LC-GRPO: VERIFIER-GATED LOGPROB LOCALIZATION FOR OUTCOME-SUPERVISED REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Outcome-supervised reinforcement learning improves language-model reasoning without dense process labels, but a sequence-level reward assigns the same credit to every token in a sampled solution. This is harmful when an incorrect response contains a useful prefix followed by a local failure. We introduce Localized-Credit Group Relative Policy Optimization (LC-GRPO), a lightweight method that reuses behavior-policy log-probabilities recorded during rollout. A verifier first determines which responses failed. For each failed response, LC-GRPO detects the largest thresholded drop in length-normalized step log-confidence, protects the preceding prefix, and applies bounded non-positive credit to the estimated failure suffix. Correct-response advantages are unchanged. The method requires no process labels, external critic, or additional model forward pass. Across two Qwen3 scales and six mathematical and code benchmarks, LC-GRPO improves the six-task average over matched-budget GRPO by 4.2 and 4.0 points at the 4B and 8B scales, respectively, and over the strongest baseline by 2.2 points at both scales. On Qwen3-8B-Base, it also lowers expected calibration error from 0.172 to 0.083 relative to GRPO and locates the annotated first error exactly in 78.1% of filtered ProcessBench failures.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become a practical route to improving mathematical and programmatic reasoning (Shao et al., 2024; DeepSeek-AI et al., 2025). A verifier only needs to determine whether a final answer passes exact matching or executable tests. However, this creates a credit-assignment mismatch: the scalar outcome is copied to an entire response even though a long derivation may contain a correct prefix followed by one fatal transition. Standard Group Relative Policy Optimization (GRPO; Shao et al., 2024) therefore decreases the probability of every token in an incorrect response, including steps that should be preserved (Figure 1).

Token likelihoods provide a useful but imperfect endogenous signal. Absolute likelihood is influenced by style, length, and lexical familiarity, so a low-likelihood step is not necessarily wrong. The transition from a stable reasoning state to an unsupported one is often more informative than the absolute likelihood of either state. We consequently focus on the *adjacent confidence drop* between reasoning steps and use it only after the external verifier has declared the complete response incorrect. This design prevents confidence from replacing correctness: the verifier decides which

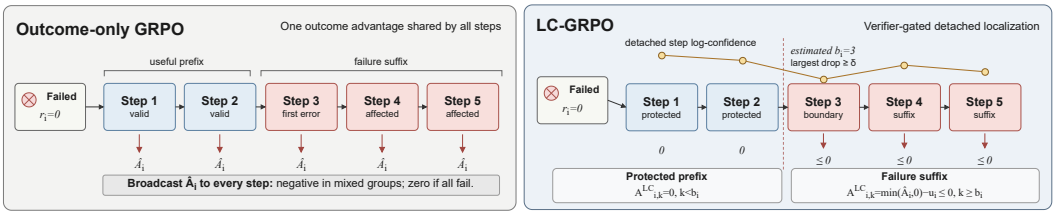

Figure 1: **Why outcome-only credit is insufficient.** GRPO penalizes an entire failed derivation, including its useful prefix. LC-GRPO estimates a verifier-gated failure boundary, protects preceding steps, and assigns bounded negative credit only to the suffix.

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.

trajectories failed, while internal confidence only decides where their negative update should begin and how strongly their suffix should be penalized.

The resulting update has three properties that are important for reliable outcome supervision. First, correct trajectories are optimized exactly as in GRPO. Second, the estimated valid prefix of an incorrect trajectory receives zero rather than negative advantage. Third, every step in its failure suffix receives a non-positive, bounded penalty that grows with that step's own confidence. Thus, unlike reward re-normalization based directly on confidence, the method cannot turn a low-confidence error into a positively reinforced sample. It also produces a learning signal for all-wrong groups, while making no claim to resolve all-correct groups.

Our contributions can be summarized as follows:

- We introduce a verifier-gated boundary detector based on adjacent changes in stored behavior-policy log-probabilities.
- We use the boundary to remove negative support from an estimated valid prefix while keeping every failed-response update non-positive.
- We evaluate the resulting update under matched rollout budgets at two model scales, with separate tests of task accuracy, boundary quality, confidence, component contributions, and systems cost.

2 Related Work

Outcome-supervised reasoning. GRPO and later RLVR variants remove the learned critic and optimize groups of sampled completions using verifiable outcomes (Shao et al., 2024; DeepSeek-AI et al., 2025). Dr. GRPO corrects length- and difficulty-dependent biases in the estimator (Liu et al., 2025), while DAPO and GSPO improve clipping, sampling, and optimization stability (Yu et al., 2025; Zheng et al., 2025). These methods primarily refine the sequence-level optimization rule; they do not identify which suffix of a failed derivation should receive negative credit.

Process supervision and error localization. Process reward models (PRMs) can score intermediate steps (Uesato et al., 2022; Lightman et al., 2023), and ProcessBench directly tests whether a verifier can identify the first erroneous step (Zheng et al., 2024). Such models add supervision and inference cost, and their step labels may not transfer across domains. Our method uses no external process labels during training. ProcessBench is used only as a diagnostic test of the endogenous boundary detector.

The closest process-supervised method is Verifiable Prefix Policy Optimization (VPPO), which uses an external process reward model to identify a first error and assigns positive reward to a verified prefix (Liu et al., 2026). LC-GRPO addresses the same prefix-suppression problem but uses only the final verifier and stored behavior-policy log-probabilities. It protects a prefix with zero advantage rather than declaring it correct or rewarding it. This makes the signal cheaper but also less semantic: its boundary is an estimate from policy likelihood, not a process-model judgment.

Confidence-aware policy optimization. Recent work uses uncertainty or confidence to shape exploration and credit, including UCAS, CAPO, and DCPO (Xie et al., 2026; Wang et al., 2026b; Ma et al., 2026). The key distinction is structural: we do not re-rank whole trajectories by confidence or allow confidence to alter the sign of an outcome-derived advantage. Confidence is verifier-gated, detached from optimization, and restricted to selecting and weighting the suffix of an already incorrect trajectory. This distinction is tested through sign-preservation, localization, and matched-budget ablations. Step-level Advantage Selection (SAS) also masks selected steps in failed rollouts (Wang et al., 2026a). SAS protects a fixed ratio of individually high-confidence steps, whereas our update selects one contiguous suffix from an adjacent confidence drop and applies a bounded negative correction to that suffix. We therefore include SAS as a direct baseline.

LC-ERD decomposes trajectory rewards through a variational logic potential and multi-agent value decomposition (Chen et al., 2026). Its goal is a general dense estimate of step utility. In contrast, LC-GRPO estimates only one contiguous support boundary and then applies a verifier-constrained update. The present claim is therefore narrower: stored log-probabilities can provide a useful, low-cost localization signal, not a general semantic process reward.

