LEARNING TO REASON FROM FEEDBACK: CONSISTENCY-AWARE POST-TRAINING FOR LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training has made substantial progress at optimizing the final outcome of reasoning tasks, but an equally important ability is rarely optimized as a target in its own right: how a model integrates newly acquired feedback into its subsequent reasoning and decisions. This becomes acute once the model must interact with external tools and execution environments, where it has to judge when additional feedback is warranted, absorb new information without derailing its existing line of reasoning, and tell which interactions actually contributed to success. We treat reasoning from feedback as an explicit post-training objective. To this end we introduce three complementary training signals: a contrastive representational constraint that keeps the model's reasoning state semantically coherent across an interaction while it absorbs the new observation; an uncertainty-based interaction policy that leads the model to seek feedback selectively rather than reflexively; and process-level credit assignment from counterfactual trajectory value estimation, which separates genuinely helpful interactions from redundant or harmful ones. The three are optimized jointly with Group Relative Policy Optimization (GRPO) and trained end-to-end on trajectories the model generates through interaction with a real execution environment. Instantiated in a sandboxed code-execution environment and evaluated on AIME 2024/2025, MATH-500, LiveCodeBench, and GPQA Diamond, our 7B model improves over the strongest same-size agentic RL baseline by 4.8 points on average. Analysis shows that the representational constraint improves feedback-utilization precision by 23%, while uncertainty-based gating reduces redundant interactions by 41%.

1 INTRODUCTION

Reinforcement learning from verifiable outcomes has become the standard recipe for post-training reasoning models (Shao et al., 2024; DeepSeek-AI et al., 2025; Yu et al., 2025a). Its training signal is almost always the final outcome: a trajectory is rewarded when its answer is correct. A strong reasoner, however, is also characterized by what it does with information it did not have at the outset. A mathematician who runs a quick computation has decided that the computation is worth running, reads its result in light of what she was trying to establish, and afterwards knows which of her checks actually settled the question. Language models exercise the same three abilities—*seeking*, *integrating*, and *attributing* feedback—whenever they interact with a code interpreter, a search engine, or another execution environment (Yao et al., 2023; Gou et al., 2024; Feng et al., 2025). Under outcome-only training, these abilities are learned only indirectly, as whatever happens to correlate with a correct final answer.

We find that this leaves all three under-developed. In a pilot study of an agentic GRPO baseline (Section 3), 34.5% of its incorrect AIME 2024 trajectories contain an *integration failure*, in which the model reads an observation and then reasons as if it had not; the most frequent pattern, present in 21% of incorrect trajectories, is overriding a correct intermediate result. Only 52.3% of its interactions are productive, and on MATH-500 it queries the interpreter on 71% of the easiest problems, almost as often as on the hardest ones (84%). Finally, a trajectory with seven interactions receives a single scalar reward, which cannot distinguish the check that settled the problem from checks that were redundant or misleading.

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.

We therefore treat *reasoning from feedback* as an explicit post-training objective. We decompose it into three capabilities and attach to each a training signal that is optimized jointly with the policy (Figure 1):

- **Coherent integration.** A contrastive constraint on hidden states requires the state after an observation to remain identifiable as the continuation of the request that produced it. This discourages the line of reasoning from drifting across the interaction, while leaving most of the representation free to absorb the new information.
- **Attributable credit.** A counterfactual estimate of how much each interaction changed the trajectory's value, computed with a learned verifier instead of additional rollouts, turns the outcome into interaction-level rewards.
- **Selective seeking.** A lightweight gate on the model's hidden state, meant to reflect its uncertainty and trained on the sign of the counterfactual credit, decides whether a requested interaction is executed.

The three signals are optimized with GRPO on trajectories that the model generates in a sandboxed code-execution environment; apart from a small supervised warm start, no teacher trajectories are used. Because each signal asks the model to remain consistent with what it has observed—in its internal state, in its decisions to query, and in the credit it assigns—we refer to the procedure as *consistency-aware post-training*.

On AIME 2024/2025, MATH-500, LiveCodeBench v6, and GPQA-Diamond, our 7B model improves the five-benchmark average by 4.8 points over the strongest same-size agentic RL baseline in our evaluation, and by 12.1 points over a GRPO baseline trained with the same data, environment, and budget. It does so with 40% fewer interactions per AIME trajectory; the reduction comes almost entirely from redundant interactions ($3.2 \rightarrow 0.7$ per trajectory), while the number of productive ones stays roughly constant. Ablations show that every signal contributes, and that the coherence and gating signals are complementary but partly redundant: each helps in the presence of the other, yet their joint gain is smaller than the sum of their separate gains.

