv:2601.02972*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Mohammad Rezaei, Jens Lehmann, and Sahar Vahdati. 2026. Llm reasoning with process rewards for outcome-guided steps. *arXiv preprint arXiv:2604.02341*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. 2025. Efficient reinforcement finetuning via adaptive curriculum learning. *arXiv preprint arXiv:2504.05520*.

Toby Simonds and Akira Yoshiyama. 2025. Ladder: Self-improving llms through recursive problem decomposition. *arXiv preprint arXiv:2503.00735*.

Qwen Team. 2025. Qwen3 technical report, 2025. *URL https://arxiv. org/abs/2505.09388*, 3(4.2):1.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. 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)*, pages 9426–9439.

Xiangfeng Wang, Hangyu Guo, Yanlin Lai, Mitt Huang, Liang Zhao, Chengyuan Yao, Yinmin Zhang, Qi Han, Xiaoxiao Ren, Chun Yuan, and 1 others. 2026. Prime: A process-outcome alignment benchmark for verifiable reasoning in mathematics and engineering. *arXiv preprint arXiv:2602.11570*.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2026. Learning to reason under off-policy guidance. *Advances in Neural Information Processing Systems*, 38:117157–117186.

Jiarui Yao, Ruida Wang, and Tong Zhang. 2026. Prl: Process reward learning improves llms' reasoning ability and broadens the reasoning boundary. *arXiv preprint arXiv:2601.10201*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale, 2025. *URL https://arxiv. org/abs/2503.14476*, 1:2.

Kaiyi Zhang, Ang Lv, Jinpeng Li, Yongbo Wang, Feng Wang, Haoyuan Hu, and Rui Yan. 2025. Stephint: Multi-level stepwise hints enhance reinforcement learning to reason. *arXiv preprint arXiv:2507.02841*.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2024.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Shu Zhou, Rui Ling, Junan Chen, Xin Wang, Tao Fan, and Hao Wang. 2026. When more thinking hurts: Overthinking in llm test-time compute scaling. *arXiv preprint arXiv:2604.10739*.

A Training Algorithm

Algorithm 1 presents the complete pseudocode for the SCOPE-RL two-stage training procedure described in Section 3.

Algorithm 1 SCOPE-RL: Two-Stage Training Framework

Input: Base model π_θ, training data $\mathcal{D}$, scaffold function $\mathcal{S}(\cdot)$, threshold τ, group size G, LLM Judge $\mathcal{J}$

— Stage 1: Adaptive Scaffolded RL —
for each training step **do**
  Sample prompt batch $\{q_1, \ldots, q_B\}$ from $\mathcal{D}$
  for each prompt q_j **do**
    Generate G rollouts from π_θ on q_j; compute $\bar{r}(q_j)$
    if $\bar{r}(q_j) < \tau$ **then**
      Replace q_j with scaffolded prompt $\mathcal{S}(q_j)$; re-generate G rollouts
      Compute R_{ASR} via Eq. (10)
    else
      Retain original rollouts with outcome reward r_{out}
    end if
  end for
  Merge all groups into unified batch; update π_θ with the chosen RLVR optimizer
end for
— Stage 2: Quality-Aware Process RL —
Initialize π_θ from best Stage 1 checkpoint
for each training step **do**
  Sample prompt batch; generate G rollouts from π_θ on original prompts
  for each rollout y **do**
    if $\hat{a}(y) = a$ **then**
      Invoke $\mathcal{J}$ to annotate steps with category labels
      Compute $Q_{\text{process}}(y)$ via Eq. (16); set $R_{\text{QPR}} = Q_{\text{process}}(y)$
    else
      Set $R_{\text{QPR}} = 0$
    end if
  end for
  Update π_θ with the chosen RLVR optimizer using R_{QPR} as reward
end for

B Pass@k Analysis

Since ASR employs an offline teacher model to generate scaffolded decompositions for difficult problems, a natural concern is whether this auxiliary supervision constrains the model's exploration capacity. We address this with a pass@k analysis on AIME 2024 (Zhang and Math-AI, 2024), AIME 2025-I, and AIME 2025-II (Zhang and Math-AI, 2025); we read pass@k=128 as an upper-bound proxy for reasoning capacity, since at this budget any solution the model can in principle find is likely to surface in at least one of 128 samples.

At k=128, ASR matches or exceeds GRPO on every subset (AIME24: 90.00% vs. 90.00%; AIME25-I: 93.33% vs. 86.67%; AIME25-II: 93.33% vs. 86.67%) and exceeds the base model by 17–27 pp. Scaffolded training therefore does not cap reasoning potential—if anything, it raises the ceiling on the harder AIME 2025 subsets.

SCOPE-RL retains the GRPO pass@128 capacity on AIME 2024 (90.00%) and AIME 2025-II (86.67%), but drops on AIME 2025-I (86.67%→80.00%). This is consistent with QPR's design as a quality refiner rather than a capacity expander: by concentrating probability mass on cleaner, shorter derivations, the policy trades a small amount of solution diversity for higher useful-step density and lower token cost (Tab. 1). Capacity itself is established at ASR; SCOPE-RL then operates on top of that capacity. For applications that prioritize raw coverage at large sampling budgets, the ASR checkpoint is the appropriate operating point; SCOPE-RL is shipped because the deployment regime targets accuracy and efficiency at small k rather than the high-k frontier.

C Motivation Probing Experiment Details

To verify that scaffold-prefix progress reflects genuine path-level learning signal rather than merely correlating with intrinsic problem easiness, we design the following probing experiment.

Data construction. We sample 755 problems from the training set whose scaffold decompositions contain exactly 4 sub-questions. Fixing the number of sub-questions eliminates chain-length variation as a confound in the binning analysis.

