BEYOND SPURIOUS REFLECTION: STRATEGIC SELF-CORRECTION AS STATE REPLACEMENT IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models excel by scaling inference-time Chain-of-Thought, yet their performance is brittle: an early mistake corrupts all subsequent reasoning. Reinforcement-learning-trained reasoners often produce reflective phrases like "wait, let me reconsider," but growing evidence shows these reflections are largely spurious—they neither alter reasoning paths nor final answers. Existing self-reflection methods keep the faulty chain in context, forcing the model to overcome its own errors. We propose REFLECT+, an end-to-end framework that directly optimizes reflection-triggered self-correction. Building on the Clean-Slate Reflection paradigm—where the model emits a structured `<reflection>` block summarizing verified sub-conclusions, diagnosed mistakes, and corrected directions, then continues from the problem and reflection alone—REFLECT+ adopts a two-stage training scheme combining supervised cold-start and role-aware reinforcement learning. This enables strategic decisions on when to reflect, what to retain, and how to resume. Experiments on Qwen2.5-Math-1.5B / Qwen3-0.6B-Base / Qwen3-4B-Base across MATH500, AMC23, GSM8K, LogiQA, GPQA, MMLU-Pro, BBH, CLUTRR and DROP evaluate both self-correction quality and reasoning accuracy, and outperforms GRPO on accuracy with selective reflection.

1 INTRODUCTION

Large reasoning models have demonstrated remarkable performance on complex tasks such as mathematical reasoning, logical reasoning, and code reasoning (Shao et al., 2024; Yang et al., 2024). These gains primarily stem from *inference-time scaling*: generating long chains-of-thought that enable problem decomposition, trajectory planning, multi-step deduction, and self-reflection (Bachmann et al., 2026; Wang et al., 2023; Wei et al., 2022). Among these capabilities, *self-correction* has emerged as a key paradigm: the model is allowed to revise its own reasoning before producing the final answer (Lee et al., 2026; Kamoi et al., 2024; Chen et al., 2023). A typical self-correction process involves reflecting on a suspected mistake and then retrying—often by appending a reflection and a new reasoning attempt to the existing context.

However, current self-correction mechanisms suffer from two critical defects. **First, *spurious reflection*.** RL-trained reasoners frequently emit reflective phrases such as "let me reconsider", yet these neither change the reasoning direction nor affect the final answer (Kirchhof et al., 2026; Yeo et al., 2025; Chen et al., 2025; Turpin et al., 2023). In many cases, the reflection acts more like commentary on top of an already-corrupted derivation than a genuine redirection of reasoning (Dang et al., 2026; Yu et al., 2026). **Second, *useless retry*.** Even when a reflection is semantically meaningful, existing reflect-then-retry pipelines keep the faulty chain in the continuation prompt (Shinn et al., 2023; Madaan et al., 2023; Gou et al., 2024), forcing the model to learn to ignore its own misleading earlier output, which limits the correction quality. One important limitation of current RL paradigms is the lack of explicit training signals for reflection quality. Current RL paradigms treat reflection as another token sequence, providing no structured feedback on *whether* a reflection is necessary, *what* it should contain, or *how* to continue afterwards. Consequently, current models learn to mimic reflective formats without gaining genuine control over the entire correction process.

To address the above challenges, we propose **REFLECT+**, an end-to-end framework that directly optimises reflection-triggered self-correction. Building on the Clean-Slate Reflection paradigm (Figure 1)—where the model detects its own errors and writes a structured summary to redirect its

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.

reasoning. Our approach proceeds in two stages: **Stage 1: Supervised cold-start.** We first teach the model to produce structured reflection blocks (a summary of the mistake and a redirection plan) and to solve problems conditioned on such reflections. **Stage 2: Role-aware RL.** We then optimise the strategic decisions—*when* to reflect, *what* to summarise, and *how* to continue—via explicit reward signals that distinguish format compliance from correction effectiveness. This employs a credit assignment mechanism tailored to the multi-role generation structure, where the model alternates between direct solving and reflection. It normalizes rewards across different events and token positions, ensuring that both format compliance and correction effectiveness are properly rewarded.

Empirically, on Qwen2.5-Math-1.5B, REFLECT+ improves the unweighted average accuracy over vanilla GRPO from 30.69% to 36.12% across nine benchmarks, and increases the macro-average pass@k from 75.42% to 79.11% across the eight benchmarks excluding MMLU-Pro, using each benchmark's evaluation budget. On Qwen3-0.6B-Base and Qwen3-4B-Base, REFLECT+ improves pass@32, demonstrating a coverage benefit at both scales, as the 4B gains extend to all six benchmarks (+3.25 points in overall pass@32). Further analyses show that learned policy reflects selectively, and adaptive reflection timing significantly outperforms forced interruption strategies. Clean-slate continuation conditioned only on the reflection r consistently surpasses both error-prefix-only (x, y^-) and appended-reflection (x, y^-, r) continuations.

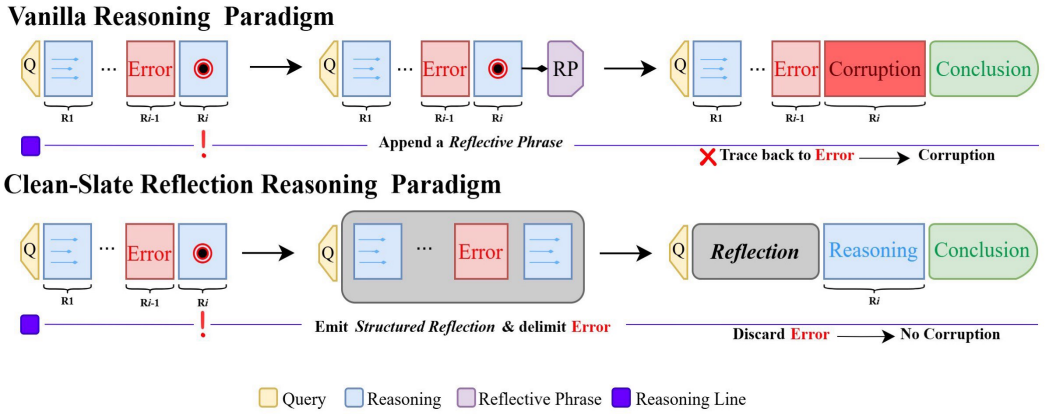

Figure 1: Clean-Slate Reflection reasoning paradigm vs. Vanilla reasoning paradigm. Upper panel: Vanilla can not trace back to the source of error when encounters a mistake mid-trace, it only appends a reflective phrase. Lower panel: Clean-Slate Reflection treats reflection as a state replacement to decouple self-correction from the context.

2 RELATED WORK

Work on self-correction found that language models often fail to revise their answers without external feedback (Lee et al., 2026; Huang et al., 2024; Pan et al., 2024), and that CoT explanations can be unfaithful to the model's internal computation (Turpin et al., 2023; Lanham et al., 2023). More recent studies on RL-trained reasoners report that long CoTs are filled with repetitive and shallow reflections that do not alter the final answer (Zhang et al., 2026; Yeo et al., 2025; Chen et al., 2025). These findings suggest that reflection must be not only imitative but also strategically effective.