Contributions. (1) We formulate reasoning from feedback as a post-training objective with three measurable components, each with its own diagnostic (Section 3). (2) We introduce three training signals and a joint GRPO objective, and state precisely what each does and does not guarantee; for instance, the gate learns a *median* criterion on counterfactual credit rather than its mean, and the credit estimator is unbiased only in an idealized setting that our practical approximation does not meet (Section 4, Appendix B). (3) We report a controlled study, holding the backbone, data, environment, and compute fixed, that measures gains in accuracy and interaction efficiency and isolates the effect of each signal (Section 5).

2 Related Work

RL for reasoning and tool use. Outcome-reward RL with group-relative advantages (Shao et al., 2024) elicits long-form reasoning (DeepSeek-AI et al., 2025); its stability has been improved by asymmetric clipping, token-level aggregation, and overlong shaping (Yu et al., 2025a), and its signal densified with implicit process rewards (Cui et al., 2025). Tool-integrated reasoning has moved from prompting (Gao et al., 2023; Chen et al., 2023; Yao et al., 2023; Paranjape et al., 2023) and supervised imitation (Schick et al., 2023; Gou et al., 2024; Wang et al., 2023a), including large-scale API use (Qin et al., 2023; Patil et al., 2023), to RL in executable environments (Li et al., 2025b; Feng et al., 2025; Dong et al., 2025b;a; Shang et al., 2025; Yu et al., 2025b) and to search agents (Li et al., 2025a; Song et al., 2025); RLAnything co-evolves the environment, policy, and reward model (Wang et al., 2026b). These methods optimize the final outcome, sometimes with shaping that encourages or limits tool use. We add signals that target how observations are integrated, sought, and credited.

Deciding when to interact. Tool overuse has been addressed with self-awareness data (Qian et al., 2025), rewards for efficient tool calls (Wang et al., 2025), uncertainty-aligned rewards over tool decisions (Zhou et al., 2026), query-level routing from counterfactual margins between tool-based and tool-free answers (Zou et al., 2026), and necessity rewards that counter spurious cues (Yang et al., 2026). Our gate operates on each interaction inside a multi-turn trajectory and learns from counterfactual credit rather than from labels or judges, and we characterize its learning target. The

token-level predictive entropy that we use to estimate this uncertainty conflates epistemic and aleatoric uncertainty (Kuhn et al., 2023); we treat it only as an inexpensive proxy.

Credit assignment in multi-step reasoning. Process supervision relies on human (Uesato et al., 2022; Lightman et al., 2024) or Monte Carlo (Wang et al., 2024; Luo et al., 2024) step labels; VinePPO replaces the critic with Monte Carlo value estimates (Kazemnejad et al., 2024); branch-based methods derive local credit from rollouts that share a prefix (Zhao et al., 2026; Wang et al., 2026a). Counterfactual and difference-based credit has a long history in multi-agent RL (Foerster et al., 2018) and in hindsight credit assignment (Harutyunyan et al., 2019). Closest to our credit signal, Meng (2026) estimates policy-conditioned contributions of interventions such as step deletion or evidence substitution, and Zhang (2026) audits step-level credit against executed replay. Our estimator is cheaper, requiring no additional rollouts, at the price of a heuristic approximation whose limitations we state explicitly.

Consistency and contrastive constraints. Consistency has been used to aggregate answers (Wang et al., 2023b), to make rationales faithful (Lyu et al., 2023), and to regularize reward models across trajectory prefixes (Zhou et al., 2025). Contrastive objectives with projection heads (van den Oord et al., 2018; Chen et al., 2020) and asymmetric architectures (Grill et al., 2020) learn invariances without representational collapse. We use such an objective as an auxiliary constraint during RL, applied across interaction boundaries within a trajectory.