Evaluation procedure. We evaluate the base model (Qwen3-8B-Instruct (Team, 2025)) on these 755 problems under two conditions: (1) *Scaffold prompt*: the model receives the full sub-question chain together with the original main problem and is asked to answer all sub-questions and the main problem sequentially; (2) *Original prompt*: the model receives only the original, undecomposed

Dataset	Method	k=1	k=8	k=16	k=32	k=64	k=128
AIME 2024	Qwen3-8B-Instruct	26.67	46.67	60.00	66.67	66.67	66.67
	GRPO	60.00	80.00	83.33	86.67	90.00	90.00
	ASR only	66.67	83.33	83.33	83.33	86.67	90.00
	SCOPE-RL	63.33	83.33	83.33	83.33	86.67	90.00
AIME 2025-I	Qwen3-8B-Instruct	26.67	46.67	53.33	53.33	66.67	66.67
	GRPO	40.00	66.67	66.67	66.67	86.67	86.67
	ASR only	53.33	66.67	73.33	80.00	93.33	93.33
	SCOPE-RL	53.33	73.33	80.00	80.00	80.00	80.00
AIME 2025-II	Qwen3-8B-Instruct	20.00	26.67	40.00	53.33	60.00	73.33
	GRPO	46.67	86.67	86.67	86.67	86.67	86.67
	ASR only	60.00	86.67	86.67	86.67	93.33	93.33
	SCOPE-RL	53.33	86.67	86.67	86.67	86.67	86.67

Table 4: Pass@k accuracy (%) on AIME subsets. Reasoning capacity is read at k=128: ASR matches or exceeds GRPO on all three subsets, showing scaffolded training does not cap potential. SCOPE-RL preserves the GRPO capacity on two of three subsets and trades coverage for trace quality on AIME 2025-I, consistent with QPR's role as a quality refiner rather than a capacity expander.

problem.

Binning and metric computation. From the scaffold-prompt rollouts, we determine the longest correct prefix for each sample—i.e., the number of consecutive sub-questions answered correctly before the first failure (ranging from 1 to 4). Samples are grouped into four bins by this prefix length. For each bin, we report two main-answer accuracy values: (a) accuracy under the scaffold prompt (green line in Figure 1a), and (b) accuracy when the same set of problems is evaluated under the original prompt (red line in Figure 1a). The comparison between the two lines on the same bin isolates the effect of scaffold-provided structure from problem-level difficulty.

D Reward Properties: Bounded Scale and Effective Gradient Support

We analyze two properties of the stage-specific rewards. **Bounded scale** (§D.1) keeps both rewards on the same $[0, 1]$ range as r_{out}, so the GRPO clip range and KL coefficient need not be re-tuned per stage. **Effective gradient support** (§D.2) shows that bounded scale alone does not ensure learning: on the prompt subsets where outcome-only training is degenerate ($\sigma^2_{r_{\text{out}}} = 0$), ASR and QPR can each provide *strictly positive* reward variance under the heterogeneity conditions stated below.

D.1 Bounded Scale

For ASR, with $m \geq 1$, $\beta \in [0, 1]$, $\Pi_i \in \{0, 1\}$, and $\mathbf{1}[\hat{a}_{\text{main}} = a] \leq 1$, Eq. (10) gives

$$0 \leq R_{\text{ASR}} \leq \beta + (1 - \beta) = 1. \tag{17}$$

For QPR, with $S_u, r_k \in [0, 1]$, $\lambda_k \in [0, 1)$, and $\alpha \geq 0$, each penalty factor satisfies $(1 - \lambda_k r_k) \in (0, 1]$, so $\Phi \in (0, 1]$; also $\kappa = 1/(1 + \alpha \ln(N + 1)) \in (0, 1]$ for $N \geq 1$. Eq. (16) gives

$$0 \leq Q_{\text{process}} \leq S_u \leq 1. \tag{18}$$

Since $R_{\text{QPR}} = Q_{\text{process}}$ for verifier-correct rollouts and $R_{\text{QPR}} = 0$ otherwise, $0 \leq R_{\text{QPR}} \leq 1$ follows.

D.2 Effective Gradient Support

A bounded reward stays on a learnable scale, but GRPO requires within-group reward variation to produce non-zero advantages. Concretely, the std-zero convention (§2) makes the per-rollout advantage piecewise:

$$A_g(q) = \begin{cases} \dfrac{R(q, y_g) - \bar{R}(q)}{\sigma_R(q)}, & \sigma^2_R(q) > 0, \\ 0, & \sigma^2_R(q) = 0, \end{cases} \tag{19}$$

where $\bar{R}(q) = \text{mean}_j R(q, y_j)$ and $\sigma_R(q) = \sqrt{\sigma^2_R(q)}$. Throughout this appendix, Var_g denotes the empirical population variance over the G sampled rollouts. The policy receives reward-driven gradient on q if and only if $\sigma^2_R(q) > 0$. The effective gradient ratio $\eta_R = \mathbb{P}_q[\sigma^2_R(q) > 0]$ (Eq. (4))

measures the fraction of prompts that pass this threshold.

For the outcome-only reward $r_{\text{out}} \in \{0, 1\}$, let $p(q) = \frac{1}{G}\sum_g r_{\text{out}}(y_g, a)$ denote the empirical pass rate in the sampled group. Then

$$\sigma^2_{r_{\text{out}}}(q) = p(q)\big(1 - p(q)\big), \tag{20}$$

which vanishes exactly when $p(q) \in \{0, 1\}$, i.e. when the group is uniformly correct or uniformly wrong. The fraction of prompts contributing reward-driven gradient under outcome-only training is therefore $\eta_{r_{\text{out}}} = \mathbb{P}_q[0 < p(q) < 1]$, the prompts where the current policy is neither saturated nor fully blocked.

ASR recovers signal via prefix disagreement. Let $L(y_g) = \sum_{i=1}^{m} \Pi_i(y_g)$ denote the length of the correct scaffold prefix. Since the prefix indicators are nested (Π_i requires consecutive correctness on sub-questions 1 through i, so $\Pi_1 \geq \Pi_2 \geq \cdots \geq \Pi_m$), $\Pi_m = \mathbf{1}[L(y_g) = m]$, and Eq. (10) rewrites as

$$\begin{aligned} R_{\text{ASR}}(y_g) = \tfrac{\beta}{m}\, L(y_g) \\ + (1-\beta)\, \mathbf{1}[L(y_g)=m]\, \mathbf{1}[\hat{a}_{\text{main}}=a]. \end{aligned} \tag{21}$$

If $L(y_g)$ differs across rollouts in a routed group and $\beta > 0$, then R_{ASR} takes distinct values across rollouts, so $\sigma^2_{R_{\text{ASR}}}(q) > 0$. This holds in particular on routed groups with $\sigma^2_{r_{\text{out}}}(q) = 0$: ASR recovers gradient signal on this degenerate subset whenever scaffold-prefix lengths disagree across rollouts, without requiring any assumption on the main-answer distribution under the scaffolded rollouts.