To address this, existing methods have evolved along two complementary directions. The first imposes constraints on when and how reflection occurs. CyclicReflex (Fan et al., 2026) dynamically modulates reflection token logits to regulate the frequency and placement. Reflexion (Shinn et al., 2023), Self-Refine (Madaan et al., 2023), and CRITIC (Gou et al., 2024) insert explicit critique or verification loops, forcing the model to pause and evaluate its own output. Self-Debug (Chen et al., 2023) apply similar ideas to mathematical and code reasoning. While these approaches improve correction rates, they typically rely on fixed schedules or external verifiers. More importantly, they always append the reflection to the original faulty chain. The model must therefore learn to ignore its own prior mistakes—which becomes increasingly burdensome as the reasoning chain length grows.

The second direction treats self-correction as a policy optimisation problem. Recent work (Sun, 2026) has explored learning reflective behaviors from erroneous reasoning trajectories via SFT and GRPO. REA-RL (Deng et al., 2026) uses an online distilled reflection model to identify the point after which a correct trajectory begins to overthink and further introduces a reflection-density reward. However, the reward signals rely on surface-level cues (e.g., keyword density) and do not explicitly distinguish meaningful reflection from spurious ones (Turpin et al., 2023).

In contrast, the REFLECT+ framework introduces three key innovations. First, unlike prior work that appends reflections to corrupted chains, we adopt the Clean-Slate Reflection paradigm, which discards the faulty prefix context and treats the reflection as a compact state replacement. Second, we optimize *when* to reflect, *what* to summarize, and *how* to continue via role-aware RL with explicit, format-gated rewards. Third, our training separates format acquisition from strategic usage, ensuring the model learns not just to produce reflections but to deploy them adaptively. Together, these advances transform model's self-correction from post-hoc commentary to genuine state replacement.

3 METHODS

In this section, we present the training process for REFLECT+ (see Figure 1). We first introduce the Clean-Slate Reflection reasoning paradigm as the foundation of our approach(Section 3.1). We then describe the two-phase cold-start procedure, which enables the model to acquire the reflection format and the reflection-based solving skill (Section 3.2). Finally, we provide a detailed description of the RL strategy that optimises the complete reflection-triggered trajectory end-to-end (Section 3.3). Implementation details of the cold-start procedure and RL strategy are demonstrated in Figure 5.

3.1 CLEAN-SLATE REFLECTION REASONING

We first contrast the vanilla reasoning paradigm with Clean-Slate Reflection paradigm to clarify the structural differences that motivate our approach (Figure 1).

Vanilla reasoning paradigm. Most existing reasoning models, following the format represented by DeepSeek-R1, give outputs consisting of a long reasoning trace and a concise conclusion. When the model encounters a mistake mid-trace, it can only *append* a reflective phrase; the faulty derivation remains in the context and continues to bias all subsequent tokens.

Clean-Slate Reflection reasoning paradigm. Clean-Slate Reflection decouples self-correction from context preservation by treating the reflection as *state replacement* rather than appended commentary. For a query x, the model produces a partial reasoning chain y^-. When it detects that y^- is wrong, it emits a structured reflection r:

$$r \sim p_\theta\big(r \mid x, y^-\big), \tag{1}$$

where r is delimited by `<reflection> ...</reflection>`, which contains three explicit fields: a verified sub-conclusions before the error, a one-sentence diagnosis of the mistake, and a one-sentence corrected direction. The continuation is sampled from the following *rebuilt* prompt:

$$y^+ \sim p_\theta\big(y^+ \mid x, r\big). \tag{2}$$

The faulty prefix y^- is not part of the continuation context; the reflection is the only carry-over. Nested reflections follow the same rule: if y^+ contains another reflection r', the next continuation is sampled from (x, r'). We cap the number of reflection events per trajectory at two.

The key distinction from vanilla reasoning is that each post-reflection iteration operates in the *purified* context: the model sees only the original query and the most recent reflection, not the faulty derivation that triggered it. This design is intended to achieve two goals simultaneously: it bounds the influence of any local mistake to the prefix before reflection, and it forces the model to distill essential progress into a reflection that supports subsequent reasoning.

3.2 COLD START

Before applying RL, we perform a cold-start stage that teaches the model the basic format of Clean-Slate Reflection-style reasoning. We transform existing supervised data to reflection format, and fine-tune the model in two phases, targeting reflection generation and reflection-conditioned solving.

Data transformation. We construct cold-start data from the hard split of Big-Math (Albalak et al., 2025). For each problem, Qwen2.5-Math-1.5B / Qwen3-0.6B-Base / Qwen3-4B-Base first samples a reasoning trajectory, which is then verified for correctness. Incorrect trajectories are transformed by a strong API teacher (e.g., Deepseek v4 Pro) into Clean-Slate Reflection training instances containing an error prefix, a structured reflection, and a corrected continuation, while correct minimal solutions are retained as direct-solving examples. This converts a vanilla pair (x, y) into:

$$(x, y) \xrightarrow{\text{teacher}} \begin{cases} (x, y^-, r, y^+) & \text{if reflective,} \\ (x, y) & \text{if perfect.} \end{cases} \tag{3}$$

We mix reflective and perfect samples during SFT; otherwise the model collapses to reflect on almost every problem. Full construction details, teacher schema, and training data are in Appendix C.

Two-phase supervised initialization. We treat `<reflection>` and `</reflection>` as plain text markers (no embedding resize) and fine-tune Qwen2.5-Math-1.5B / Qwen3-0.6B-Base / Qwen3-4B-Base with a LoRA (Hu et al., 2021) adapter (hyper-parameters are in Appendix C).

In Phase 1 (error-prefix $\rightarrow$ reflection), the supervision target is r; the system prompt, problem, and faulty prefix y^- are masked from the loss shown below. On perfect instances, the entire solution is the supervision target with no y^-; the model is supervised *not* to emit a reflection.

$$\mathcal{L}_{\text{p1}} = -\sum_{t=1}^{|r|} \log p_\theta\big(r_t \mid x, y^-, r_{<t}\big). \tag{4}$$

In Phase 2 (reflection $\rightarrow$ solution), Phase 1's adapter continues training without being merged into the base, so the same low-rank subspace keeps being refined. The supervision target is y^+ with r included in the prompt but masked from the loss:

$$\mathcal{L}_{\text{p2}} = -\sum_{t=1}^{|y^+|} \log p_\theta\big(y_t^+ \mid x, r, y_{<t}^+\big). \tag{5}$$

Crucially, y^- no longer appears in the context: Phase 2 trains the model to solve from the reflection alone, matching the form required at RL inference. To prevent catastrophic forgetting, Phase 2 mixes three example types—reflection-to-solution (main objective), Phase-1 replay (preserves reflection generation), and perfect samples (preserves direct solving)—with the mixing ratio reported in Appendix C. After this stage, the model can produce syntactically valid reflection blocks and solve from a reflection alone, but it has not learned *when* a reflection should fire, *what* sub-conclusions to carry on, or *how* to adapt depth to difficulty; these all require role-level optimization in the RL stage.

3.3 Reinforcement Learning

The cold-start stage teaches the model the reflection format; the RL stage then teaches its strategic use. We adapt GRPO (Shao et al., 2024) to the structure of reflection-triggered reasoning, in which a single problem yields a trajectory that may include one or more clean-slate reflection events.