3 Reasoning from Feedback as a Post-Training Objective

Setting. A policy π_θ solves a problem x by alternating reasoning and interaction. A trajectory

$$\tau = (x,\ s_1, q_1, o_1,\ s_2, q_2, o_2,\ \ldots,\ s_M, q_M, o_M,\ s_{M+1}) \tag{1}$$

consists of reasoning segments s_m generated by π_θ; interaction requests q_m (here, Python programs) that close segment m; observations $o_m \sim \mathcal{E}(\cdot \mid q_m)$ returned by the environment $\mathcal{E}$ (program output or an error message); and a final segment s_{M+1} that ends with an answer $\hat{y}$. The outcome reward is $R(\tau) = +1$ if $\hat{y}$ is correct and -1 otherwise. We view generation as a Markov decision process whose state is the full interaction history, with deterministic transitions for generated tokens and stochastic transitions for observations. For each interaction we record two last-layer hidden states: $\boldsymbol{h}_m^- \in \mathbb{R}^d$ at the final token of s_m, just before the request is emitted, and $\boldsymbol{h}_m^+$ at the first token of s_{m+1}, after o_m has been read.

Three capabilities. We say a model reasons well from feedback if it (C1) *integrates* each observation coherently, continuing the line of reasoning that motivated the request while taking the observation into account; (C2) *seeks* feedback selectively, interacting when interaction is likely to help; and (C3) can *attribute* its success to the interactions that caused it. Outcome reward optimizes none of these directly.

Pilot evidence. We examined the agentic GRPO baseline of Section 5.1 on AIME 2024. For (C1), two annotators, blinded to model identity, independently marked an integration failure when the post-observation reasoning (i) contradicted a correct observation without justification, (ii) ignored an observation directly relevant to the active sub-goal, or (iii) used a value inconsistent with the observation; disagreements were resolved by adjudication. At least one integration failure occurs in 34.5% of incorrect and 6.0% of correct trajectories. The most common pattern (21% of incorrect trajectories) is *result override*: the program returns a correct intermediate result, and the model continues as if it had returned something else. For (C2), each executed interaction was independently labeled *productive* if its observation was subsequently used to change the reasoning direction, correct an error, or verify an intermediate result needed for the final solution, and *redundant* if it affected neither the subsequent reasoning nor the final answer; disagreements were again adjudicated. Only 52.3% of interactions are productive, and on problems that the model reliably solves without interaction, allowing interaction lowers accuracy by 8.4 points. For (C3), trajectories contain 6.8 interactions on average, all of which share one scalar outcome.

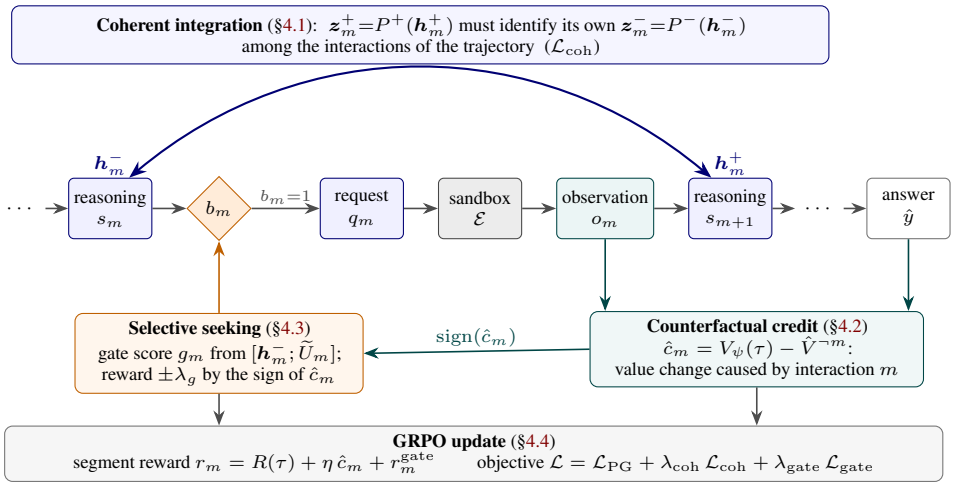


Figure 1: **Overview.** The policy alternates reasoning segments and interaction requests. Before a request is executed, a gate decides whether to execute it; if $b_m{=}0$, decoding continues without an observation. Three training signals act on each interaction: a contrastive constraint between the states before and after the interaction (C1), a gate trained to query selectively (C2), and counterfactual credit that measures how much the interaction changed the trajectory's value (C3). All are optimized jointly with GRPO.