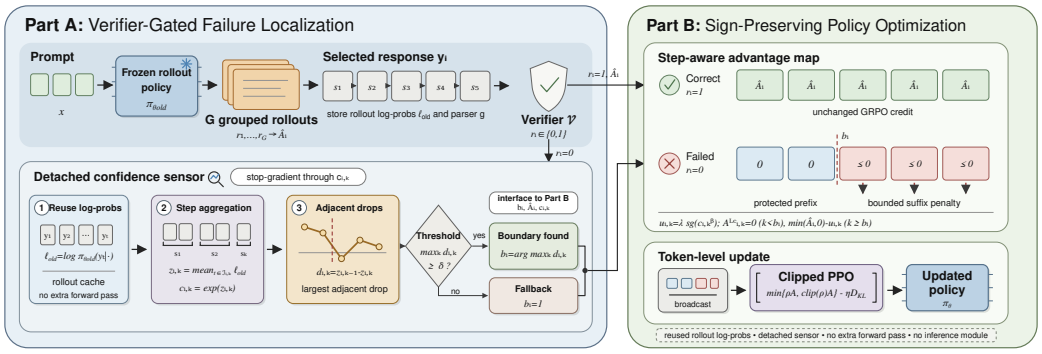

Figure 2: **Overview of LC-GRPO.** The verifier first gates confidence processing to incorrect rollouts. The trajectory sensor converts stored token log-probabilities into normalized step log-confidence, detects the largest thresholded adjacent drop, and returns an estimated failure boundary. The advantage rule preserves correct-rollout GRPO updates, masks the estimated valid prefix of a failed rollout, and applies a detached, step-specific non-positive correction to its suffix. The shaped advantages are then broadcast to tokens for the standard clipped Proximal Policy Optimization (PPO) update (Schulman et al., 2017). No gradient passes through the confidence sensor, and no additional model forward pass is introduced.

3 Motivation and Design

Figure 1 separates two phenomena that a sequence reward conflates. An incorrect response can contain both useful early steps and a later failure. Absolute likelihood alone is also insufficient: rare but valid notation may have low likelihood, while a familiar error may have high likelihood. We therefore use the final verifier to decide *which* responses failed, a relative likelihood change to estimate *where* their negative update should begin, and bounded step confidence only to determine *how strongly* the suffix is penalized. Section 7 tests the boundary against independent ProcessBench labels rather than treating this schematic motivation as empirical evidence.

4 Preliminaries

Let $x \sim \mathcal{D}$ be a prompt and let the rollout policy $\pi_{\theta_{\text{old}}}$ sample a group $\{y_i\}_{i=1}^{G}$. Each response $y_i = (y_{i,1}, \ldots, y_{i,T_i})$ is assigned a binary reward

$$r_i = \mathcal{V}(x, y_i) \in \{0, 1\},$$

where $\mathcal{V}$ is an exact-answer or executable-test verifier. Standard GRPO computes

$$\overline{r} = G^{-1} \sum_{j=1}^{G} r_j, \qquad \widehat{A}_i = \frac{r_i - \overline{r}}{\sqrt{G^{-1} \sum_{j=1}^{G} (r_j - \overline{r})^2 + \epsilon_A}}. \tag{1}$$

The same $\widehat{A}_i$ is then broadcast to all response tokens. For an incorrect response in a mixed group, this penalizes both the useful prefix and the actual failure suffix. If all responses are incorrect, $\widehat{A}_i = 0$ for every i, so standard GRPO supplies no preference signal for that prompt.

We assume that a deterministic parser partitions each response into K_i contiguous steps $s_{i,1:K_i}$. A step ends at a newline, a sentence boundary, or a code-block boundary; empty segments are merged with the preceding step. Let $\kappa_i(t)$ map token t to its step index. The same parser is fixed for all methods and reported with the released code.

5 Localized-Credit GRPO

Figure 2 summarizes the complete data flow.

5.1 From Outcome Credit to Localized Credit

For a boundary b, let $\mathcal{P}_i(b) = \{t : \kappa_i(t) < b\}$ and $\mathcal{S}_i(b) = \{t : \kappa_i(t) \geq b\}$ denote prefix and suffix token sets. Under the local-failure assumption, an incorrect response has an unobserved first-error boundary $b_i^\star$: steps before it remain useful, while its suffix should receive negative credit. Let $h_{i,t} = \nabla_\theta \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})$. Standard GRPO instead produces

$$g_i^{\mathrm{GRPO}} = \frac{\widehat{A}_i}{T_i} \left(\sum_{t \in \mathcal{P}_i(b_i^\star)} h_{i,t} + \sum_{t \in \mathcal{S}_i(b_i^\star)} h_{i,t} \right). \tag{2}$$

For an incorrect completion in a mixed group, $\widehat{A}_i < 0$. The first sum is therefore an avoidable prefix-suppression term: it reduces the probability of tokens that precede the assumed failure. Because $b_i^\star$ is unavailable under outcome-only supervision, LC-GRPO estimates its support from rollout likelihoods and changes only the location and magnitude of negative credit. It never uses confidence to overturn the verifier label.

5.2 Detached Step Confidence

For step $s_{i,k}$, let $\mathcal{I}_{i,k}$ contain its token indices. We separate the signal used for localization from the bounded signal used for weighting:

$$z_{i,k} = \frac{1}{|\mathcal{I}_{i,k}|} \sum_{t \in \mathcal{I}_{i,k}} \log \pi_{\theta_{\mathrm{old}}}(y_{i,t} \mid x, y_{i,<t}), \qquad c_{i,k} = \exp(z_{i,k}). \tag{3}$$

Here $z_{i,k} \leq 0$ is length-normalized step log-confidence and $c_{i,k} \in (0, 1]$ is its bounded counterpart. We use the untruncated behavior-policy log-probability stored for the PPO ratio. Temperature and top-p determine which token is sampled but are not applied again when constructing $z_{i,k}$. We stop gradients through both statistics, so confidence is metadata for shaping rather than a reward that the policy can increase directly.

5.3 Verifier-Gated Failure Boundary

We measure a relative transition in log-confidence:

$$d_{i,k} = z_{i,k-1} - z_{i,k}, \qquad k = 2, \ldots, K_i. \tag{4}$$

Only an incorrect trajectory is localized:

$$b_i = \begin{cases} \arg\max_{2 \leq k \leq K_i} d_{i,k}, & \max_{2 \leq k \leq K_i} d_{i,k} \geq \delta, \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

The threshold rejects small fluctuations. If no drop passes it, or if the response contains one step, $b_i = 1$ conservatively recovers full-response penalization. We call b_i an estimated failure boundary rather than a ground-truth error label.

Shift invariance. Suppose trajectory-level style or difficulty adds a constant a_i to every step log-confidence, $z'_{i,k} = z_{i,k} + a_i$. Then $d'_{i,k} = z'_{i,k-1} - z'_{i,k} = d_{i,k}$, so Eq. equation 5 selects the same boundary. The detector is therefore insensitive to a trajectory-wide log-confidence offset, unlike selection by the absolute minimum confidence.

5.4 Boundary Semantics and Degenerate Cases

The maximum-drop rule should not be read as a calibrated probability that a step is wrong. It is a *support estimator*: conditioned on the external verifier declaring the response incorrect, it identifies the strongest thresholded transition after which negative credit is allowed to act. Ties are resolved in favor of the earliest transition. This restricted role is important because a confident but invalid step may still have high absolute likelihood, while a correct step containing rare notation may have low likelihood. Selecting a relative transition avoids treating either event as sufficient evidence of failure. The threshold δ further separates a localized decision from the conservative fallback $b_i = 1$; it is tuned only on development prompts and is fixed across test benchmarks.

Three degenerate reward groups require explicit treatment. In a mixed group, an incorrect rollout has $\widehat{A}_i < 0$, so localization removes the avoidable prefix part of the ordinary GRPO gradient. In an all-wrong group, $\widehat{A}_i = 0$, but the confidence penalty supplies a bounded suffix-only learning signal instead of discarding the prompt. In an all-correct group, GRPO and LC-GRPO both produce zero group-relative advantage. We do not inject a confidence reward into this last case because doing so would optimize self-likelihood without evidence of improved correctness. Hence confidence can refine the support of a verifier-derived negative update, but cannot create a positive update by itself.