Rollout. For each problem x, we draw G rollouts with a vLLM rollout engine that implements the Clean-Slate Reflection prompt rebuild: whenever a complete `<reflection>...</reflection>` block is emitted, the engine discards the pre-reflection text from the continuation prompt and re-seeds generation from (x, r) before resuming sampling. Nested reflections follow the same rule, up to a cap of two reflection events per trajectory (i.e. at most three generation rounds: an initial chain, optionally a first reflection-triggered continuation, and optionally a second). A rollout terminates when (i) the model gives a final answer, (ii) reflection depth reaches the cap, or (iii) the maximum response length is reached. Each rollout is either a *direct* trajectory (no reflection, denoted as o^{dir}) or a *reflection-triggered* trajectory (denoted as o^{ref}).

Reward. We assign a single scalar reward to each rollout. A direct rollout that produces a correct answer receives the binary task reward; a direct rollout that produces a wrong answer receives a small wrong-no-reflect penalty, which encourages the model to attempt reflection rather than silently fail:

$$R^{\text{dir}} = \begin{cases} 1 & \text{correct,} \\ -\big((n_w - 1)/\phi\big)^2 & \text{wrong,} \end{cases} \tag{6}$$

with n_w=3, ϕ=4. A reflection-triggered rollout combines a correctness term, an efficiency factor (penalizing over-reflection), and a multiplicative format gate:

$$R^{\text{ref}} \;=\; \mathbf{1}[\hat{a}{=}a^{\star}] \;\cdot\; R_{\text{eff}} \;\cdot\; g_{\text{fmt}}, \tag{7}$$

where $R_{\text{eff}} \;=\; 1 \;-\; \big((d \;-\; d_{\text{free}})_{+}/\phi\big)^{2}$ with depth d (the number of reflection events in the trajectory), free budget d_{free}=1, and ϕ=4. The first reflection is cost-free; only deeper reflections incur a smooth quadratic penalty. The format gate $g_{\text{fmt}} \in \{0, 1\}$ requires every <reflection>...</reflection> block to contain the cues "so far I have" and "however"; when any block is malformed, g_{fmt}=0 zeroes the entire reward, preventing the policy from gaming the binary task signal with reflection-shaped but structurally invalid outputs. Importantly, the format gate only enforces a minimal structural constraint; semantic usefulness is determined by downstream task success and policy advantage.

Counterfactual-baseline advantage. For each prompt x we partition its G rollouts into a direct sub-group and a reflection sub-group and compute advantages relative to the *per-prompt counterfactual baseline*: the mean reward μ_x^{dir} of the direct sub-group. Both sub-groups are standardized against the full-group standard deviation σ_x, applied to direct and reflection samples alike:

$$A_i \;=\; \frac{R_i - \mu_x^{\text{dir}}}{\sigma_x + \varepsilon}, \tag{8}$$

The intuition is that reflection should be judged against the same problem's direct rollouts rather than against an indiscriminate full-group mean: a reflection rollout that succeeds where direct attempts fail receives strong positive advantage, while one that fails on a problem the direct rollouts mostly solve is discouraged. When a prompt has no direct rollout, we fall back to the full-group mean.

Difficulty-gated advantage floor. Equation 8 alone causes reflection to collapse on easy prompts: when most direct rollouts succeed ($\mu_x^{\text{dir}}{\approx}1$), a reflection rollout is strongly negatively advantaged—whether it fails outright or succeeds but pays the efficiency factor in Eq. 7—even when the reflection itself was reasonable. The policy then learns to suppress reflection altogether and loses the ability to reflect on the few hard prompts that genuinely need it. We address this with a *difficulty-gated* asymmetric floor applied only to reflection samples:

$$\tilde{A}_i^{\text{ref}} \;=\; w_x \cdot \max\!\big(A_i^{\text{ref}},\, A_{\text{floor}}\big) \;+\; (1 - w_x) \cdot A_i^{\text{ref}}, \tag{9}$$

where $w_x \;=\; \text{clip}(\mu_x^{\text{dir}}, 0, 1)$ is the per-prompt difficulty gate and $A_{\text{floor}}{=}{-}0.5$. On easy prompts ($w_x \to 1$) the floor is fully active and damps the negative gradient on unsuccessful reflections, preserving the policy's freedom to explore reflection without being crushed for it. On hard prompts ($w_x \to 0$) the floor is disabled and the full reflection gradient flows—exactly where reflection is the only chance of recovery. The mechanism is asymmetric in two senses: it leaves positive advantages untouched, and the gating scales with prompts' difficulty, rather than with a global schedule.

Policy gradient. Tokens are optimised with the asymmetric PPO-clip surrogate (Schulman et al., 2017), and clip range details are in Appendix D). $\rho_t \;=\; \pi_\theta(o_t)/\pi_{\theta_{\text{old}}}(o_t)$ and $c_\epsilon(\rho_t) \;=\; \text{clip}(\rho_t,\, 1{-}\epsilon_{\text{lo}},\, 1{+}\epsilon_{\text{hi}})$, the objective is:

$$\mathcal{J}(\theta) = \mathbb{E}_t\Big[\min\!\big(\rho_t \hat{A}_t,\; c_\epsilon(\rho_t)\, \hat{A}_t\big)\Big], \tag{10}$$

with the scalar advantage from Eq. 8 / 9 broadcast to every response token including the pre-reflection prefix; this gradient on pre-reflection tokens is what teaches the model *when* to pivot to <reflection> rather than continue with another erroneous step.

Training stability. When all G rollouts of a prompt share the same outcome, the group standard deviation may become extremely small, and dividing by σ_x in Eq. 8 can amplify small stochastic reward differences into disproportionately large normalized advantages. We skip standardisation for such uniform-outcome groups (optimiser stack and other training-time safeguards in Appendix D).

4 EXPERIMENTS

Training. We conduct experiments on Qwen2.5-Math-1.5B / Qwen3-0.6B-Base / Qwen3-4B-Base (Yang et al., 2024). Cold start implements the reflection-augmented Big-Math corpus (Section 3.2) and a quota-aware pool of API teachers (Deepseek-V4-Pro), with LoRA (Hu et al., 2021) SFT (Appendix C). For RL, we train on a math-focused split using `verl` (Sheng et al., 2024). Hyper-parameters, the task verifier, and the format gate regular expression are shown in Appendix B and Appendix D.

Evaluation. We evaluate all four configurations on the nine benchmarks reported in Table 1: the math benchmarks MATH500, AMC23, and GSM8K, together with LogiQA (logical reasoning), GPQA (graduate-level science), MMLU-Pro (broad knowledge), BBH (challenging reasoning), CLUTRR (compositional relational reasoning), and DROP (discrete reasoning over paragraphs). For MATH500, AMC23, GSM8K, LogiQA, and GPQA, details are shown in Appendix E. Section 4.1 summarizes the cross-model effects on Qwen3-0.6B and Qwen3-4B, while Appendix K reports the complete per-dataset pass@32 results together with all evaluation configurations.

Extended experiments and analyses. We provide additional experiments and analyses in the appendix: (i) the failure-mode taxonomy that motivates our design (Appendix A); (ii) the format-gate regular expression and reward hyper-parameters (Appendix B); (iii) per-step reflection trigger rate, depth distribution, and recovery/regression statistics over training (Appendix F); (iv) a reflection-template example (Appendix G); (v) representative continuation case studies (Appendix H); and (vi) a recovery analysis on hard reflection points (Appendix J).