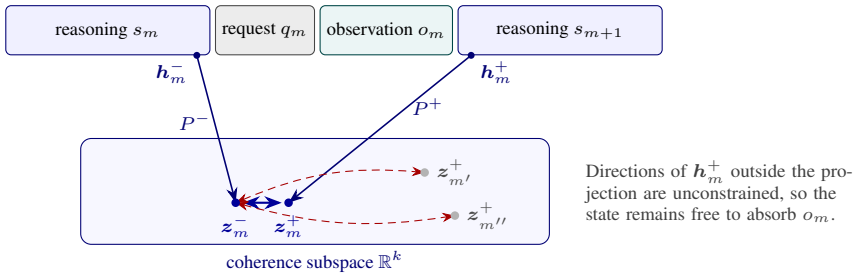


Figure 2: **Coherence constraint** (Equation (2)). The last state before the request, $\boldsymbol{h}_m^-$, and the first state after the observation, $\boldsymbol{h}_m^+$, are projected into a k-dimensional subspace. There, $\boldsymbol{z}_m^-$ must be closer to its own $\boldsymbol{z}_m^+$ (solid) than to the post-observation projections of the trajectory's other interactions (dashed: negatives).

4 Method

Figure 1 summarizes the approach. Each capability of Section 3 is paired with a training signal: a contrastive constraint for (C1), counterfactual credit for (C3), and a gate trained from that credit for (C2). We describe credit before the gate because the gate learns from it.

4.1 Coherent integration via state contrast

Two projection heads $P^-, P^+ : \mathbb{R}^d \to \mathbb{R}^k$ (two-layer MLPs, $k \ll d$) map the pre-request and post-observation states to unit vectors $\boldsymbol{z}_m^\pm = P^\pm(\boldsymbol{h}_m^\pm)/\|P^\pm(\boldsymbol{h}_m^\pm)\|$. For a trajectory with $M \geq 2$ interactions, the coherence constraint is

$$\mathcal{L}_{\text{coh}}(\tau) = -\frac{1}{M}\sum_{m=1}^{M} \log \frac{\exp\left(\langle \boldsymbol{z}_m^-, \boldsymbol{z}_m^+\rangle/\kappa\right)}{\sum_{m'=1}^{M}\exp\left(\langle \boldsymbol{z}_m^-, \boldsymbol{z}_{m'}^+\rangle/\kappa\right)}, \tag{2}$$

with temperature κ (Figure 2). The loss asks that the state after each observation identify its own request among all requests of the trajectory: whatever the observation says, the model should still be recognizably pursuing the sub-goal that motivated the interaction. Negatives come from the same trajectory, so they share the problem context and differ in what was being pursued, which is

exactly the distinction the constraint should preserve. For $M = 1$, Equation (2) is identically zero, so trajectories with $M \leq 1$ do not contribute to this term. Gradients reach the backbone through $\boldsymbol{h}_m^{\pm}$.

Why constrain projections contrastively. A direct penalty $\|\boldsymbol{h}_m^- - \boldsymbol{h}_m^+\|^2$ is minimized by ignoring the observation, which defeats the purpose of interacting; in our ablation this variant is worse than removing the constraint altogether (Table 2). Equation (2) constrains only a k-dimensional projection, and because the two sides have separate heads, the post-observation state remains free to change outside what P^+ reads.

What the constraint does not guarantee. Identifiability of the request is necessary for coherent integration but not sufficient: a state can remain identifiable while ignoring the observation (Appendix B.1). We therefore do not treat the learned similarity as evidence of integration, and measure integration behaviorally instead (Section 5.3).

4.2 Counterfactual interaction credit

Estimand. Let $\tau^{\neg m}$ denote the trajectory in which interaction m is removed: the request q_m is not issued, and π_θ continues from the prefix $\tau_{<m}$ that ends with s_m. The credit of interaction m is the change in success probability that it causes,

$$c_m^\star = V^\star(\tau) - V^\star(\tau^{\neg m}) \in [-1, 1], \tag{3}$$

where $V^\star$ is the probability that the final answer is correct, taken in expectation over the policy's continuations in the case of $\tau^{\neg m}$. If a value model V_ψ is unbiased at both arguments, then $V_\psi(\tau) - V_\psi(\tau^{\neg m})$ is unbiased for $c_m^\star$ by linearity of expectation (Appendix B.3). Evaluating it, however, requires sampling a continuation of $\tau^{\neg m}$ for every interaction.