Parser independence. The step parser uses only response text and is applied before verifier evaluation. It never observes rewards, reference answers, or ProcessBench boundaries. One-step responses and parser failures use $b_i = 1$. These choices make the detector deterministic and prevent benchmark annotations from leaking into training-time localization.

5.5 Sign-Preserving Suffix Advantage

Let $\lambda > 0$ cap the confidence correction and $\beta \geq 1$ control its curvature. Each suffix step receives its own detached penalty

$$u_{i,k} = \lambda \, \mathrm{sg}(c_{i,k}^{\beta}) \in (0, \lambda], \tag{6}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. The shaped step advantage is

$$A_{i,k}^{\mathrm{LC}} = \begin{cases} \widehat{A}_i, & r_i = 1, \\ 0, & r_i = 0 \text{ and } k < b_i, \\ \widehat{A}_i - u_{i,k}, & r_i = 0 \text{ and } k \geq b_i. \end{cases} \tag{7}$$

Correct completions retain their GRPO advantage, plausible failed prefixes are protected, and an incorrect completion cannot obtain positive shaped advantage.

5.6 Token-Level Clipped Objective

Write $\ell_{\theta,i,t} = \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})$, with $\ell_{\mathrm{old},i,t}$ its rollout counterpart, and define

$$\rho_{i,t}(\theta) = \exp(\ell_{\theta,i,t} - \ell_{\mathrm{old},i,t}). \tag{8}$$

Let $\bar{\rho}_{i,t} = \mathrm{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon)$. After broadcasting each step advantage to its tokens, $A_{i,t}^{\mathrm{LC}} = A_{i,\kappa_i(t)}^{\mathrm{LC}}$, we maximize

$$\mathcal{J}(\theta) = \frac{1}{G} \sum_{i=1}^{G} \frac{1}{T_i} \sum_{t=1}^{T_i} \min\{\rho_{i,t} A_{i,t}^{\mathrm{LC}}, \bar{\rho}_{i,t} A_{i,t}^{\mathrm{LC}}\} - \eta D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}}). \tag{9}$$

This response-normalized reduction matches the gradient convention in Eq. equation 2: every sampled response has equal weight regardless of its realized length. Token-level ratios avoid multiplying probabilities across an entire step. Appendix B.1 summarizes the scope of these guarantees and the cost of computing the update.

6 Properties and Proofs

Proofs of all propositions are given in Appendix B.

6.1 Support and Sign

Proposition 1. *At the advantage-assignment level, LC-GRPO leaves the GRPO advantage unchanged for every correct response. For an incorrect response, it assigns zero advantage before the estimated boundary b_i and non-positive advantage from b_i onward.*

6.2 Bounded Confidence Correction

Let $\widetilde{A}_{i,k}$ denote the advantage before applying the confidence correction: it equals $\widehat{A}_i$ on a correct response, zero on an incorrect prefix, and $\widehat{A}_i$ on an incorrect suffix.

Proposition 2. *The confidence correction changes the pre-correction advantage by at most λ:*

$$\left| A_{i,k}^{\mathrm{LC}} - \widetilde{A}_{i,k} \right| \leq \lambda. \tag{10}$$

The bound prevents the confidence term from producing an arbitrarily large update. It does not require the confidence value itself to be a calibrated estimate of step correctness.

6.3 Boundary-Error Stability

Let $h_{i,t}$ be the token-level score function defined in Section 5, let $\mathcal{S}_i(b)$ denote the set of suffix tokens induced by boundary b, and let $g_i(b)$ be the detached, unclipped score-function contribution obtained using that boundary.

Proposition 3. *Assume $\|h_{i,t}\| \leq L$ and $|\widehat{A}_i| \leq M$. For an oracle boundary $b_i^\star$,*

$$\|g_i(b_i) - g_i(b_i^\star)\| \leq \frac{L(M+\lambda)}{T_i} \left| \mathcal{S}_i(b_i) \triangle \mathcal{S}_i(b_i^\star) \right|, \tag{11}$$

where $\triangle$ denotes the symmetric difference between two token sets.

6.4 Detector-Margin Robustness

Suppose the observed step log-confidence is $\widetilde{z}_{i,k} = z_{i,k} + e_{i,k}$, where $|e_{i,k}| \leq \xi$. Let $d_{i,(1)}$ and $d_{i,(2)}$ be the largest and second-largest adjacent drops computed from the unperturbed confidence sequence.

Proposition 4. *If*

$$d_{i,(1)} - d_{i,(2)} > 4\xi, \qquad d_{i,(1)} \geq \delta + 2\xi, \tag{12}$$

then the perturbed detector selects the same non-fallback boundary as the unperturbed detector.

This result gives a sufficient condition rather than a necessary one. When the margin is small, the conservative fallback remains part of the method.

Propositions 1–4 describe the support, sign, magnitude, and local stability of the constructed advantage. They do not establish semantic correctness of the estimated boundary, convergence of the complete clipped optimization procedure, or task-level performance guarantees.

7 Experiments

7.1 Research Questions

Our experiments aim to answer the following four research questions:

RQ1: Does suffix-localized advantage shaping improve pass@1 under the same rollout and update budget?

RQ2: Does it improve the alignment between endogenous confidence and verifier correctness?

RQ3: Does the maximum confidence drop identify the first erroneous step better than simpler likelihood heuristics?

RQ4: Are gains attributable to localization and sign preservation rather than additional compute?

Claim-to-evidence separation. Each question is tested with separate evidence: pass@1 under matched budgets (RQ1), calibration and probing (RQ2), ProcessBench first-error labels (RQ3), and grouped ablations that alter one component at a time, together with throughput and peak memory (RQ4). The diagnostics establish agreement with the intended mechanism; they are not treated as causal proof of downstream gain.

Table 1: **Reasoning performance under matched generation cost.** Pass@1 (%), mean of three training seeds. Math Avg. covers AIME24, AIME25, MATH-500, and AMC; Code Avg. covers LiveCodeBench (LCB) and BigCodeBench-Hard (BCB-H); All Avg. covers all six tasks (per-task scores in Table 5). Rel. Gen.: training tokens relative to GRPO. **Best**, strongest baseline; superscript: gain over the strongest baseline.