4.1 MAIN RESULTS

Table 1 reports accuracy on nine benchmarks and pass@k on eight benchmarks for the pretrained Qwen2.5-Math-1.5B backbone, the cold-start SFT model, a vanilla GRPO baseline, and REFLECT+. The evaluation budgets are specified in the table caption. The Overall column gives unweighted averages across nine benchmarks for accuracy and eight for pass@k, excluding MMLU-Pro from the latter. Except for MMLU-Pro, results are averaged over five random seeds.

Model	Metric	MATH500	AMC23	GSM8K	LogiQA	GPQA	MMLU-Pro	BBH	DROP	CLUTRR	Overall
Base	Mean	20.24	17.06	18.31	9.37	4.73	5.39	5.67	3.68	0.19	9.40
	Pass	87.40	81.00	92.90	90.41	72.63	–	34.05	22.95	1.50	60.36
SFT	Mean	52.01	26.80	61.19	28.44	**20.86**	20.20	21.01	24.87	3.06	28.72
	Pass	88.48	77.00	97.24	**98.83**	**97.58**	–	61.35	64.76	19.44	75.59
GRPO	Mean	62.07	**43.36**	71.66	22.35	10.73	13.76	22.39	27.27	2.58	30.69
	Pass	**91.52**	**89.00**	**97.85**	97.11	83.54	–	59.35	67.36	17.64	75.42
REFLECT+	Mean	**66.43**	39.52	**80.07**	**29.31**	18.77	**22.54**	**28.76**	**35.70**	**3.99**	**36.12**
	Pass	89.60	88.00	96.69	98.61	94.85	–	**68.70**	**72.36**	**24.08**	**79.11**

Table 1: Main results on nine benchmarks (%). Mean and Pass denote mean accuracy and pass@k, respectively. MATH500, AMC23, GSM8K, LogiQA, and GPQA use 32 samples per problem with temperature 1.0 and report Mean@32/Pass@32; BBH, DROP, and CLUTRR report Mean@8/Pass@8. MMLU-Pro uses greedy decoding: its accuracy is reported in the Mean row, and the Pass entry is not applicable. *Overall* is the unweighted macro-average across all nine benchmarks for Mean and across the eight benchmarks excluding MMLU-Pro for Pass, using each benchmark's stated k. **Base**: pretrained Qwen2.5-Math-1.5B; **SFT**: cold-start model; **GRPO**: vanilla GRPO baseline; **REFLECT+**: difficulty-gated reflection RL. Best results within each metric are in **bold**.

REFLECT+ improves overall reasoning performance on Qwen2.5-Math-1.5B. REFLECT+ improves accuracy over vanilla GRPO on eight of the nine benchmarks, including GSM8K (80.1 vs. 71.7), LogiQA (29.3 vs. 22.4), MMLU-Pro (22.5 vs. 13.8), and DROP (35.7 vs. 27.3). AMC23 is the only exception (39.5 vs. 43.4). Across all nine benchmarks, the unweighted average accuracy increases from 30.69% to 36.12%. Across the eight benchmarks excluding MMLU-Pro, the macro-average pass@k increases from 75.42% to 79.11%, with $k = 32$ for MATH500, AMC23, GSM8K, LogiQA, and GPQA, and $k = 8$ for BBH, DROP, and CLUTRR.

REFLECT+ retains direct solving on easy problems. Beyond accuracy, REFLECT+ reshapes the trade-off between direct self-correction and decoding cost. As cold-start mixes perfect samples with reflection samples and the RL format gate denies credit to malformed <reflection></reflection> spans, the trained model preserves direct solving on easier problems while concentrating reflections on harder cases. The reflection diagnostics in Table 3 support this: the trigger rate is 12.2% on GSM8K, 18.3% on MATH500, and 27.5% on AMC23 rather than collapsing into reflection on every example, in contrast to vanilla long-CoT models where "reflective" phrases often appear without changing the answer (Yeo et al., 2025; Chen et al., 2025).

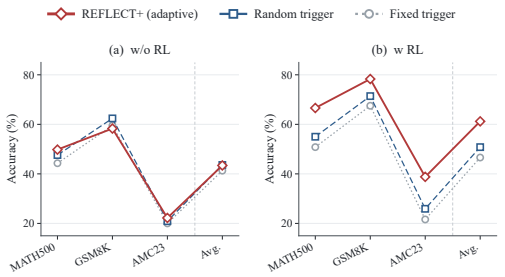

Figure 2: Reflection-timing ablation over training. We compare REFLECT+'s adaptive trigger against a uniformly random interrupt ($B \sim U[128, 768]$) and a fixed interrupt ($B = 256$), both with and without RL. Mean@8 on MATH500 / GSM8K / AMC23. Sampling: n=8, temperature=1.0.

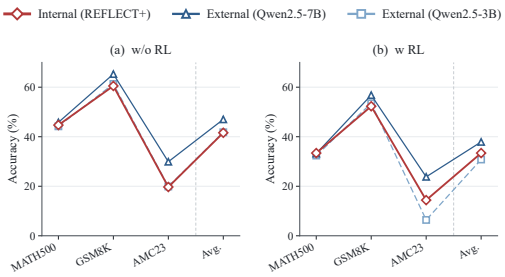

Figure 3: Reflection-source ablation over training. At inference stage we replace the model-generated reflection r with one produced by an external teacher (Qwen2.5-3B / 7B-Instruct) and continue from the same rebuilt prompt (x, r), holding the continuer fixed. Reported as mean@8 on MATH500 / GSM8K / AMC23. Sampling: n=8, temperature=1.0.

Cross-model evaluation on Qwen3-0.6B and Qwen3-4B reveals a solution-coverage benefit. Across backbones, REFLECT+ improves pass@32 over vanilla GRPO on BBH, DROP, and GSM8K. The gains are especially prominent on DROP (+7.7 points at 0.6B and +1.7 at 4B), while GSM8K improves by 1.1 points at both scales. On Qwen3-4B, this coverage advantage extends to all of the benchmarks: the benchmark pass@32 increases by 3.3 points. The largest 4B gains occur on GPQA (+10.1), indicating that reflection is particularly effective on challenging scientific, mathematical, and knowledge-intensive tasks. The reliability benefit is capacity- and task-dependent, and all per-dataset results are in Appendix K.

4.2 REFLECT+ ENHANCES SELF-CORRECTION

Through end-to-end optimisation, REFLECT+ enables reasoning models to acquire *effective* and *efficient* self-correction capabilities. Effective self-correction is challenged by three key questions, each tied to an earlier limitation. *When to reflect* determines the appropriate timing for switching from direct solving to reflection-triggered correction, influencing the trade-off between exploration and exploitation—and directly addressing the problem of *spurious reflection*. *What to reflect* defines the mechanism by which essential reasoning states are distilled and propagated to the rebuilt continuation—addressing the *lack of explicit signals for reflection quality*. *How to continue* specifies how the model conditions future reasoning steps on the compressed reflection while ignoring the discarded prefix—addressing the *useless retry* problem caused by retaining the faulty chain. We further examine whether the resulting policy remains robust against spurious reflection.