Remark on Π_m as a verification gate. The factor Π_m in the second term means that when $\Pi_m = 0$ on all rollouts, the main-answer match contributes nothing even if some $\hat{a}_{\text{main}}(y_g) = a$ holds by coincidence. By design, this prevents R_{ASR} from rewarding lucky main-answer guesses without prerequisite progress, in line with ASR's design intent of crediting verifiable progress rather than coincidence; on such "lucky" rollouts r_{out} would have variance, but the second term of R_{ASR} intentionally contributes none. The empirical gap between GRPO and SCOPE-RL Stage 1 in Figure 3a indicates that ASR's prefix signal outweighs the lucky-guess variance suppressed by Π_m in practice.

QPR recovers signal on outcome-uniform-correct groups. Let $p_c(q) = \frac{1}{G}\sum_g \mathbf{1}[\hat{a}(y_g) = a]$ denote the correct-rollout fraction in the sampled group (equivalently, $p_c \equiv p$). If $p_c(q) = 0$, then R_{QPR} is identically zero and $\sigma^2_{R_{\text{QPR}}}(q) = 0$. For $p_c(q) > 0$, let μ_c, ν_c denote the mean and variance of $Q_{\text{process}}(y)$ conditioned on $\hat{a}(y) = a$. The QPR gate (Eq. (12)) makes R_{QPR} a mixture of $\{0\}$ and Q_{process}, and a law-of-total-variance decomposition gives

$$\sigma^2_{R_{\text{QPR}}}(q) = p_c\, \nu_c + p_c(1 - p_c)\, \mu_c^2, \tag{22}$$

where p_c, μ_c, ν_c are evaluated at q and $\mu_c \geq 0$ since $Q_{\text{process}} \in [0, 1]$. Eq. (22) is strictly positive iff $p_c(q) > 0$ and either (i) $\nu_c(q) > 0$ (heterogeneity of Q_{process} among correct rollouts), or (ii) $0 < p_c(q) < 1$ and $\mu_c(q) > 0$ (mixed group with a non-zero typical correct-rollout reward). The first term captures intra-correct variance in process shape; the second is the binary-pass-rate variance scaled by μ_c^2. On groups with mixed correct and incorrect rollouts ($0 < p_c < 1$), $\sigma^2_{r_{\text{out}}}$ is already positive; QPR additionally provides finer-grained variation under condition (i) or (ii), and loses signal only in the boundary case where the judge returns identically zero on all correct rollouts ($\mu_c = \nu_c = 0$). The genuine recovery happens on *outcome-uniform-correct* groups ($p_c = 1$, where $\sigma^2_{r_{\text{out}}} = 0$):

$$\sigma^2_{R_{\text{QPR}}}(q) \;=\; \nu_c(q), \tag{23}$$

which is positive whenever correct rollouts differ in process shape; if the judge collapses to a constant value on all correct rollouts of such a group, $\nu_c = 0$ and QPR provides no signal on that prompt.

The two mechanisms target the two regimes that outcome-only training cannot resolve: ASR provides signal on routed groups via prefix-length disagreement, recovering gradient on the routed subset with $\sigma^2_{r_{\text{out}}} = 0$; QPR provides signal on outcome-uniform-correct groups via heterogeneous trajectory quality, recovering gradient on $\{p_c = 1\}$. Both recovered subsets are degenerate under r_{out}.

E Protocol Validation: Scaling Discrimination

Table 5 validates the Step-Quality Evaluation Protocol by applying it to the Qwen3.5 model family (Qwen Team, 2026) across five scales. Metrics improve consistently with model capacity, confirming the protocol's discriminative power.

Metric	Qwen3.5-0.8B	Qwen3.5-2B	Qwen3.5-4B	Qwen3.5-9B	Qwen3.5-27B
Accuracy (%) ↑					
Overall	8.7	27.0	64.4	72.5	84.2
$\text{GPQA}_{@1}$	18.7	42.9	73.2	77.3	80.3
$\text{MATH500}_{@1}$	18.1	45.9	88.7	92.5	95.5
$\text{AIME 2024}_{@8}$	1.3	17.5	62.1	71.7	86.3
$\text{AIME 2025}_{@8}$	2.9	12.9	46.3	58.8	79.2
Process Quality					
Useful-step ratio (%) ↑	50.1	61.6	64.2	63.6	63.9
Error-step ratio (%) ↓	16.5	3.5	0.7	0.6	0.3
Redundant-step ratio (%) ↓	12.3	7.6	6.6	6.9	7.6
Reversion-step ratio (%) ↓	2.1	1.8	0.9	0.6	1.0
First-error rel. position ↑	0.353	0.529	0.734	0.664	0.769
Efficiency					
Truncation rate (%) ↓	59.1	36.7	5.1	4.1	4.9
Avg. steps	27.2	28.1	29.8	29.7	31.7
Avg. tokens	10,606	8,998	5,747	5,238	7,162

Table 5: Validation of the Step-Quality Evaluation Protocol across model scales. Metrics are computed on the same evaluation set ($\text{GPQA}_{@1}$, $\text{MATH500}_{@1}$, $\text{AIME 2024}_{@8}$/$\text{2025}_{@8}$). Monotonic improvements in accuracy and error-step suppression confirm the protocol's discriminative power.

F Scaffold Decomposition Prompt

The system prompt used in the automated decomposition pipeline (Section 3.2; statistics in Appendix M) is shown in Figure 5.

G Concise Reasoning Prompt

To establish a non-RL baseline for reasoning compression, we prepend the following concise-reasoning instruction to the system prompt when evaluating the base model without any RL training:

Prompt Template for Concise Reasoning

System: Answer as concisely as possible. Avoid unnecessary redundancy, repetition, and verbose explanations. Focus on key reasoning steps and provide the final answer directly.