		Mathematical reasoning			Code reasoning			Summary	
Model	Method	Math Avg.	AIME25	MATH-500	Code Avg.	LCB	BCB-H	All Avg.	Rel. Gen.
Qwen3-4B-Base									
	SFT (warm start)	50.9	31.5	75.8	29.5	35.1	23.8	43.8	–
Outcome-only RLVR									
	GRPO	54.6	35.7	78.2	32.2	38.0	26.4	47.1	1.00×
	Dr. GRPO	55.2	36.4	78.5	32.6	38.4	26.8	47.6	1.00×
	DAPO	55.8	37.2	78.9	33.0	38.8	27.2	48.2	1.00×
Step- and confidence-aware RLVR									
	SAS	56.2	37.8	79.1	33.3	39.1	27.5	48.6	1.00×
	UCAS	56.5	38.1	79.2	33.6	39.3	27.8	48.8	1.00×
	CAPO	56.8	38.5	79.4	33.8	39.5	28.0	49.1	1.00×
	LC-GRPO	**59.0**$^{+2.2}$	**41.2**$^{+2.7}$	**80.6**$^{+1.2}$	**35.8**$^{+2.0}$	**41.6**$^{+}$	**30.0**$^{+2.}$	**51.3**$^{+2.2}$	1.00×
Qwen3-8B-Base									
	SFT (warm start)	63.3	45.0	84.2	39.8	45.5	34.0	55.5	–
Outcome-only RLVR									
	GRPO	66.8	49.2	86.0	42.7	48.3	37.0	58.7	1.00×
	Dr. GRPO	67.3	49.8	86.2	43.1	48.7	37.4	59.2	1.00×
	DAPO	67.8	50.7	86.5	43.6	49.2	37.9	59.7	1.00×
Step- and confidence-aware RLVR									
	SAS	68.1	51.0	86.6	43.8	49.4	38.1	60.0	1.00×
	UCAS	68.3	51.3	86.7	44.1	49.6	38.5	60.2	1.00×
	CAPO	68.6	51.6	86.8	44.3	49.8	38.7	60.5	1.00×
	LC-GRPO	**70.7**$^{+2.1}$	**54.4**$^{+2.8}$	**88.0**$^{+1.2}$	**46.7**$^{+2.4}$	**52.1**$^{+}$	**41.2**$^{+2.}$	**62.7**$^{+2.2}$	1.00×

7.2 SETUP

Training and matching. We train Qwen3-4B-Base and Qwen3-8B-Base policies (Yang et al., 2025) separately for mathematics and code from shared domain-specific supervised fine-tuning (SFT) checkpoints. Every method uses $G = 8$, a prompt batch size of 128, an 8192-token response limit, 500 updates, AdamW (Loshchilov & Hutter, 2019) with learning rate 10^{-6}, $\varepsilon = 0.2$, and $\eta = 0.01$. We set $(\delta, \lambda, \beta) = (0.10, 0.4, 2)$ on development prompts. Prompt batches, decoding, generated-token caps, update limits, checkpoint selection, and three training seeds are matched. Stored untruncated behavior-policy log-probabilities are parsed and reduced on the CPU with gradients stopped.

Evaluation. We report pass@1 on AIME 2024/2025, MATH-500 (Hendrycks et al., 2021; Lightman et al., 2023), AMC, a date-bounded LiveCodeBench release (Jain et al., 2025), and BigCodeBench-Hard (Zhuo et al., 2025), using 16 completions per problem at temperature 0.6, top-$p = 0.95$. All Avg. is the unweighted mean of the six tasks. We compare GRPO, Dr. GRPO, DAPO, SAS, UCAS, and CAPO under equal generated tokens; DCPO is added for calibration diagnostics. Exact-answer and executable-test verifiers are fixed across methods, and normalized exact matching plus 5-gram MinHash removes evaluation overlap.

Reporting. Scores are three-seed means. Appendix A specifies the parser, and Appendix C gives the complete training, data-registry, verifier, tuning, and system protocol; Appendix D defines run-level standard deviations, paired 95% intervals, and Holm-adjusted tests; Appendix E defines VPPO controls matched by generated tokens and accelerator-hours including PRM calls. The complete diagnostic protocol is in Appendix F.

Table 2: **Mechanism diagnostics on Qwen3-8B-Base.** ECE/Brier/AUROC score q_i; OC-Err is the percentage of wrong trajectories with $q_i \geq 0.8$. Exact and ± 1 are exact and within-one-step ProcessBench localization accuracies in $[0, 1]$; Gap is $\pm 1 - \text{Exact}$. **Best** , strongest baseline .

	Calibration and overconfidence				First-error localization and representation			
Method	ECE↓	Brier↓	AUROC↑	OC-Err↓	Exact↑	±1↑	Probe-F1↑	Gap↓
GRPO	.172	.193	.689	81.2	.421	.619	.54	.198
Dr. GRPO	.165	.188	.701	78.5	.428	.626	.56	.198
SAS	.148	.176	.728	59.4	.612	.768	.62	.156
UCAS	.126	.162	.758	41.6	.684	.824	.68	.140
CAPO	.104	.148	.792	34.8	.652	.801	.72	.149
DCPO	.096	.143	.806	31.9	.438	.631	.74	.193
LC-GRPO	**.083**	**.132**	**.837**	**18.7**	**.781**	**.914**	**.79**	**.133**

Table 3: **Grouped ablation on Qwen3-8B-Base.** Δ: All Avg. change from the full method; Exact: first-error localization. Bold marks the complete method.

Failure-boundary estimation					Advantage support and sign				
Variant	Avg.↑	Δ	ECE↓	Exact↑	Variant	Avg.↑	Δ	ECE↓	Exact↑
LC-GRPO (full)	**62.7**	–	**.083**	**.781**	**LC-GRPO (full)**	**62.7**	–	**.083**	**.781**
Minimum-confidence boundary	60.7	−2.0	.108	.614	No prefix protection	60.4	−2.3	.121	.779
Token-level maximum drop	61.3	−1.4	.101	.699	Whole-response confidence penalty	59.8	−2.9	.129	.780
Random boundary	59.3	−3.4	.139	.223	Uniform suffix penalty	61.5	−1.2	.111	.777
No drop threshold	61.6	−1.1	.094	.724	Unconstrained confidence bonus	59.2	−3.5	.147	.775
Sentence-only parser	61.8	−0.9	.092	.738	Linear penalty ($\beta = 1$)	62.0	−0.7	.096	.780
					No all-wrong suffix signal	60.9	−1.8	.117	.779

7.3 Main Reasoning Performance

Overall efficacy and scale. Under matched generation budgets (Table 1), LC-GRPO raises All Avg. over GRPO by $4.2/4.0$ points on 4B/8B and exceeds CAPO, the strongest baseline, by 2.2 at both scales. Stable margins across scales argue against a small-model-only regularization effect.

Long-horizon and cross-domain behavior. The largest margin over CAPO occurs on AIME25 at both scales ($+2.7/+2.8$ for 4B/8B). On code, the margin on BigCodeBench-Hard grows from $+2.0$ to $+2.5$ with scale, consistent with late failures that lead uniform GRPO to suppress longer useful prefixes. MATH-500 has the smallest gain ($+1.2$ at both scales), consistent with its higher baseline accuracy and shorter solutions. Because the two domains are trained separately, these results show effectiveness in each domain under deterministic verification, not transfer between domains.

7.4 Mechanism Diagnostics

Protocol and findings. The label-independent score $q_i = \min_k c_{i,k}$ is evaluated with equal-mass expected calibration error (ECE), Brier score, and the area under the ROC curve (AUROC) for correctness; it is a length-sensitive diagnostic rather than a correctness probability. For ProcessBench, we teacher-force final-answer-incorrect examples with annotated first errors and apply the same fixed parser, detector, and preselected δ to every checkpoint. The frozen linear probe uses a $60/20/20$ split and never updates the policy. Full construction, sample counts, alignment rules, and uncertainty are in Appendix F.

Relative to GRPO, LC-GRPO lowers ECE from 0.172 to 0.083, raises AUROC from 0.689 to 0.837, and raises Probe-F1 from 0.54 to 0.79. Its exact first-error localization accuracy (78.1%) exceeds the strongest localization baseline, UCAS, by 9.7 points and the strongest calibration baseline, DCPO, by 34.3 points. The 13.3-point gap to within-one-step accuracy (91.4%) indicates that part of the remaining error reflects parser and annotation ambiguity.