When to reflect. To analyse the practical effect of REFLECT+ for models following the reflection-triggered paradigm, we introduce two alternative interruption strategies for ablation. The first is **Fixed**: the model is forcibly interrupted after a fixed budget of B=256 tokens and required to emit a reflection. The second is **Random**: the model is interrupted after a budget sampled uniformly from $[128, 768]$ tokens. We compare these against the adaptive trigger learned by REFLECT+, evaluated on the three in-distribution math benchmarks MATH500, GSM8K, and AMC23 in Figure 2. Under

Continuer	MATH500	GSM8K	AMC23	Avg.
SFT (w/o RL)	27.2	52.9	23.8	34.6
REFLECT+ (w RL)	33.2	56.2	25.0	38.1
Qwen2.5-3B (external)	35.4	71.2	18.8	41.8
Qwen2.5-7B (external)	46.0	80.1	24.6	50.2

Table 2: Continuation skill (mean@8, %): four continuers resume from an identical Qwen2.5-7B teacher reflection r on the same reflection points, holding both the problem x and the reflection content fixed. Any gap is attributable solely to how well the continuer can act on the given reflection block. Sampling: n=8, temperature=1.0.

Benchmark	Refl. acc.	Direct acc.	Trig. rate
MATH500	29.2	74.8	18.3
AMC23	13.6	49.5	27.5
GSM8K	42.4	84.9	12.2
LogiQA	27.6	30.0	51.0
GPQA	19.2	17.8	56.7
MMLU-Pro	21.4	23.8	53.6

Table 3: Reflection behaviour of REFLECT+ on each benchmark (%). *Refl. acc.* and *Direct acc.* are the accuracy on reflection-triggered and on directly-answered samples; *Trig. rate* is the fraction of samples that emit a <reflection> block.

RL, REFLECT+'s adaptive trigger consistently outperforms both forced strategies: $+15.8/+11.6$ points over Fixed/Random on MATH500, $+10.8/+6.9$ on GSM8K, and $+17.2/+12.9$ on AMC23, for an in-distribution average of 61.2% vs. $46.6\%/50.8\%$. Forcing reflection at externally imposed positions degrades performance, indicating that the gain comes from *when* the model chooses to reflect, not from the act of reflecting itself. Without RL, the three strategies are statistically indistinguishable ($43.4/43.6/41.3$ average), and Random even edges above adaptive: cold-start supervision alone is sufficient to emit reflections in the correct format, but provides no signal about *when* they help. Role-aware policy optimisation is what turns the adaptive trigger into a real policy, lifting it from 43.4% to 61.2% while Fixed/Random gain comparatively less. These three patterns show that REFLECT+ learns a difficulty-adaptive timing policy, whose value is unlocked specifically by RL.

What to reflect. *What to reflect* is crucial because the reflection must preserve verified sub-conclusions and provide a corrected direction for clean-slate continuation. To evaluate reflection quality, we conduct a controlled replacement experiment: at inference time, we replace the model's own reflection r with one generated by an external teacher and continue from the rebuilt prompt (x, r) while keeping the continuer fixed. We use two off-the-shelf instruction-tuned teachers of different capacities (Qwen2.5-7B-Instruct and Qwen2.5-3B-Instruct). Results are shown in Figure 3.

Without RL, the cold-start model's reflections lag the stronger 7B teacher by 5.4 points and are comparable to the smaller 3B teacher, suggesting reflection-format supervision alone is insufficient to produce reflections that support continuation. Under RL, the absolute performance of all reflection sources decreases (e.g., on MATH500: $45.8 \rightarrow 33.2$ for the 7B teacher and $44.2 \rightarrow 32.4$ for the 3B teacher). We attribute this to a population shift rather than degraded reflection quality: RL-trained policies trigger reflection more selectively on difficult examples, making the reflection subset substantially harder. This is consistent with the trigger statistics in Table 3, where reflection is invoked more frequently on harder and OOD benchmarks. The meaningful comparison is therefore within the w-RL block. Here, REFLECT+'s internal reflections close the gap to the 3B teacher and even outperform it on MATH500 and AMC23 (33.4 vs. 32.4; 14.4 vs. 6.4). These results suggest that role-level optimisation improves the alignment between reflections and the model's own continuation policy, rather than imitating a generic external teacher.

How to continue. *How to continue* determines whether the model can coherently resume reasoning from the reflection without re-anchoring on the discarded prefix. In the Clean-Slate Reflection paradigm this property is enforced *structurally*: the continuation prompt is rebuilt as (x, r), so the faulty prefix is not available to bias subsequent tokens, thereby removing the failure mode of vanilla appended reflection by construction. To isolate the continuation skill from the reflection content, we hold r fixed across continuers: on the reflection points captured from REFLECT+, we attach the *same* Qwen2.5-7B teacher reflection to four different continuers (SFT, REFLECT+, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct) and let each one finish under clean-slate rebuild (x, r). Table 2 reports the results on the three in-distribution math benchmarks. REFLECT+ outperforms its SFT initialisation on every dataset, confirming that RL teaches a continuation skill rather than just a reflection-emission habit. A 1.5B-parameter continuer cannot close the gap to the much larger 3B/7B instruction-tuned teachers in absolute terms, but REFLECT+ is the strongest among models of its own scale, showing

the gain comes from how the continuer uses a fixed reflection rather than from the reflection text or the prompt-rebuild mechanism alone (representative reflection events are shown in Appendix H).

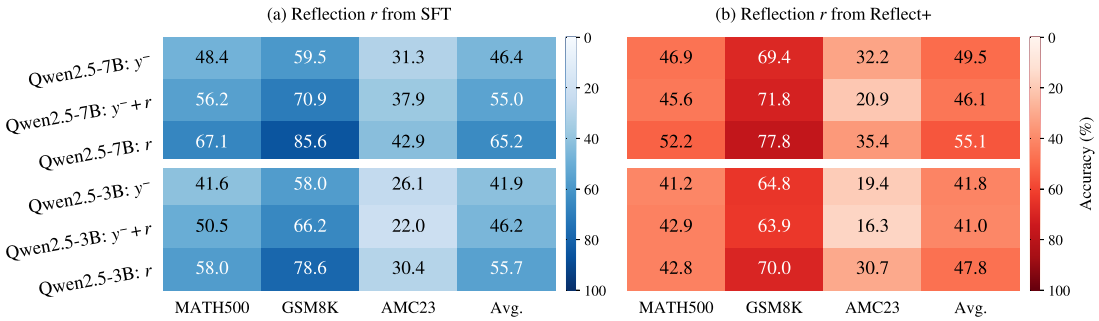

Figure 4: Reflection vs. error-prefix continuation. Mean@8 accuracy (%) across three input formats: (x, y^-), (x, y^-, r), and (x, r). Reflections r are generated by either the SFT model or REFLECT+.

Reflection tag vs. error prefix. Beyond continuation skill, we also investigate whether the reflection block r itself—rather than mere access to the faulty prefix y^-—is what drives the performance gain. For each reflection point, we compare three inputs to the same continuer: the faulty prefix alone (x, y^-), the appended-reflection rebuild (x, y^-, r), and the clean-slate rebuild (x, r). As shown in Figure 4, we find that conditioning solely on r consistently outperforms the other two setups, while appending r to y^- yields only mixed benefits depending on the reflection source. This confirms that (i) the reflection block captures a useful corrective signal independent of its origin, and (ii) discarding the faulty prefix—the defining feature of the Clean-Slate Reflection rebuild—provides an additive gain that does not require the reflection to override the misleading context.