User: [Question]

Figure 6: Concise reasoning prompt prepended at inference time for the non-RL compression baseline.

H Step-Quality Evaluation Protocol Details

Evaluation set instantiation. As a concrete instantiation of the protocol, we select four established datasets spanning different reasoning domains and difficulty levels: GPQA (Rein et al., 2023) for graduate-level scientific reasoning, MATH500 (Lightman et al., 2024) for competition mathematics, and AIME 2024 (Zhang and Math-AI, 2024) and AIME 2025 (Zhang and Math-AI, 2025) for high-difficulty olympiad-style problems. The protocol itself is not tied to these specific datasets and can be applied to any STEM evaluation set with verifiable answers. GPQA and MATH500 are evaluated with single-sample generation. To reduce variance on the smaller and harder AIME subsets, each AIME problem is sampled 8 times from the evaluated model and scores are averaged across rollouts.

Step-level judging protocol. For each generated response, final-answer correctness is computed independently using a rule-based answer checker. Process evaluation is applied to the reasoning trace rather than to the final answer, and the LLM Judge is never used to decide whether the final answer is correct. The evaluation judge (Gemini-3-flash-preview (Pichai et al., 2025)) is also a different model from the one used to produce QPR training rewards (GPT-4.1-mini), so the protocol's diagnostic measurements are not generated by the same judge that shaped the policy. The judge first parses the reasoning trace into N atomic steps $\{s_1, \ldots, s_N\}$ without summarizing or rewriting the model's solution. It then assigns each step a category label c_i from the same five-way taxonomy used in QPR: *useful*, *mechanical*, *redundant*, *reversion*, and *error*.

The five labels separate distinct process behaviors. *Useful* steps make genuine cognitive progress toward the solution. *Mechanical* steps are correct

Prompt Template for Scaffold Decomposition

System: You are a mathematics problem decomposition engine. You will be given a single original math problem and its final correct answer. Your task is to decompose the original problem into several sub-problems.

Mandatory Rules:
1. Ground-truth validity constraint. The ground truth of the main problem MUST be a single value, interval, finite set, or algebraic expression. If not, return an empty JSON object {}.
2. Sub-problem ground truth constraint. Each sub-problem MUST have a ground truth that is also a single value, interval, finite set, or algebraic expression. Forbidden outputs: "true/false", "holds if and only if ...", any form of proof or explanation.
3. Meaningful decomposition. Each sub-problem must represent a distinct mathematical concept or technique, require mathematical reasoning (not just arithmetic), and add genuine pedagogical value. Forbidden: pure calculation steps, trivial simplifications.
4. Progressive problem chain. Each sub-problem should use results from previous sub-problems. The last sub-problem must be genuinely different from the main problem. The main problem should require combining insights from multiple sub-problems.
5. Problem structure format.
`[Given conditions and context]`
`Sub-problem 1: [First conceptual step]`
`Sub-problem 2: Using the result of Sub-problem 1, [Second step]`
`...`
`Main Problem: [Final question using results from multiple sub-problems]`
6. Answer format specification. When an answer could be expressed in multiple equivalent ways, append a format requirement: "Express your answer as [specific format]." Prefer asking for counts when possible.
7. Ground truth LaTeX format constraint. All answers must be in valid LaTeX math format (without surrounding $).

Output JSON structure:
`{"data_source": "step_math", "prompt": [{"role": "system", "content": "Solve each sub-problem step by step. Put ONLY the final answer in \boxed{} labeled [SUB-X ANSWER] / [MAIN ANSWER]."}, {"role": "user", "content": "[problem text]"}], "reward_model": {"ground_truth": {"sub1": "...", ..., "main": "..."}, "style": "rule"}}`

Quality check: (1) Does each sub-problem teach a distinct concept? (2) Is the last sub-problem different from the main problem? (3) Does the main problem require synthesizing multiple results? (4) Would a student learn mathematical thinking? If any answer is "no", return {}.

User: [Question] Answer: [Ground-truth answer]

Figure 5: System prompt for the scaffold decomposition pipeline. The decomposer generates answer-hidden sub-question chains from a given math problem and its ground-truth answer.

but mostly executional. *Redundant* steps repeat information or re-derive already established facts. *Reversion* steps occur when the model unnecessarily second-guesses or backtracks on previously correct reasoning, causing the solution path to regress. *Error* steps contain locally invalid reasoning, even when the final answer is eventually correct. This taxonomy allows the protocol to distinguish shorter high-quality reasoning from outputs that are merely short, lucky, or under-explained.

Diagnostic metrics. The protocol reports decomposed metrics rather than relying only on the aggregate reward used for training. Let $\mathcal{P}$ be the set of evaluated instances, $\hat{a}_p$ the extracted final answer, and a_p the ground-truth answer. We report final-answer accuracy as

$$F = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathbf{1}[\hat{a}_p = a_p], \tag{24}$$

with truncated responses counted as incorrect in the overall accuracy setting.

For process quality, we report the useful-step ratio

$$S_u = \frac{1}{N} \sum_i \mathbf{1}[c_i = u], \tag{25}$$

and the negative-step ratios

$$r_k = \frac{1}{N} \sum_i \mathbf{1}[c_i = k], \quad k \in \mathcal{K}. \tag{26}$$

Here, u denotes the *useful* label, and $\mathcal{K} = \{\mathrm{mec}, \mathrm{red}, \mathrm{rev}, \mathrm{err}\}$ denotes mechanical, redundant, reversion, and error steps. These metrics directly quantify whether a model uses more substantive reasoning steps and fewer low-value or harmful steps after training.

We additionally measure where the reasoning chain first breaks down. For instances containing at least one error step, we compute

$$S_{\text{first-err}} = \frac{1}{|\mathcal{P}_{\mathrm{err}}|} \sum_{p \in \mathcal{P}_{\mathrm{err}}} \frac{i_{\mathrm{err}}^{(p)}}{N_p}, \tag{27}$$

where $i_{\mathrm{err}}^{(p)}$ is the first error-step index and N_p is the number of steps in instance p. Higher values indicate that errors occur later in the reasoning chain, reflecting longer locally coherent derivations.