7.5 Component Ablation

Findings. Minimum-confidence and random boundaries lose 2.0 and 3.4 All Avg. points, while removing prefix protection loses 2.3 with nearly unchanged localization. This separates sensor quality from the use of its boundary. Uniform suffix weighting loses 1.2, showing that confidence magnitude adds value beyond suffix masking. The unconstrained confidence bonus changes only the failed-suffix update to $A_{i,k}^{\text{bonus}} = \widehat{A}_i + u_{i,k}$, leaving the boundary and prefix mask unchanged; because it can assign positive credit to a failed suffix, it loses 3.5 points. Removing the all-wrong signal loses 1.8. Appendix G defines every variant and gives the expanded plots.

7.6 Efficiency and Robustness

The step parser and confidence reductions run on the CPU and reuse rollout log-probabilities. The implementation retains 98.6% of GRPO throughput and uses 0.7% more peak memory in the reported trainer; inference uses only the updated policy and therefore has the same modules as GRPO. These relative measurements show that the method does not obtain its accuracy gain by generating more training tokens or invoking an additional model. Appendix H describes the absolute systems accounting.

8 Discussion

Relation to process supervision. LC-GRPO assigns segment-specific updates but is not a process reward model. Unlike process models, which provide semantic step-level judgments, our sensor uses only policy-likelihood changes after the final verifier marks a response incorrect. This is cheaper and yields a clear sign rule, but the inferred boundary remains approximate.

Compatibility with other RLVR objectives. Our method modifies only the step advantage, leaving the verifier, rollout policy, token ratios, and KL term unchanged. It is therefore compatible with GRPO variants using token log-probabilities and group-relative advantages, though clipping and length normalization may differ. We use a shared trainer to control for these effects.

Expected range of benefit. The method helps most when failed responses have long valid prefixes, a clear confidence drop near the first error, or all samples fail. Benefits are smaller for short responses, first-step failures, and all-correct groups, where there is little negative signal to refine. These are predictions of the update rule, not tested subgroup results.

9 Limitations

Likelihood is not correctness: a familiar error can remain likely, while rare notation or a domain shift can create a false confidence drop. The detector selects one contiguous suffix, so it cannot represent several independent errors or a later recovery, and step parsing adds ambiguity in dense derivations and code. Because the method gates only on the final verifier, it cannot detect a flawed process that reaches a correct answer. The propositions constrain update sign and support given the estimated boundary, not its semantic correctness or the stability of clipped PPO. Finally, the confidence score can depend on response length and step count, and the experiments cover only deterministic mathematical and executable-code verification.

10 Conclusion

LC-GRPO uses rollout confidence drops to protect an estimated valid prefix and assign a bounded non-positive penalty to an incorrect suffix while leaving correct GRPO updates unchanged. Six-task performance tests efficacy; ProcessBench, calibration, and probing test the boundary mechanism; grouped ablations test boundary choice, support, and confidence direction. The guarantees concern the constructed update rather than semantic correctness, and the results support a narrow claim: stored behavior-policy likelihoods can provide a useful, low-cost localization signal when they are gated by an objective verifier.

AI USE STATEMENT

In this work, we used generative AI tools for [tasks with required disclosure]. We have not used generative AI tools for [other tasks with required disclosure], and [the rest of the required disclosure tasks] are not applicable to this work. Additionally, we used generative AI tools for [tasks with recommended disclosure]. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 5 specifies the complete update rule, and Section 6 states its support, sign, magnitude, and stability properties, which Appendix B proves. Appendix A gives the update pseudocode, the confidence-extraction and parsing rules, and the treatment of degenerate reward groups. Appendix C lists the shared training configuration, data and contamination controls, verifiers, and baseline tuning. Appendices D, E, F, G, and H define the statistical analysis, the matched-compute VPPO protocol, the diagnostic protocols, all ablation variants, and the systems accounting, respectively.

REFERENCES

Y. Chen, J. Jiang, D. Yu, Z. Wu, J. Liu, J. Han, X. Guo, J. Qi, Y. Li, Y. Zhang, and I. King. LC-ERD: Mining latent logic for self-evolving reasoning via consistency-regulated reward decomposition. *arXiv preprint arXiv:2605.24005*, 2026.

DeepSeek-AI et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*, 2025.

H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

H. Liu, D. Yu, S. Lu, Y. Zhou, R. Liu, Z. Liang, H. Mi, C.-Y. Wei, and D. Yu. Save the good prefix: Precise error penalization via process-supervised RL to enhance LLM reasoning. *arXiv preprint arXiv:2601.18984*, 2026.

Z. Liu, C. Chen, W. Li, P. Qi, T. Pang, C. Du, W. S. Lee, and M. Lin. Understanding R1-Zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Z. Ma, X. Wen, B. Cao, Y. Lu, H. Lin, J. Yang, M. He, X. Han, and L. Sun. Decoupling reasoning and confidence: Resurrecting calibration in reinforcement learning from verifiable rewards. *arXiv preprint arXiv:2603.09117*, 2026.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, and D. Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

J. Uesato, N. Kushman, R. Kumar, F. Song, N. Siegel, L. Wang, A. Creswell, G. Irving, and I. Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

H. Wang, X. Yu, J. Wu, J. Liu, X. Sun, M. Bansal, and Z. Liu. Stabilizing efficient reasoning with step-level advantage selection. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026a.

Z. Wang, X. Lou, M. Wu, Z. Wen, and J. Zhang. Calibration-aware policy optimization for reasoning LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026b.

C. Xie, R. Pan, X. Wu, Y. Zhang, J. Fu, T. Gao, and G. Zhou. Unlocking exploration in RLVR: Uncertainty-aware advantage shaping for deeper reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

A. Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Q. Yu et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

C. Zheng, Z. Zhang, B. Zhang, R. Lin, K. Lu, B. Yu, D. Liu, J. Zhou, and J. Lin. ProcessBench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*, 2024.

C. Zheng, S. Liu, M. Li, X.-H. Chen, B. Yu, C. Gao, K. Dang, Y. Liu, R. Men, A. Yang, J. Zhou, and J. Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

T. Y. Zhuo, M. C. Vu, J. Chim, H. Hu, W. Yu, R. Widyasari, et al. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations (ICLR)*, 2025.

This appendix gives the update pseudocode and implementation details omitted from the main paper, followed by the full experimental protocol, statistical analysis plan, process-diagnostic construction, ablation definitions, and systems reporting schema. Symbols follow the main paper.

A Complete Update and Implementation

A.1 One policy update

Algorithm S1: One LC-GRPO policy update.