Robust against spurious reflection. A defining property of REFLECT+ is that role-aware credit assignment and the format gate suppress spurious reflection. Table 3 shows two clear patterns: reflection is difficulty-adaptive, from 12.2% on GSM8K to 18.3% on MATH500 and 27.5% on AMC23 on 1.5B backbone, and the 4B preserves similarly, with trigger rates at 12.66%, 17.58%, and 34.61% respectively. While on five of six benchmarks, reflection-triggered samples are less accurate than directly answered ones, indicating that reflection is invoked mainly when direct solving is likely to fail. These results all suggest REFLECT+ learns selective rather than indiscriminate reflection.

5 CONCLUSION

We propose REFLECT+, an end-to-end framework for reflection-triggered self-correction at the role level. By separating format learning from strategy optimization, REFLECT+ enables models to learn *when* to reflect, *what* to summarise in the reflection block, and *how* to continue afterwards. Building on the Clean-Slate Reflection paradigm, where the faulty prefix is discarded and continuation resumes from the original problem and the reflection alone, our two-phase cold-start and REFLECT+ RL stage objective address the spurious-reflection and context-corruption failure modes reported in prior RL-trained reasoners (Huang et al., 2024; Yeo et al., 2025; Chen et al., 2025). Experimental results and reflection diagnostics show that the learned reflection behaviour is adaptive rather than imitative, with role-level optimization as the key factor. Cross-model evaluations on Qwen3-0.6B-Base and Qwen3-4B-Base indicate that REFLECT+ consistently broadens the solution-coverage, while per-sample improvements remain capacity- and task-dependent. REFLECT+ also produces the shortest responses on average (769 tokens), reducing generation length by 17% relative to GRPO and 34% relative to the base model on Qwen2.5-Math-1.5B, suggesting improved efficiency rather than verbose deliberation. Future work may extend verification beyond math to open-ended scientific reasoning and code generation, support open-ended evaluation, calibrate trigger policies across difficulty levels, and combine Clean-Slate Reflection with process-level reward models.

AI USE DISCLOSURE

In this work, we used generative AI tools for language polishing and grammar correction of the manuscript. We also used AI models as components within the experimental pipeline: specifically, we employed a commercial API teacher (e.g. DeepSeek-V4-Pro) for synthesizing structured reflection training instances from incorrect reasoning trajectories, and we used instruction-tuned Qwen models as external continuers in controlled replacement experiments. We have not used AI tools for research ideation, hypothesis generation, experimental design, or drafting substantive scientific content. All AI-assisted text and data have been verified by the authors. We take full responsibility for the final content, including all claims and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide comprehensive details for reproducing our results. Training data: All benchmark specifications are in Sections 3.2–4. Cold-start data construction: teacher schema, filtering criteria, and the 7,632 reflective / 2,057 perfect instance counts are in Appendix C. RL configuration: reward definitions, advantage computation, difficulty-gated floor, and PPO hyper-parameters are in Section 3.3 and Appendix D. Evaluation: sampling seeds (1234, 2026, 3407, 4519, 8675), temperature, pass@k computation, and answer verifier details are in Appendix E. Code and artifacts: Code, data preparation scripts, and evaluation pipelines will be released upon acceptance. All claims are supported by at least one of the following: a table, a figure, an appendix item, or an artifact command.

REFERENCES

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. Big-Math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. URL https://arxiv.org/abs/2502.17387.

Gregor Bachmann, Yichen Jiang, Seyed-Mohsen Moosavi-Dezfooli, and Moin Nabi. The potential of cot for reasoning: A closer look at trace dynamics. In *International Conference on Learning Representations*, volume 2026, pp. 96695–96717, 2026.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? on the overthinking of o1-like LLMs, 2025. URL https://arxiv.org/abs/2412.21187.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug, 2023. URL https://arxiv.org/abs/2304.05128.

Renfei Dang, Zhening Li, Shujian Huang, and Jiajun Chen. The first impression problem: Internal bias triggers overthinking in reasoning models. In *International Conference on Learning Representations*, volume 2026, pp. 141366–141404, 2026.

Hexuan Deng, Wenxiang Jiao, Xuebo Liu, Jun Rao, and Min Zhang. Rea-rl: Reflection-aware online reinforcement learning for efficient reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 47958–47980, 2026.

Chongyu Fan, Yihua Zhang, Jinghan Jia, Alfred Hero, and Sijia Liu. Cyclicreflex: Improving reasoning models via cyclical reflection token scheduling. In *International Conference on Learning Representations*, volume 2026, pp. 109066–109088, 2026.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing, 2024. URL https://arxiv.org/abs/2305.11738.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2024. URL https://arxiv.org/abs/2310.01798.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs, 2024. URL https://arxiv.org/abs/2406.01297.

Michael Kirchhof, Luca Füger, Adam Golinski, Eeshan Gunesh Dhekane, Arno Blaas, Seong Joon Oh, and Sinead Williamson. Selfreflect: Can llms communicate their internal answer distribution? In *International Conference on Learning Representations*, volume 2026, pp. 49948–50000, 2026.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamile Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

Young-Jun Lee, Seungone Kim, Byung-Kwan Lee, Minkyeong Moon, Yechan Hwang, Jong Myoung Kim, Graham Neubig, Sean Welleck, and Ho-Jin Choi. Refinebench: Evaluating refinement capability of language models via checklists. In *International Conference on Learning Representations*, volume 2026, pp. 81042–81085, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506, 2024. doi: 10.1162/tacl_a_00660. URL https://doi.org/10.1162/tacl_a_00660.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework, 2024. URL https://arxiv.org/abs/2409.19256.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Lina Sun. ReActR: Reasoning through error-activated reflection for LLM post-training. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 43017–43028, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1993. URL https://aclanthology.org/2026.acl-long.1993/.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting, 2023. URL https://arxiv.org/abs/2305.04388.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in LLMs, 2025. URL https://arxiv.org/abs/2502.03373.

Zewei Yu, Lirong Gao, Yuke Zhu, Bo Zheng, Junbo Zhao, Sheng Guo, and Haobo Wang. Stop unnecessary reflection: Training lrms for efficient reasoning with adaptive reflection and length coordinated penalty. *arXiv preprint arXiv:2602.12113*, 2026.

Xinliang Frederick Zhang, Anhad Mohananey, Alexandra Chronopoulou, Pinelopi Papalampidi, Somit Gupta, Tsendsuren Munkhdalai, Lu Wang, and Shyam Upadhyay. Do LLMs really need 10+ thoughts for "find the time 1000 days later"? towards structural understanding of LLM overthinking. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17005–17030, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.773. URL https://aclanthology.org/2026.acl-long.773/.

A Failure Modes of Existing Self-Correction

We catalogue three failure modes that motivate REFLECT+. *Under-trigger* occurs when the model commits to a wrong direction and never reflects; the per-prompt counterfactual baseline together with the difficulty-gated advantage floor in Section 3.3 mitigate this by giving reflection rollouts positive advantage on problems where direct attempts fail and by preventing reflection from being crushed on easy prompts. *Over-trigger* occurs when the model reflects on every problem, paying token cost for no accuracy benefit; the perfect-sample mixing in Section 3.2 and the efficiency factor R_{eff} in Eq. 7 suppress this. *Direction-preserving reflection* occurs when the reflection text is syntactically valid but semantically does not change the reasoning—the classic spurious-reflection failure identified in Yeo et al. (2025). Clean-slate prompt rebuild is the structural countermeasure: even when the reflection is weak, the continuation is no longer locked to the faulty prefix and is free to follow the surviving correct sub-conclusions.