For efficiency, we report average step count and average completion tokens. These efficiency metrics are interpreted jointly with accuracy and useful-step density: a reduction in length is considered beneficial only when final accuracy and useful-step density are preserved or improved.

Evaluation set statistics. Table 6 summarizes the composition of the evaluation set used in this work. In total, it contains 392 unique problems and produces 812 evaluation instances per model.

Dataset	Problems	Samples	Total
GPQA	198	1	198
MATH500	134	1	134
AIME 2024	30	8	240
AIME 2025	30	8	240
Total	392	—	812

Table 6: Composition of the evaluation set. AIME problems are sampled multiple times to reduce evaluation variance on high-difficulty olympiad problems.

Implementation pipeline. The protocol is implemented as a three-stage pipeline. We deliberately use a different model for evaluation than for QPR training: all evaluation-time judging (step parsing, step scoring, and the IAA study in Appendix O) uses **Gemini-3-flash-preview**, whereas QPR training rewards are produced by GPT-4.1-mini (Appendix I). This decoupling ensures that the diagnostic metrics reported in the main results are not produced by the same model that shaped the policy during training. Given a set of problems with ground-truth answers and a target model to evaluate, the pipeline proceeds as follows:

1. **Model Rollout.** Each problem is sent to the target model. The model generates a complete reasoning trace followed by a final answer. Responses that exceed the maximum token budget are marked as truncated and counted as incorrect in overall accuracy, but excluded from process-quality evaluation (since their reasoning chains are incomplete).

2. **Step Parsing.** An LLM Judge (Gemini-3-flash-preview) segments each non-truncated reasoning trace into atomic steps. The judge is instructed to faithfully reconstruct the original reasoning process without summarizing, merging, or optimizing — preserving redundancy, hesitation, and self-correction as separate steps.

3. **Step Scoring.** A second LLM Judge call evaluates each parsed step by assigning a mutually exclusive category label applied in strict priority order (reversion > error > redundant > mechanical > useful). The judge also renders a binary final-answer correctness decision by comparing the model's answer against the ground truth.

All metrics reported in the main paper are computed from the Step Scoring outputs: category ratios are step-count fractions, and first-error position is computed per-problem then averaged.

H.1 Step Parsing Prompt

The following prompt instructs the LLM Judge to segment a model's reasoning trace into atomic steps without any modification or judgment.

Prompt Template for Step Parsing

System: You are a mathematical reasoning analyst. Your goal is to faithfully reconstruct the ORIGINAL reasoning process step by step, NOT to summarize or clean it.

Critical Rules:
- Preserve redundancy: if the model repeats the same idea, create multiple steps.
- Preserve verification: include sanity checks, examples, and side explorations.
- Preserve hesitation and re-explanations.
- DO NOT merge similar steps.
- DO NOT optimize or simplify the reasoning.
- DO NOT judge whether any step is correct, useful, or redundant — just segment faithfully.

Each step should correspond to ONE actual reasoning action in the original text.

Output format:
```
{"steps": [{"step_id": 0, "content": "what
the model is doing in this exact moment"},
...]}
```

User: Problem: [Question]
Model output: [Full model reasoning trace]

Figure 7: Prompt template for the step parsing stage. The judge segments each reasoning trace into atomic steps without summarization or modification.

H.2 Step Scoring Prompt

The following prompt instructs the LLM Judge to assign category labels to each parsed step, and to determine final-answer correctness.

Prompt Template for Step Scoring

System: You are an expert evaluator of mathematical reasoning quality. You will be given: (1) a math problem, (2) the ground truth answer, (3) the model's full reasoning output and its structured reasoning steps. Evaluate on three dimensions:

Dimension 1 — Per-step Category (mutually exclusive, applied in priority order):

1. **reversion** — The model was on a correct reasoning path but unnecessarily second-guesses, backtracks, or revises a previously correct step, causing the reasoning to regress (look for "wait", "actually", "let me reconsider", "I think I was wrong" when the original reasoning was in fact correct).
2. **error** — The step contains a mathematical or logical mistake NOT accompanied by explicit self-correction (silently wrong).
3. **redundant** — Mathematically correct but merely restates something already established, adding no new information.
4. **mechanical** — Correct and non-redundant, but purely executional with no new cognitive contribution (e.g., expanding algebra, substituting values, arithmetic).
5. **useful** — Everything else: genuine cognitive contribution that advances the solution (choosing strategy, identifying insight, applying non-trivial theorem, case-splitting).

Dimension 2 — Answer Correctness:
Compare model's final answer with ground truth. Set `is_correct` to true/false.

Output format:
```
{"is_correct": true, "per_step_scores":
[{"step_id": 0, "category": "useful"},
...]}
```

User: Problem: [Question]
Ground truth answer: [Answer]
Model's full output: [Reasoning trace]
Model's structured reasoning steps: [Parsed steps from Stage 2]

Figure 8: Prompt template for the step scoring stage. The judge assigns a category label to each parsed step and determines final-answer correctness.

I QPR Reward Details

Step labels. QPR uses five mutually exclusive labels to characterize the role of each reasoning step after final-answer correctness has been verified. *Useful* steps make genuine cognitive progress toward the solution. *Mechanical* steps are correct but primarily executional computations. *Redundant* steps restate or re-derive information that has already been established. *Reversion* steps occur when the model unnecessarily second-guesses or backtracks on previously correct reasoning, causing the solution path to regress. *Error* steps contain locally invalid reasoning even when the final answer is eventually correct.

Judge annotation. For each correct-answer rollout, the LLM Judge parses the trace into atomic reasoning steps without rewriting the solution and assigns one category label to each step. These annotations are used only to construct the process-quality reward; final-answer correctness remains determined by the rule-based verifier.

Judge prompt template. The LLM Judge (GPT-4.1-mini) receives each problem, its ground-truth answer, and the model's complete reasoning trace; it returns a structured JSON with per-step category labels.

Reward hyperparameters. In our experiments, we set $\alpha = 0.5$. The penalty weights are $\lambda_{\text{mec}} = 0.05$, $\lambda_{\text{red}} = 0.20$, $\lambda_{\text{rev}} = 0.25$, and $\lambda_{\text{err}} = 0.40$, assigning the strongest penalty to locally invalid reasoning and smaller penalties to low-value but correct computation.