Require: Old policy $\pi_{\theta_{\text{old}}}$, reference policy π_{ref}, verifier $\mathcal{V}$, deterministic parser, group size G, and δ, λ, β

Sample G responses and retain behavior-policy token log-probabilities
Compute $r_i = \mathcal{V}(x, y_i)$ and group-relative $\widehat{A}_i$
for each response i **do**
  if $r_i = 1$ **then**
    Set $A^{\text{LC}}_{i,k} \leftarrow \widehat{A}_i$ for all steps
  else
    Parse y_i into contiguous steps $s_{i,1:K_i}$
    Compute detached $z_{i,k}$ and $c_{i,k} = \exp(z_{i,k})$
    if $K_i > 1$ and $\max_{k \geq 2}(z_{i,k-1} - z_{i,k}) \geq \delta$ **then**
      Set $b_i \leftarrow \arg\max_{k \geq 2}(z_{i,k-1} - z_{i,k})$, breaking ties early
    else
      Set $b_i \leftarrow 1$
    end if
    for each step k **do**
      if $k < b_i$ **then**
        Set $A^{\text{LC}}_{i,k} \leftarrow 0$
      else
        Set $A^{\text{LC}}_{i,k} \leftarrow \widehat{A}_i - \lambda\, \text{sg}(c^{\beta}_{i,k})$
      end if
    end for
  end if
end for
Broadcast each step advantage to its tokens
Apply the response-normalized clipped PPO objective and KL penalty

A.2 Confidence source and detachment

The confidence sensor consumes the same behavior-policy log-probabilities stored for the PPO importance ratio. It does not call the policy a second time. If $\ell_{i,t}^{\mathrm{old}}$ is the log-probability of the sampled token before temperature scaling or nucleus truncation is re-applied, then

$$z_{i,k} = \frac{1}{|\mathcal{I}_{i,k}|} \sum_{t \in \mathcal{I}_{i,k}} \ell_{i,t}^{\mathrm{old}}, \qquad c_{i,k} = \exp(z_{i,k}).$$

Sampling temperature and top-p affect the selected token but do not renormalize this statistic. Both $z_{i,k}$ and $c_{i,k}$ are detached before the boundary and penalty are constructed. The boundary mask is built once from the rollout batch and reused for all PPO epochs on that batch.

Padding, prompt tokens, chat-template insertions, stop tokens, and verifier metadata are excluded. A final nonempty span is retained even when the response has no terminal delimiter. Empty spans are merged with the preceding step. These rules make the confidence tensor independent of padding and batch packing.

A.3 Deterministic parser

The parser is fixed before training and receives response text only. Its precedence order is:

1. close fenced code blocks before applying sentence rules;
2. split at nonempty line boundaries;
3. outside equations and code, split remaining long lines at terminal sentence punctuation;
4. merge empty fragments and fragments containing only delimiters;
5. map every retained token to exactly one contiguous step.

The parser never sees the verifier reward, a reference answer, or a ProcessBench annotation. A parse failure returns one step and therefore the conservative boundary $b_i = 1$.

A.4 Degenerate reward groups

In a mixed group, every incorrect response satisfies $\widehat{A}_i < 0$; LC-GRPO removes its estimated prefix from the update and retains a negative suffix update. In an all-wrong group, $\widehat{A}_i = 0$, so the confidence term supplies a bounded suffix-only signal. In an all-correct group, $\widehat{A}_i = 0$ and the method makes no update. Confidence is never used to create a positive advantage without positive verifier evidence.

B Properties and Proofs

This section proves the propositions stated in Section 6.

Proof of Proposition 1. For $r_i = 1$, the first branch of the advantage definition directly returns $\widehat{A}_i$. For $r_i = 0$, the group-relative advantage satisfies

$$\widehat{A}_i = \frac{-\overline{r}}{\sqrt{G^{-1} \sum_{j=1}^{G} (r_j - \overline{r})^2 + \epsilon_A}} \leq 0. \tag{13}$$

The prefix branch sets $A_{i,k}^{\mathrm{LC}} = 0$ for $k < b_i$. For $k \geq b_i$, the detached confidence correction is $u_{i,k} = \lambda \, \mathrm{sg}(c_{i,k}^{\beta}) \geq 0$. Therefore, $A_{i,k}^{\mathrm{LC}} = \widehat{A}_i - u_{i,k} \leq 0$. □

This proposition follows directly from the advantage definition. The ablation study examines whether enforcing this support and sign structure is useful in training; it does not serve as empirical proof of the algebraic statement.

Proof of Proposition 2. The two advantages are identical outside an incorrect suffix. On an incorrect suffix, their absolute difference is $u_{i,k} = \lambda \operatorname{sg}(c_{i,k}^{\beta})$. Because $c_{i,k} \in (0, 1]$, we have $c_{i,k}^{\beta} \in (0, 1]$ and hence $u_{i,k} \in (0, \lambda]$. This proves Eq. equation 10. □

Proof of Proposition 3. The two score-function contributions contain identical token terms outside $\mathcal{S}_i(b_i) \triangle \mathcal{S}_i(b_i^{\star})$, so these terms cancel. By Proposition 2, the magnitude of every remaining shaped advantage is at most $M + \lambda$. Applying the triangle inequality to the remaining token terms gives Eq. equation 11. □

Equation equation 11 shows that the effect of boundary error grows with the number of incorrectly included or excluded suffix tokens. It applies only to the detached, pre-clipping contribution. It is not a convergence result for PPO and does not imply that the selected boundary is semantically correct.

Proof of Proposition 4. Each adjacent drop contains two confidence values, so its perturbation has magnitude at most 2ξ. The difference between the largest and second-largest drops can therefore decrease by at most 4ξ. The first condition in Eq. equation 12 preserves the identity of the largest drop. Its value can decrease by at most 2ξ, and the second condition ensures that it remains above the detector threshold δ. The selected non-fallback boundary is therefore unchanged. □

B.1 Scope

For binary verification, Eq. equation 7 leaves correct completions unchanged, gives an incorrect prefix zero support, and gives its suffix non-positive support because $\widehat{A}_i \le 0$ and $u_{i,k} \ge 0$. Relative to uniform suffix credit, the confidence correction is bounded by λ. Moreover, if token score functions are bounded by L and $|\widehat{A}_i| \le M$, replacing an oracle boundary by b_i changes the detached, unclipped gradient by at most $L(M + \lambda)|\mathcal{S}_i(b_i) \triangle \mathcal{S}_i(b_i^{\star})|/T_i$. Thus a nearby boundary error affects only the intervening support; this is not a convergence claim for clipped PPO.

The detector is stable to step-log-confidence perturbations of size ξ when the largest drop exceeds the runner-up by 4ξ and exceeds δ by 2ξ. Computing reductions, drops, and masks costs $\mathcal{O}(T_i)$ time and $\mathcal{O}(K_i)$ storage and adds no forward pass. We use one contiguous suffix because later steps may depend on the first invalid transition, but do not claim that every suffix step is semantically wrong. The proofs above establish these properties, and Appendix A gives the update pseudocode and edge cases.

B.2 Objective Normalization and Scope

The implemented objective first averages token losses within each response and then averages over responses. Define $\bar{\rho}_{i,t} = \operatorname{clip}(\rho_{i,t}, 1 - \varepsilon, 1 + \varepsilon)$. The optimization objective is

$$\begin{aligned} \mathcal{J}(\theta) = \frac{1}{G} \sum_{i=1}^{G} \frac{1}{T_i} \sum_{t=1}^{T_i} \min\Big\{ \rho_{i,t} A_{i,t}^{\mathrm{LC}}, \bar{\rho}_{i,t} A_{i,t}^{\mathrm{LC}} \Big\} \\ - \eta D_{\mathrm{KL}}\big(\pi_\theta \| \pi_{\mathrm{ref}}\big). \end{aligned} \tag{14}$$

The factor $1/T_i$ gives each response equal weight regardless of its generated length. Consequently, long responses do not receive greater weight solely because they contain more tokens.

C Reproducibility Protocol

C.1 Shared training configuration

Mathematics and code use separate domain-specific SFT checkpoints. Within a domain and scale, all methods start from the same checkpoint and receive the same prompt batches, decoding configuration, generated-token cap, planned update limit, optimizer schedule, gradient accumulation, precision, clipping, and checkpoint evaluation interval. A run stops when either the planned update limit or token cap is reached. Model selection uses development verifier accuracy only.