B Implementation Details

Reflection prompt. The system prompt used for cold-start SFT, and carried over to the RL stage:

> You are a math reasoning assistant. When you make an error during reasoning, use <reflection>...</reflection> to reflect on the mistake, then continue reasoning based on the reflection.

Format-gate regular expression. A reflection block is well-formed if its body matches the case-insensitive, dotall pattern:

```
\bso\s+far\s,?\s+i\s+have\b .?\bhowever\b
```

and the numbers of opening and closing <reflection>/</reflection> tags are equal.

Reward and advantage hyper-parameters. Wrong-no-reflect penalty parameters n_w=3, ϕ=4 (yielding penalty $-(2/4)^2 = -0.25$); efficiency-factor free budget d_{free}=1 and exponent ϕ=4 (so two reflections give $R_{\text{eff}} = 1-(1/4)^2 = 0.9375$); difficulty-gated floor $A_{\text{floor}} = -0.5$ with per-prompt gate $w_x = \text{clip}(\mu_x^{\text{dir}}, 0, 1)$; asymmetric PPO clip ϵ_{lo}=0.20, ϵ_{hi}=0.26; KL coefficient=0.

C Cold-Start Configuration

Cold-start data construction details. We construct the cold-start corpus from the hard split of Big-Math (Albalak et al., 2025). For each problem, Qwen2.5-Math-1.5B samples one reasoning trajectory at temperature 0.5, which is then verified for correctness. Incorrect trajectories are transformed by a strong API teacher (Deepseek-V4-Pro) into Clean-Slate Reflection training instances, while correct minimal solutions are retained as direct-solving examples.

For each incorrect solution, the teacher identifies the first faulty step and returns a structured analysis consisting of: (i) `cut_point`, the reasoning prefix before the error; (ii) `verified_states`, the correct intermediate conclusions established so far; (iii) `error_reason`, a one-sentence diagnosis of the mistake; and (iv) `new_direction`, a one-sentence corrected direction.

The structured analysis is converted into the fixed reflection template:

> <reflection> So far I have: {verified_states}. However, {error_reason}. {new_direction} </reflection>

The final corpus contains 7,632 reflective and 2,057 perfect instances, with a 90/10 train/eval split within each type. Perfect samples are mixed into training to preserve direct-solving behaviour and prevent the model from reflecting indiscriminately on easy problems.

LoRA adapter. Both phases fine-tune Qwen2.5-Math-1.5B with a LoRA (Hu et al., 2021) adapter of rank 16, α=32, dropout 0.05, applied to all attention projections and MLP gates. The original tokenizer is retained and <reflection>/</reflection> are treated as plain text markers, so no embedding resize is required.

Optimisation. Phase 1 trains for three epochs at learning rate 2×10^{-4}, maximum sequence length 2048, cosine schedule, and a 90/10 train/eval split within each record type (random seed 42). Phase 2 starts from the Phase 1 adapter and trains for two epochs at learning rate 5×10^{-5} with the three-way example mixing of Section 3.2 in roughly 64:19:17 ratio (reflection-to-solution : Phase-1 replay : perfect). Both phases use the chat template of the underlying tokenizer.

D RL Hyper-parameters

Optimisation and sampling. Group size G=8, prompt batch size 64, actor learning rate 1×10^{-6}, maximum response length 1024, maximum model length 4096, sampling temperature 1.0, top-p 1.0, asymmetric PPO clip $(\epsilon_{\text{lo}}, \epsilon_{\text{hi}})$=(0.20, 0.26), KL coefficient 0, maximum two reflection events per trajectory (i.e. at most three generation rounds).

Training stack. We implement Reflect+ on the `verl` framework (Sheng et al., 2024), which runs rollout (vLLM) and policy update (FSDP) on separate engines, with token-level masking for tokens whose inference- vs. training-engine log-probabilities diverge.

Stability safeguards. For uniform-outcome groups (all G rollouts of a prompt correct or all wrong), we skip the standardisation in Eq. 8 and instead use the unnormalised advantage $R_i - \mu_x^{\text{dir}}$. This avoids amplifying small stochastic reward differences caused by formatting, depth penalties, or sampling variability into unstable normalized gradients.

E Evaluation Configuration

For the results in Table 1, MATH500, AMC23, GSM8K, LogiQA, and GPQA use 32 generations per problem at temperature 1.0 and report Mean@32 and Pass@32. BBH, DROP, and CLUTRR

report Mean@8 and Pass@8. MMLU-Pro uses greedy decoding and reports accuracy only. Overall accuracy is the unweighted average across all nine benchmarks; overall pass@k averages the eight benchmarks excluding MMLU-Pro.

F Training Dynamics

We track the reflection trigger rate, the per-role reward, the recovery rate, and the regression rate over the course of RL training. Curves are produced by the training-time monitor. The target signature is: a selective trigger rate on in-distribution validation (not all-example reflection), a positive event reward, monotonically improving validation accuracy, and a recovery rate clearly above the regression rate.

G Reflection Template Example

A typical training example is:

> **Problem:** Find $\sum_{k=1}^{10} k(k+1)$.
> **Error prefix:** ``We have $k(k+1) = k^2 - k$, so the sum is $\sum k^2 - \sum k$...''
> **Reflection:** <reflection> So far I have: the goal is to evaluate $\sum_{k=1}^{10} k(k+1)$, and the standard identities $\sum k = n(n+1)/2$ and $\sum k^2 = n(n+1)(2n+1)/6$ are available. However, the previous step expanded $k(k+1)$ as $k^2 - k$ instead of k^2+k. Re-expand $k(k+1) = k^2+k$ and apply the two standard identities. </reflection>
> **Clean-slate continuation:** ``$\sum k^2 + \sum k = 385 + 55 = 440$. The answer is $\boxed{440}$.''

H Continuation Case Studies

To make the *how-to-continue* effect concrete, we present three representative reflection events drawn from GSM8K (two in-distribution arithmetic cases) and LogiQA (one out-of-distribution case). For each event we hold the reflection r fixed and place the two continuations side by side: the clean-slate continuation conditioned on (x, r), and the context-preserving continuation conditioned on (x, y^-, r). Because r is identical, any difference in the continuation is attributable solely to the presence or absence of the faulty prefix y^-. In all three cases the clean-slate continuation reaches the gold answer while the appended continuation re-anchors on the slip already present in y^- and arrives at a different, incorrect answer.

Case 1 — GSM8K arithmetic (`gsm8k-905`). *Query x:* "Harold sleeps for 10 hours a night, works 2 hours less than he sleeps, and walks his dog for 1 hour each day. How much free time does Harold have left in his day?" Gold answer 5 hours.
Faulty prefix y^-: the model correctly derives "sleep $= 10$, work $= 10 - 2 = 8$, dog $= 1$," but then writes "the total number of hours Harold spends on sleep and walking his dog is $10 + 1 = 11$ hours" – silently *dropping the work hours* from the budget sum.
Reflection r: <reflection> So far I have: Harold sleeps 10 h; works $10 - 2 = 8$ h; walks the dog 1 h. However, the model incorrectly assumes work hours are included in the sleep-and-dog-walking total, which omits the work hours entirely. Sum sleep, work, and dog-walking and subtract from 24 h. </reflection>
Clean-slate continuation (x, r): sums all three activities ($10 + 8 + 1 = 19$ h) and reports $24 - 19 = \boxed{5}$ h. The reflection's diagnosis ("work was dropped") is explicitly cashed in.
Appended continuation (x, y^-, r): re-uses the prefix's "$10 + 1 = 11$" line verbatim instead of summing all three activities, and reports $24 - 11 = \boxed{13}$ h. The reflection is in context but the visible prefix arithmetic wins.