J Training Implementation Details

All experiments are conducted on a single node equipped with 8 NVIDIA H800 GPUs, using the verl 0.7.0 training framework with vLLM 0.9.2 as the rollout engine. The base model is **Qwen3-8B-Instruct** (Team, 2025). Both stages use GRPO (Shao et al., 2024) with n=8 rollout samples per prompt, a prompt batch size of 48, and a maximum prompt length of 1,024 tokens. KL penalty is disabled in both stages, including both reward-side KL and loss-side KL loss, and the entropy coefficient is set to 0.

Stage 1 — Adaptive Scaffolded RL. Each batch begins with rollouts on the original problems under the outcome reward. Any group whose mean outcome reward falls below the threshold τ is re-rolled out on the cached scaffolded prompt and scored with the prefix-consistent scaffold reward defined in Section 3.2. Within a routed group, scaffolded rollouts *replace* rather than augment the original rollouts, so each parameter update aggregates the same number of loss-contributing trajectories (G=8 per prompt) as outcome-only GRPO; this preserves compute and gradient-batch parity with the baseline. Scaffold chains are generated offline with a teacher decomposer and cached on disk; no decomposition model is invoked during RL. The retained original-problem trajectories and

the scaffolded trajectories are combined into a single batch for the GRPO (Shao et al., 2024) parameter update. No separate value network is used; the mixed reward serves directly as the scalar signal for advantage estimation. The learning rate is set to 1×10^{-6} with a 60-step linear warmup and no decay, and the maximum response length is 8,192 tokens.

Stage 2 — Quality-Aware Process RL. QPR is initialized from the best ASR checkpoint and trained with a lower learning rate of 5×10^{-7} under a cosine decay schedule. The maximum response length is extended to 16,384 tokens to accommodate longer reasoning traces during the early phase of QPR training. Each rollout is scored by the correctness-gated process reward defined in Eq. (16): incorrect final answers receive zero reward, while correct-answer rollouts are annotated by an LLM Judge implemented using GPT-4.1-mini. The judge decomposes each reasoning trace into atomic steps and assigns one of five mutually exclusive labels (*useful, mechanical, redundant, reversion, error*).

K Scaffold Routing Threshold Sensitivity

Table 7 reports the effect of the scaffold routing threshold τ on Stage 1 (ASR) performance. We vary $\tau \in \{0.25, 0.5, 0.75\}$ across both training sources and report per-benchmark accuracy and average accuracy. The default threshold used in the main experiments is $\tau = 0.5$.

L ASR Reward-Signal Ablation

To separate scaffold-provided structure from reward densification, Table 8 compares two additional ASR variants on the same scaffolded prompts. *Scaffold + Final Reward* keeps the scaffolded input format but rewards only the final main answer, removing all sub-answer rewards; it improves only modestly over outcome-only GRPO and remains far below ASR, indicating that the scaffold prompt itself is not the main source of the gain. *Scaffold + Independent Rewards* rewards each correct sub-answer independently, without the prefix-consistency gate; it performs better than final-only scaffolding but remains below ASR, showing that dense sub-answer rewards are important and that enforcing prerequisite order further improves credit assignment.

M Scaffold Construction Statistics

Table 9 summarizes the coverage and granularity of the automated scaffold construction pipeline across the two training sources. "Filtered" refers to problems automatically excluded because their ground-truth answers do not satisfy the verifiability constraint, such as proof-based or open-ended problems. "Avg. Subs" denotes the average number of sub-problems per successfully decomposed question.

N Human Validation of Scaffold Prerequisite Chains

To verify that the generated scaffolds genuinely form prerequisite chains rather than loosely related sub-questions, we randomly sampled 200 scaffolds from the training data and conducted manual annotation. Two annotators independently examined each scaffold and judged whether every later sub-question explicitly depends on the result of at least one earlier sub-question (i.e., no dependency breakage or redundant sub-questions that bypass the chain). A scaffold is marked invalid if any posterior sub-question can be answered without using the result of its designated predecessor.

All 200 inspected scaffolds (100%) satisfy strict prerequisite-chain integrity: each sub-question builds upon previous sub-question results, and no dependency breakage is observed. This confirms that the automated validity filter (Eq. 1) effectively ensures well-formed prerequisite structures before RL training.

O Protocol Validation: Step-Label Agreement with Human Experts

The Step-Quality Evaluation Protocol's headline metric (*useful*-step ratio S_u) and QPR's positive reward term both depend on the binary distinction between *useful* steps and the four low-value categories. We therefore directly validate this binary distinction against human experts.

Annotation setup. We randomly sample 50 reasoning trajectories generated by Qwen3-8B-Instruct (the model used in our main experiments) across our evaluation benchmarks (GPQA, MATH500, AIME). All trajectories are pre-parsed into atomic steps by the same LLM Judge used in evaluation; human annotators see the trajectory with step boundaries already drawn and do not re-segment. We deliberately scope this study to

Training Data	τ	GPQA$_{@1}$	MATH500$_{@1}$	AIME24$_{@8}$	AIME25$_{@8}$	Avg
DAPO-Math	0.25	51.52	89.55	67.08	47.08	63.81
	0.50	51.52	**90.23**	65.00	**56.67**	**65.86**
	0.75	51.52	85.71	63.75	55.42	64.10
Big-Math	0.25	40.91	87.97	57.92	50.00	59.20
	0.50	**53.03**	87.22	**59.17**	**50.00**	**62.36**
	0.75	50.00	85.71	45.42	35.83	54.24

Table 7: Sensitivity of ASR (Stage 1) to the scaffold routing threshold τ. Bold indicates the best average accuracy within each training-data group. $\tau = 0.5$ achieves the best average accuracy on both data sources.