Table 4: Shared configuration used in the main comparison.

Item	Setting
Base checkpoints	Qwen3-4B-Base; Qwen3-8B-Base
Training domains	mathematics and code, trained separately
Group size G	8 responses per prompt
Global prompt batch	128
Maximum response length	8192 tokens
Planned updates	500
Optimizer	AdamW
Learning rate	10^{-6}
PPO clip ε	0.2
KL coefficient η	0.01
LC-GRPO parameters	$\delta = 0.10, \lambda = 0.4, \beta = 2$
Evaluation sampling	16 completions, $T = 0.6$, top-$p = 0.95$
Training seeds	3

C.2 Data and immutable identifiers

The anonymous run manifest must record the following fields for every SFT and RL corpus: public source name, release or commit, license, acquisition date, raw count, retained count, domain count, deduplication rule, contamination filter count, and hash of the final prompt list. Development prompts must be disjoint from training and all six test sets. The same final training hashes must be shared by all compared methods within a domain.

Contamination checking first normalizes whitespace, answer markup, and code formatting. It then removes exact prompt or reference-answer matches and reviews candidate near matches returned by 5-gram MinHash. Filtering is completed before the first RL run; no method receives a different filtered corpus.

C.3 Verifiers

The mathematics verifier extracts a boxed answer or an explicitly marked final answer, normalizes integers, fractions, signs, and simple symbolic forms, and attempts symbolic equality. If symbolic comparison is not applicable, it uses normalized exact matching. Extraction failure is incorrect.

Code completions run in a fresh sandbox under the benchmark time and memory limits. A timeout, runtime error, compilation error, or failed hidden test is incorrect. The benchmark release, test harness commit, language runtime, and container digest are fixed across methods and recorded in the manifest.

C.4 Baseline tuning

Every baseline uses its released objective and recommended search interval. Learning rate, KL weight, and method-specific coefficients are selected using the same development prompts and the same trial count. One selected setting is then reused for all three main seeds. Where a step-aware baseline accepts text segments, it receives the same deterministic parser; otherwise its released token-level rule is retained. Test benchmarks never select a hyperparameter or checkpoint.

C.5 System record

For every completed run, the system record contains accelerator model and count, memory per accelerator, host CPU and memory, interconnect, framework versions, precision, parallelism, wall-clock training time, generated tokens, optimizer updates, peak allocated memory, rollout throughput, and update throughput. Section H defines the aggregate report.

Table 5: Three-seed task means used in the main table.

Scale	Method	AIME24	AIME25	MATH-500	AMC	LCB	BCB-H	All Avg.
4B	SFT	41.7	31.5	75.8	54.6	35.1	23.8	43.8
	GRPO	46.0	35.7	78.2	58.4	38.0	26.4	47.1
	Dr. GRPO	46.8	36.4	78.5	58.9	38.4	26.8	47.6
	DAPO	47.5	37.2	78.9	59.6	38.8	27.2	48.2
	SAS	47.9	37.8	79.1	60.0	39.1	27.5	48.6
	UCAS	48.4	38.1	79.2	60.2	39.3	27.8	48.8
	CAPO	48.7	38.5	79.4	60.5	39.5	28.0	49.1
	LC-GRPO	**51.0**	**41.2**	**80.6**	**63.1**	**41.6**	**30.0**	**51.3**
8B	SFT	56.3	45.0	84.2	67.8	45.5	34.0	55.5
	GRPO	60.7	49.2	86.0	71.2	48.3	37.0	58.7
	Dr. GRPO	61.4	49.8	86.2	71.6	48.7	37.4	59.2
	DAPO	62.0	50.7	86.5	72.1	49.2	37.9	59.7
	SAS	62.3	51.0	86.6	72.4	49.4	38.1	60.0
	UCAS	62.6	51.3	86.7	72.7	49.6	38.5	60.2
	CAPO	62.9	51.6	86.8	73.0	49.8	38.7	60.5
	LC-GRPO	**65.1**	**54.4**	**88.0**	**75.3**	**52.1**	**41.2**	**62.7**

D Statistical Analysis and Run-Level Reporting

D.1 Estimands

For a task d, method m, and seed s, the primary score is the mean problem-level verifier success across the fixed evaluation set. A problem score is first averaged over its 16 completions. The reported task mean is the arithmetic mean over three seeds. All Avg. is then the unweighted mean of the six task means, preventing larger benchmarks from dominating the summary.

The two pre-specified paired comparisons are LC-GRPO versus GRPO and LC-GRPO versus CAPO. Their task-level effect is $\Delta_{d,s} = P^{\mathrm{LC}}_{d,s} - P^{\mathrm{base}}_{d,s}$. Pairing uses the same prompts and seed index.

D.2 Confidence intervals and tests

The 95% interval uses a hierarchical paired bootstrap. Each replicate resamples the three seed blocks with replacement and then resamples evaluation problems within each selected block. Method labels remain paired. The interval is the 2.5th and 97.5th percentiles of 10,000 replicates.

The paired randomization test swaps the two method labels within paired seed–problem blocks, recomputes the mean difference, and uses a two-sided tail. Holm correction is applied once to the 12 pre-specified hypotheses: six tasks times two comparators. The supplement reports raw and adjusted p-values together with effect size and interval. Significance is not used as a substitute for effect size.

AIME has few problems, so the problem-level observation is its success rate over 16 completions rather than one binary completion. The hierarchy retains both seed variation and problem sampling. All randomization and bootstrap seeds are recorded separately from training seeds.

D.3 Aggregate task means

Table 5 reproduces the six task means used by the main paper. It is included to make arithmetic checks possible independently of the PDF layout.

E Matched-Compute VPPO Protocol

VPPO is the closest process-supervised comparison because it uses an external process reward model (PRM) to protect a verified prefix. A token-only budget does not charge its additional model calls,

while accelerator-hours alone can hide different rollout counts. We therefore define two separate controls.

Matched generated tokens. Both methods receive the same prompts, response limit, number of sampled training tokens, optimizer-update ceiling, SFT start, and evaluation protocol. VPPO additionally invokes its released PRM. This comparison isolates accuracy at the same policy-generated data volume.

Matched accelerator-hours. The budget includes policy rollout, reference-policy evaluation, PRM forward passes, policy updates, and synchronization. Runs stop when total accelerator time reaches the budget used by LC-GRPO. This comparison measures accuracy under total training compute.

Required report. Each condition records task scores, generated tokens, PRM-scored tokens, policy updates, accelerator-hours, throughput, and peak memory. Hyperparameters are selected with the same development trial budget. The PRM name, checkpoint, input granularity, and inference batch size are fixed and reported.

F Diagnostic Protocol

F.1 Calibration and overconfidence

For one completion from each prompt–temperature pair, define $q_i = \min_k c_{i,k}$. The verifier label is not used to construct q_i. Fifteen equal-mass bins are formed from the pooled evaluation index and then held fixed across methods. We report ECE, Brier score, correctness AUROC, and OC-Err, the percentage of incorrect responses with $q_i \geq 0.8$.

Because a minimum changes with the number of parsed steps, q_i is not presented as a calibrated probability of correctness. The machine-readable diagnostic report also stratifies ECE and AUROC by response-length and step-count quantiles so that a change in length distribution is not mistaken for calibration.