Practical estimator. To avoid these rollouts, we approximate the counterfactual value from quantities available on the factual trajectory:

$$\hat{V}^{\neg m} = V_\psi(\tau_{<m}) + w_m\, V_\psi(\tau_{>m}), \qquad w_m = 1 - \cos(\boldsymbol{h}_m^+, \boldsymbol{h}_m^-), \qquad \hat{c}_m = V_\psi(\tau) - \hat{V}^{\neg m}, \tag{4}$$

where $\tau_{>m}$ is the continuation after o_m, whose value is weighted by the dissimilarity w_m between the states before and after the interaction, with no additional clipping. This costs one verifier pass per interaction. It is a heuristic: the unbiasedness argument above does not cover it, since the observed continuation may itself depend on o_m, and $\hat{V}^{\neg m}$ is not guaranteed to lie in $[0, 1]$. We assess its quality against expert step labels in Section 5.5 and discuss calibration against executed replay in Section 6.

Verifier. V_ψ is initialized from the supervised checkpoint and regressed with a squared loss onto Monte Carlo targets: for a partial trajectory, the fraction of 32 sampled continuations that reach a correct answer. Since $V^\star$ is a probability while $R(\tau) \in \{\pm 1\}$, credit enters the reward with a weight η (Section 4.4).

4.3 Selective feedback seeking

When the policy emits a request q_m, a gate decides whether it is executed. Let $U_m = -\sum_{v \in \mathcal{V}} p_\theta(v \mid \tau_{<m}) \log p_\theta(v \mid \tau_{<m})$ be the predictive entropy of the next-token distribution over the vocabulary $\mathcal{V}$, and $\widetilde{U}_m = U_m / \log|\mathcal{V}| \in [0, 1]$ its normalized value. The gate is a two-layer MLP with a sigmoid output that takes the pre-request state and the normalized entropy as input,

$$g_m = \sigma\big(\mathrm{MLP}_\phi([\boldsymbol{h}_m^-; \widetilde{U}_m])\big). \tag{5}$$

During training the decision is sampled, $b_m \sim \mathrm{Bernoulli}\big(\sigma(\beta(g_m - \mu))\big)$, where the threshold μ is an exponential moving average of recent gate scores; at inference, $b_m = \mathbf{1}[g_m > \mu]$ with μ frozen. If $b_m = 0$, the request is not executed and decoding continues without an observation. The gate receives the reward

$$r_m^{\mathrm{gate}} = \begin{cases} +\lambda_g & \text{if } b_m = 1 \text{ and } \hat{c}_m > 0, \\ -\lambda_g & \text{if } b_m = 1 \text{ and } \hat{c}_m \leq 0, \\ 0 & \text{if } b_m = 0, \end{cases} \tag{6}$$

and is trained with the score-function objective $\mathcal{L}_{\mathrm{gate}}(\phi) = -\mathbb{E}\big[r_m^{\mathrm{gate}} \log \Pr_\phi(b_m)\big]$.

What the gate learns. In a fixed context, executing yields expected reward $\lambda_g(2p-1)$ with $p = \Pr(\hat{c}_m > 0 \mid \text{context})$, and not executing yields 0. The optimal decision is therefore to execute if and only if $p > 1/2$, that is, if the *median* credit is positive (Proposition 1). This differs from a positive *mean*: credit $+0.1$ with probability 0.6 and -1 with probability 0.4 gives $p = 0.6$ but a mean of -0.34. The median criterion is robust to occasional large credit estimates, which an approximate estimator can produce, but it tolerates interactions that are usually slightly helpful and occasionally very harmful; a magnitude-aware reward $\lambda_g \hat{c}_m$ would target the mean instead.

Warm-up. Early credit estimates are noisy, and a gate trained on them can suppress interaction before the policy has learned when interaction helps. For the first 10% of RL steps we therefore keep the gate open ($b_m = 1$) and set $\lambda_{\text{gate}} = 0$, so that credit is estimated over the full range of interactions before pruning begins.

4.4 Joint optimization

Rewards and advantages. Every segment receives a reward: $r_m = R(\tau) + \eta\, \hat{c}_m + r_m^{\text{gate}}$ for segments that end in a request ($m \le M$), and $r_{M+1} = R(\tau)$ for the final segment. For a group of G trajectories sampled for the same problem, advantages are normalized over all segment rewards in the group, $\hat{A}_m^{(n)} = (r_m^{(n)} - \bar{r})/(\text{std}(r) + \epsilon)$, and every token of s_m and q_m receives $\hat{A}_m^{(n)}$; observation tokens are excluded from the loss.