Variant	Scaffold	Sub-answer reward	Avg.
GRPO	–	–	55.80
Scaffold + Final Reward	✓	–	57.63
Scaffold + Independent Rewards	✓	independent	60.19
ASR only	✓	prefix-consistent	**65.86**

Table 8: Reward-signal ablation for ASR. Scaffold + Final Reward uses the same answer-hidden scaffolded prompts as ASR but keeps only the final main-answer reward. Scaffold + Independent Rewards rewards each scaffolded sub-answer independently, removing the prefix-consistency gate. Avg. is the arithmetic mean of GPQA, MATH500, AIME24, and AIME25 accuracies under the same evaluation protocol as Table 2.

label-level agreement *conditional on* the protocol's segmentation, since this matches how the protocol is used at evaluation time; segmentation reliability is checked separately by the scaling-discrimination study (Appendix E), where the protocol cleanly separates five model sizes. The 50 trajectories contain approximately 1,250 atomic steps in total (mean 25 steps per trajectory), approximately balanced across the three benchmarks.

Asymmetric annotation protocol. To control annotation cost while keeping the LLM Judge consistent with how it is actually used during evaluation, we adopt an asymmetric setup:

- **Human annotators** mark each step as *useful* or *non-useful* only. Two domain experts with PhD-level training in mathematics and physics annotate independently. We do not require them to discriminate among the four low-value categories, since the binary distinction (a) determines the protocol's headline metric S_u and the sign of QPR's positive reward, (b) is the boundary that experts can adjudicate quickly and reliably, and (c) avoids inflating disagreement on inter-category boundaries that are not consumed downstream.
- **LLM Judge** produces the same five-class labels (*useful*, *mechanical*, *redundant*, *reversion*, *error*) it produces during evaluation, with no special configuration for this study. For comparison against the human binary labels, we collapse the four low-value classes into *non-useful*; this is the same aggregation convention used in QPR's Φ penalty and in reporting the useful-step ratio S_u.

Annotators receive the same rubric used by the LLM Judge (Appendix H) but are blinded to the LLM Judge's outputs and to which method produced each trajectory.

Step-level binary agreement. Table 10 reports binary agreement between expert consensus and the LLM Judge, with inter-annotator agreement (IAA) between the two human experts as the human ceiling. Expert consensus is defined as the agreed label when both experts match (covering 90.0% of steps); on the remaining 10.0% the LLM is compared against each expert separately and scored values are averaged.

Per-class binary metrics. Table 11 reports per-class precision/recall/F1 of the LLM Judge against expert consensus for the *useful* and *non-useful* classes, alongside the corresponding inter-annotator F1.

Scope of this validation. This study validates the *protocol* only: it shows that the LLM Judge's per-step useful/non-useful labels are consistent with expert labels at near-ceiling levels. The pairwise preference results in Section 5.6 are a separate piece of evidence about *model output quality* (SCOPE-RL vs. GRPO under expert preference) and are not used here as protocol-validation evidence.

P Pairwise Expert Evaluation

Table 12 reports the full dimension-level results of the pairwise expert evaluation. Three do-

Dataset	Total	Success	Filtered	Failed	Succ. Rate	Avg. Subs	Max	Min
DAPO-Math (2.4K)	2,400	2,382	16	2	99.3%	3.84	13	2
DAPO-Math (full)	17,406	17287	116	3	99.3%	3.80	15	1
Big-Math (2.4K)	2,400	2,251	149	0	93.8%	3.01	9	1
Big-Math (full)	12,400	11,846	554	0	95.5%	2.97	11	1

Table 9: Statistics of the automated decomposition pipeline across datasets.

Pair	Accuracy	Cohen's κ
Expert A vs. Expert B (IAA, ceiling)	0.90	0.79
LLM Judge vs. Expert consensus	0.88	0.75
LLM Judge vs. Expert A	0.87	0.73
LLM Judge vs. Expert B	0.88	0.75

Table 10: Step-level useful/non-useful binary agreement on 1,250 steps from 50 Qwen3-8B-Instruct trajectories. The LLM Judge here refers to the evaluation judge (Gemini-3-flash-preview (Pichai et al., 2025)), the same model used to produce all diagnostic metrics in the main results. Cohen's κ is within 0.04 of the human ceiling, indicating that the protocol's core useful/non-useful distinction is approximately as consistent with each expert as the experts are with each other.

Class	Precision	Recall	F1	IAA F1
useful	0.88	0.90	0.89	0.91
non-useful	0.87	0.85	0.86	0.89

Table 11: Per-class agreement of the LLM Judge against expert consensus on 1,250 steps. The IAA F1 column reports the corresponding per-class F1 between the two human experts. The LLM Judge's F1 is within 0.02–0.03 of the human ceiling on both classes, supporting the use of the useful-step ratio S_u as a reliable diagnostic metric and as input to QPR's positive reward term.

main experts (Ph.D.-level mathematicians) independently judge 200 examples where both GRPO and SCOPE-RL produce the correct final answer. For each example and dimension, the majority vote among the three annotators determines the winner.

Dimension	SCOPE-RL	GRPO
Clarity	131	69
Conciseness	144	56
Non-redundancy	138	62
Logical Coherence	119	81
Overall	141	59

Table 12: Pairwise expert evaluation (count out of 200) on examples where both GRPO and SCOPE-RL produce the correct final answer. Winner is determined by majority vote among three domain experts.

Q Robustness Results: GSPO Backend and 0.6B Scale

Table 13 reports results when SCOPE-RL is instantiated with the GSPO optimizer instead of GRPO (panel a), and on the smaller Qwen3-0.6B-Instruct model (panel b). Both experiments use DAPO-Math as the training source and follow the same two-stage protocol as the main experiments. The GSPO results (Table 13a) confirm that SCOPE-RL's gains are not tied to a single optimizer implementation: ASR improves accuracy from 61.60% to 66.04%, and QPR further raises it to 66.93% while improving the useful-step ratio from 60.89% to 63.27% and reducing token usage by 13.1%, both relative to the GSPO baseline. The 0.6B results (Table 13b) show that even under limited model capacity, the stage-wise decomposition remains effective: ASR lifts average accuracy from 26.06% to 30.39%, and QPR improves it to 32.06% while raising Useful% from 51.79% to 58.43% and reducing tokens by 22.9%, both relative to the GRPO baseline.