F.2 ProcessBench filtering and alignment

Boundary evaluation includes ProcessBench derivations that satisfy both: (i) an annotated first error exists, and (ii) the same final-answer verifier used by training judges the response incorrect. A final-answer-correct derivation with an annotated process error is excluded because the training-time gate would not activate. This exclusion is reported rather than silently scored as a localization failure.

Every checkpoint is evaluated by teacher forcing the same derivations. The same parser and maximum-drop detector are applied to every method, even when that method does not natively output a boundary. Thus rows compare representations induced by training, not different localization algorithms. The threshold δ is fixed before ProcessBench evaluation and is never tuned on annotated boundaries.

The annotated error span is mapped to the first parsed step that overlaps it. Exact is one when b_i equals that step; within-one is one when their distance is at most one. Both are reported because punctuation and line breaks can put a parser boundary next to the annotated span.

F.3 Frozen linear probe

The $60/20/20$ split is used only for the probe and is grouped by problem to prevent trajectory leakage. Step states are mean-pooled from three fixed, spaced layers. Layer choice and L_2 strength are selected on the development split; F1 is computed once on the held-out split. The encoder is frozen and no probe gradient reaches the policy. The probe tests whether first-error information is represented, not whether the proposed log-probability detector uses hidden states.

Table 6: Diagnostic values reported in the main paper.

Method	ECE↓	AUROC↑	Exact↑	±1↑
GRPO	.172	.689	.421	.619
Dr. GRPO	.165	.701	.428	.626
SAS	.148	.728	.612	.768
UCAS	.126	.758	.684	.824
CAPO	.104	.792	.652	.801
DCPO	.096	.806	.438	.631
LC-GRPO	**.083**	**.837**	**.781**	**.914**

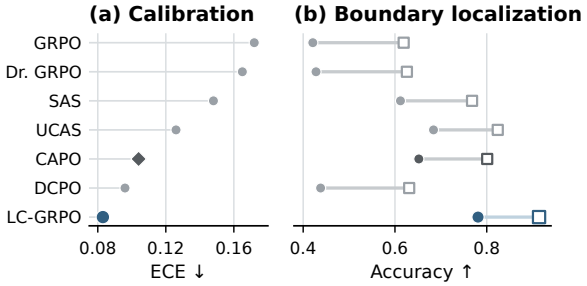

Figure 3: **Calibration and first-error localization.** The left panel reports ECE. The right panel reports exact and within-one-step ProcessBench localization under the shared detector.

G Expanded Results and Ablations

G.1 Definitions of all variants

- **Minimum-confidence boundary:** replace the maximum adjacent drop with $b_i = \arg\min_k c_{i,k}$.
- **Token-level maximum drop:** compute adjacent changes at the token level and map the selected token back to its parsed step.
- **Random boundary:** sample b_i uniformly from valid step indices using the run seed.
- **No drop threshold:** always take the maximum drop, even when it is smaller than δ.
- **Sentence-only parser:** ignore newline precedence and segment only at sentence punctuation outside code.
- **No prefix protection:** apply the failed-suffix formula to every step while retaining the same estimated b_i for diagnostics.
- **Whole-response confidence penalty:** use one response-level confidence weight on every step of an incorrect response.
- **Uniform suffix penalty:** replace $c_{i,k}^{\beta}$ with one for $k \geq b_i$.
- **Unconstrained confidence bonus:** change only the suffix rule to $A_{i,k}^{\text{bonus}} = \widehat{A}_i + u_{i,k}$. Boundary, prefix mask, rollout batch, and seed are unchanged. This can create positive credit on a failed suffix; it is not deletion of a redundant sign clamp.
- **Linear penalty:** set $\beta = 1$.
- **No all-wrong suffix signal:** set every shaped advantage to zero when the complete group has reward zero.

Minimum confidence loses both localization and task accuracy because a later uncertain recovery can be selected instead of the first damaging transition. The random-boundary result shows that the gain is not explained by simply masking an arbitrary prefix. No prefix protection changes Exact little but loses 2.3 points, separating the sensor from the optimization rule that uses it. Uniform suffix credit loses 1.2 points, so the confidence magnitude adds signal beyond localization alone. The unconstrained bonus is the worst support variant because it may reward a verifier-rejected suffix.

Table 7: Full grouped ablation on Qwen3-8B-Base.

Variant	Avg.↑	Δ	ECE↓	Exact↑
LC-GRPO (full)	**62.7**	–	**.083**	**.781**
Failure-boundary estimation				
Minimum-confidence boundary	60.7	-2.0	.108	.614
Token-level maximum drop	61.3	-1.4	.101	.699
Random boundary	59.3	-3.4	.139	.223
No drop threshold	61.6	-1.1	.094	.724
Sentence-only parser	61.8	-0.9	.092	.738
Advantage support and confidence direction				
No prefix protection	60.4	-2.3	.121	.779
Whole-response confidence penalty	59.8	-2.9	.129	.780
Uniform suffix penalty	61.5	-1.2	.111	.777
Unconstrained confidence bonus	59.2	-3.5	.147	.775
Linear penalty ($\beta = 1$)	62.0	-0.7	.096	.780
No all-wrong suffix signal	60.9	-1.8	.117	.779

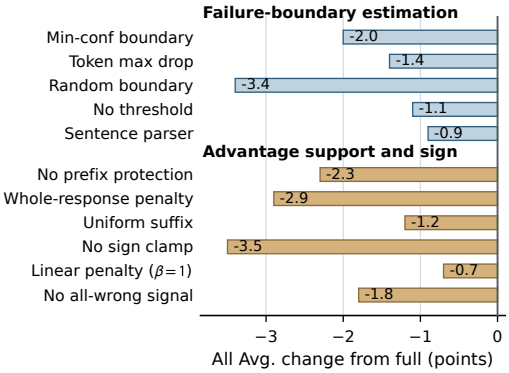

Figure 4: **Ablation impact.** All Avg. change from the complete LC-GRPO configuration.

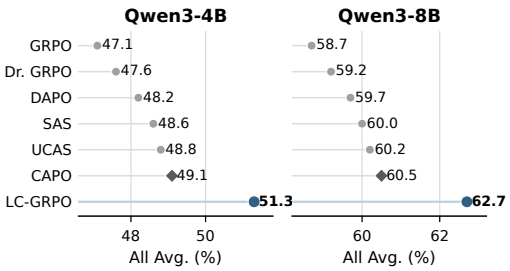

Figure 5: **Six-task average under matched generated tokens.** Values reproduce the All Avg. column of the main table.

H Systems Cost and Run Manifest

The confidence reductions and parser execute on the CPU and reuse rollout log-probabilities. Relative to GRPO in the shared trainer, LC-GRPO retains 98.6% of rollout-plus-update throughput and increases peak memory by 0.7%. It adds no inference module and therefore has the same inference architecture as GRPO.

Relative percentages do not replace absolute reporting. The final systems table is computed from the same three main runs and records the mean and standard deviation of wall-clock time, generated tokens, optimizer updates, tokens per second, peak memory, and accelerator-hours. Accelerator-hours are the number of active accelerators multiplied by elapsed training hours and include rollout and update workers. VPPO additionally includes its PRM workers.

The manifest also records software versions, precision, parallelism, host configuration, container or environment hash, checkpoint hashes, dataset hashes, evaluation commands, and all random seeds. These fields prevent a percentage-only efficiency claim from hiding a different hardware or parallelism configuration.