Case 2 — GSM8K arithmetic (`gsm8k-526`). *Query x:* "Ada uses 12 kWh of electricity per day; a new device consumes 2 kWh per day on top of that. Electricity costs \$1.50 per kWh. What is

the difference in her weekly electric bill before vs. after adding the device?" Gold answer $21.
Faulty prefix y^-: the prefix mis-types the unit as "12 kilowatts *per hour*" and writes "weekly consumption $= 12 \times 24 \times 7 = 2016$ kWh"—double-counting the per-hour factor.
Reflection r: `<reflection>` So far I have: daily consumption 12 kWh; new device 2 kWh/day; combined daily 14 kWh. However, the prefix treats "12 kWh" as a rate per hour and multiplies by 24, while the problem gives a daily total. Use 12 and 14 as daily kWh totals when computing the weekly bill. `</reflection>`
Clean-slate continuation (x, r): treats the 12 and 14 values as daily kWh; difference $= 2$ kWh/day; weekly $= 2 \times 7 = 14$ kWh; bill difference $= 14 \times \$1.50 = \boxed{\$21}$.
Appended continuation (x, y^-, r): keeps the prefix's $\times 24 \times 7$ expansion in scope, writes $2 \times 24 \times 7 \times \$1.50 = \boxed{\$453.60}$. Despite the reflection's explicit rewording of the unit, the prefix's formula is re-applied.

Case 3 — LogiQA (out-of-distribution; `logiqa-7364`). *Query x:* a two-player payoff-matrix question (Zhang vs. Li with payoffs $(A, A) = (3, 3)$, $(A, B) = (0, 5)$, $(B, A) = (5, 0)$, $(B, B) = (2, 2)$) asking which of four statements about the players' best responses is correct. Gold answer B ("when Zhang chooses A, Li will choose B").
Faulty prefix y^-: the model commits to "Answer: C — when Li chooses scheme B, Li will choose scheme B," which is a circular re-statement of the choice rather than a best-response argument, and it pairs the wrong payoffs with the wrong cells.
Reflection r: `<reflection>` So far I have: payoff matrix is correctly identified. However, the previous step contradicts the matrix: Li gets 2 when she chooses B *only* if Zhang also chooses B; the "Li chooses B, so Li chooses B" chain is not a valid best-response. Compute Li's best response by comparing her two payoffs under each of Zhang's choices and identify the dominant strategy. `</reflection>`
Clean-slate continuation (x, r): reconstructs the matrix from (x, r) alone, derives that "if Zhang plays A, Li's payoffs are 3 vs. 5, so Li plays B; if Zhang plays B, Li's payoffs are 0 vs. 2, so Li plays B," concludes that B is Li's dominant strategy, and outputs $\boxed{B}$.
Appended continuation (x, y^-, r): reproduces the matrix correctly but then re-derives the prefix's circular "when Li chooses B, Li chooses B" line and again outputs $\boxed{C}$. Even with the explicit reflection in context, the visible prefix re-anchors the continuation on the same non-argument.

I Cold Start and the RL Training Procedure

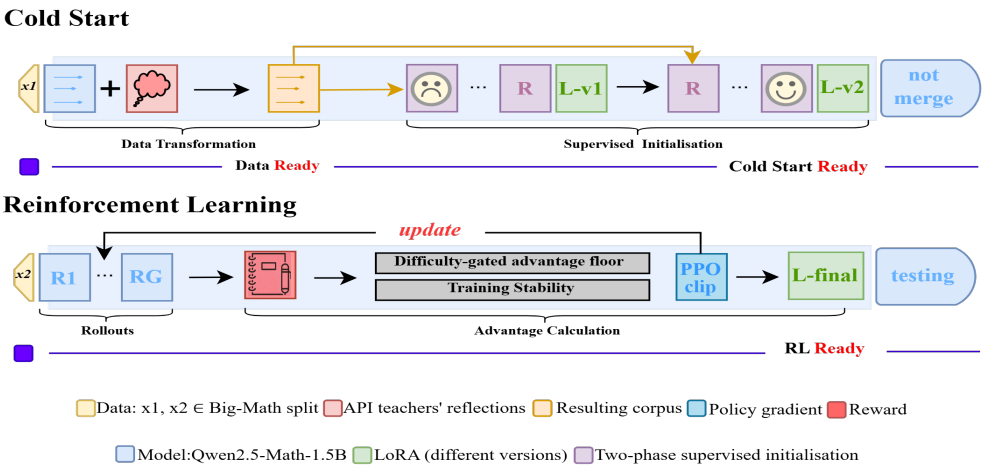

Figure 5: Two-phase cold start and the RL training procedure of REFLECT+. Cold-start supervision teaches the reflection format; RL teaches when, what, and how to reflect (1.5B model as an example).

Benchmark	n_{hard}	Recovery (%)
MATH500	350	6.0
GSM8K	329	13.4
AMC23	45	6.7
GPQA	500	9.0
LogiQA	2821	20.3

Table 4: Recovery on *hard* reflection points where the faulty prefix y^- cannot be salvaged by any of four strong continuers (GRPO, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, and Qwen3-8B) continuing from (x, y^-) alone. n_{hard} is the size of this subset on each benchmark; the *Recovery* column reports the fraction recovered by REFLECT+'s own reflect-then-continue policy (reflection drawn from REFLECT+, continuation in clean-slate rebuild).

J RECOVERY ON HARD REFLECTION POINTS

To complement the aggregate accuracy, we isolate the *hard* subset of reflection points: cases where four strong continuers (GRPO, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-8B) all fail to recover the answer when continuing from the error prefix y^- alone. On this hard subset, REFLECT+'s own reflect-then-continue policy still succeeds non-trivially (Table 4): 13.4% on GSM8K and 20.3% on LogiQA, with smaller but non-zero recovery rates on MATH500, AMC23, and GPQA. These cases are by construction unreachable by direct continuation from y^-, so any recovery is attributable to the combined effect of the reflection block and the clean-slate rebuild.

K FULL CROSS-BACKBONE RESULTS

Model	MATH500	GSM8K	LogiQA	GPQA	BBH	DROP	Avg.
Qwen3-0.6B							
SFT	**84.40**	86.43	51.40	37.88	78.41	17.79	59.39
GRPO	83.80	90.90	**58.52**	**45.96**	87.53	12.78	63.25
REFLECT+	83.20	**91.96**	53.31	44.95	**89.13**	**20.49**	**63.84**
Qwen3-4B							
SFT	92.20	96.51	90.33	80.81	87.01	86.51	88.90
GRPO	91.60	96.74	90.90	82.32	86.78	85.39	88.96
REFLECT+	**94.60**	**97.80**	**94.27**	**92.42**	**87.08**	**87.11**	**92.21**

Table 5: Cross-model pass@32 results (%) on six representative benchmarks for Qwen3-0.6B and Qwen3-4B. *Avg.* is the unweighted macro-average of the six displayed benchmarks. **SFT** denotes the cold-start model, **GRPO** the vanilla GRPO baseline, and **REFLECT+** the difficulty-gated reflection RL model. Best results are highlighted in **bold** within each backbone.