Method	Accuracy (%) ↑					Reasoning Quality ↑		Avg. Tokens ↓
	GPQA$_{@1}$	MATH500$_{@1}$	AIME24$_{@8}$	AIME25$_{@8}$	Avg	Useful%	1st-Err-Pos	
(a) GSPO optimizer on Qwen3-8B-Instruct								
GSPO	48.48	85.82	57.08	55.00	61.60	60.89	0.736	6,719
ASR only	**50.51**	90.30	65.83	57.50	66.04	58.51	0.697	7,460
SCOPE-RL	50.00	**91.04**	**67.92**	**58.75**	**66.93**	**63.27**	**0.752**	**5,836**
(b) GRPO optimizer on Qwen3-0.6B-Instruct								
Qwen3-0.6B-Instruct	20.20	32.84	3.75	8.75	16.39	40.24	0.393	3,467
GRPO	15.66	51.49	14.58	22.50	26.06	51.79	0.574	10,188
ASR only	20.20	59.70	16.67	25.00	30.39	55.15	0.579	9,367
SCOPE-RL	**21.72**	**61.94**	**18.33**	**26.25**	**32.06**	**58.43**	**0.624**	**7,852**

Table 13: Robustness checks on alternative optimizer and model scale (both trained with DAPO-Math). (a) GSPO optimizer on Qwen3-8B-Instruct: SCOPE-RL's two-stage gains transfer to an alternative RL optimizer. (b) GRPO optimizer on Qwen3-0.6B-Instruct: despite limited model capacity, the two-stage decomposition yields consistent gains in accuracy, reasoning quality, and efficiency.

R Case Study: Reasoning Trace Comparison

Problem. Let a, b, c be distinct real numbers. Find the degree of $p(x) = \frac{(x-b)(x-c)}{(a-b)(a-c)} + \frac{(x-a)(x-c)}{(b-a)(b-c)} + \frac{(x-a)(x-b)}{(c-a)(c-b)}$.
Answer: 0.

GRPO (Baseline)

... We are to find the **degree** of the polynomial $p(x)$, given that a, b, c are distinct real numbers. Let me analyze this expression. ...

This expression resembles the **Lagrange interpolation polynomial**.

So if we take $f(x) = x^2$, then this is the Lagrange interpolation of x^2 at three points. Since interpolating a degree-2 polynomial with three points gives exactly that polynomial: $p(x) = x^2$. Therefore, the degree is **2**.

But let me verify this with an example. Let me take $a{=}0,\ b{=}1,\ c{=}{-1}$.

First term: $\frac{(x-1)(x+1)}{(-1)(1)} = -x^2 + 1$; Second: $\frac{x(x+1)}{2} = \frac{x^2+x}{2}$; Third: $\frac{x(x-1)}{2} = \frac{x^2-x}{2}$

Add: $(-x^2+1) + \frac{2x^2}{2} = -x^2 + 1 + x^2 = 1$

Wait, that gives 1? But that contradicts degree 2. So something is wrong with my reasoning.

Why? If $f(x) = x^2$ and we interpolate with 3 points, it should give x^2... but the result is 1? Wait, maybe I confused $f(x)$?

Let me check $p(a), p(b), p(c)$. ... All equal 1. So $p(x)$ is 1 at three distinct points.

But a polynomial of degree ≤ 2 that equals 1 at three distinct points must be the constant 1. But that would mean $p(x) = 1$, degree 0. That contradicts my earlier conclusion... Let me reconsider.

But why did I think it was x^2?

I confused the Lagrange interpolation of $f(x) = x^2$ with the sum of basis polynomials. In our case, $p(x) = L_a(x) + L_b(x) + L_c(x) = 1$. Because for any x, the sum of the basis polynomials is 1. So $p(x) = 1$, degree is 0.

But why is the problem non-trivial? Maybe the problem is different? ... Assuming correct, the answer is 0. ... Let me check once more. ...

So the answer is 0. Therefore, the degree is $\boxed{0}$.

Stage 1 (ASR)

... We are to find the degree of this polynomial, given that a, b, c are distinct real numbers. ...

Each numerator is a quadratic in x; each denominator is a constant (since a, b, c are constants). So each term is a quadratic polynomial. But when added together, the sum may simplify to a lower degree.

The question is: what is the degree of the resulting polynomial?

This expression resembles the **Lagrange interpolation polynomial**. Specifically, it is the sum of the Lagrange basis polynomials for interpolation at three points $(a, 1), (b, 1), (c, 1)$.

A quadratic polynomial taking the value 1 at three distinct points must be the constant 1. Therefore, $p(x) = 1$, and the degree is 0.

But wait, that seems surprising. Let me verify with an example.

$a{=}0,\ b{=}1,\ c{=}2$. First: $\frac{(x-1)(x-2)}{2}$; Second: $-x^2 + 2x$; Third: $\frac{x(x-1)}{2}$

Combine: $\frac{(x^2-3x+2)+(-2x^2+4x)+(x^2-x)}{2} = \frac{2}{2} = 1$

So indeed, $p(x) = 1$, a constant polynomial. Therefore, the degree is 0.

So the degree of the polynomial is $\boxed{0}$.

Stage 2 (QPR)

... We are to find the degree of the polynomial $p(x)$. ...

The denominators are constants (since a, b, c are fixed), and the numerators are quadratic polynomials in x. So each term is a quadratic polynomial divided by a constant. ... when we add them together, the result may simplify to a lower-degree polynomial.

Specifically, if we have three distinct points a, b, c, then the **Lagrange basis polynomials** are:

$$L_a(x) = \frac{(x-b)(x-c)}{(a-b)(a-c)},$$
$$L_b(x) = \ldots,\ L_c(x) = \ldots$$

Then the sum $L_a(x) + L_b(x) + L_c(x)$ is the sum of the Lagrange basis polynomials for interpolation at three points. But the sum of all Lagrange basis polynomials is the constant polynomial 1, because for any x, the sum of $L_a(x) + L_b(x) + L_c(x)$ equals 1.

Therefore, the given expression $p(x)$ is equal to 1 for all x, hence it is the constant polynomial 1. Therefore, the degree is 0.

Quick check: $a{=}0, b{=}1, c{=}2$. ... numerator is 2, denominator is 2: $p(x) = 1$. ✓

So indeed, $p(x) = 1$, a constant polynomial. Therefore, the degree of $p(x)$ is $\boxed{0}$.

Useful insight **Error** **Reversion** **Redundant** **Mechanical / omitted**