Objective. We minimize

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{PG}}(\theta) + \lambda_{\text{coh}}\, \mathbb{E}_\tau\big[\mathcal{L}_{\text{coh}}(\tau)\big] + \lambda_{\text{gate}}\, \mathcal{L}_{\text{gate}}(\phi), \tag{7}$$

where $\mathcal{L}_{\text{PG}}$ is the clipped GRPO surrogate (Shao et al., 2024) with asymmetric clipping ($\epsilon_{\text{low}} = 0.2$, $\epsilon_{\text{high}} = 0.28$), token-level aggregation, no KL penalty, and overlong shaping with a 4,096-token buffer (Yu et al., 2025a). Before RL, the policy is warm-started by supervised fine-tuning on 3K interaction demonstrations. Algorithm 1 in Appendix A gives the full procedure. Relative to the GRPO baseline, the coherence term adds about 5% to training time, the gate less than 1%, and the verifier passes about 15%.

5 Experiments

5.1 Setup

Environment and models. Interactions are Python programs executed in SandboxFusion (ByteDance Seed Foundation Code Team et al., 2024) with a 30 s timeout and 2 GB of memory, with at most 16 interactions and 20,480 generated tokens per trajectory. Our main backbone is Qwen2.5-7B-Instruct (Qwen Team et al., 2024); we also train Qwen3-4B-Instruct-2507 (Yang et al., 2025).

Training. The supervised warm start uses 3K demonstrations (1K each for mathematics, code, and science; 2 epochs, learning rate 2×10^{-5}). RL then runs for 1,000 steps on 30K prompts (10K mathematics, 10K code, 10K science and mixed) with group size $G = 16$, 64 prompts per batch, and learning rate 10^{-6}. We use $\lambda_{\text{coh}} = 0.1$, $\lambda_{\text{gate}} = 0.05$, $\eta = 0.3$, $\kappa = 0.07$, $\lambda_g = 0.1$, and $\beta = 5$, selected on a held-out validation set of 500 problems from the original MATH training split, after excluding all MATH-500 evaluation instances; sensitivity is reported in Appendix C. Dataset decontamination is described in Appendix A.

Benchmarks and protocol. We evaluate on AIME 2024 and 2025 (30 problems each), MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), LiveCodeBench v6 (August 1, 2024–May 1, 2025; Jain et al., 2025), and GPQA-Diamond (198 questions; Rein et al., 2024). Except for SC@64, every method is evaluated with one greedily decoded sample per problem (temperature 0), and we report accuracy. A single AIME problem is worth 3.3 points, so we treat the five-benchmark average as the primary metric.

Baselines. *Text-only*: the backbone without interaction, and self-consistency over 64 samples (SC@64; Wang et al., 2023b). *Agentic GRPO*: our reimplementation of GRPO from the same warm start, on the same data and environment, with the recipe of Yu et al. (2025b), namely token-level aggregation, clip-higher, overlong shaping, and a clipped tool-use bonus; it equals our method with all three signals removed. *DemyAgent recipe*: our own implementation of the recipe of Yu et al.

Table 1: **Main results** (accuracy, %). Avg. is the unweighted mean over the five benchmarks. Our results are means over three training seeds.

Method	Interact.	AIME24	AIME25	MATH-500	LCB v6	GPQA-D	Avg.
Qwen2.5-7B-Instruct							
Backbone	✗	16.7	13.3	83.2	28.4	38.9	36.1
Backbone + SC@64	✗	36.7	30.0	90.2	33.6	43.4	46.8
Agentic GRPO	✓	43.3	36.7	91.2	38.6	44.9	50.9
DemyAgent recipe	✓	56.7	50.0	93.0	43.2	48.5	58.3
Ours	✓	**63.3**	**56.7**	**95.2**	**47.6**	**52.5**	**63.1**
Qwen3-4B-Instruct-2507							
DemyAgent recipe	✓	53.3	46.7	91.8	40.5	46.0	55.7
Ours	✓	**56.7**	**50.0**	**93.4**	**43.8**	**49.0**	**58.6**

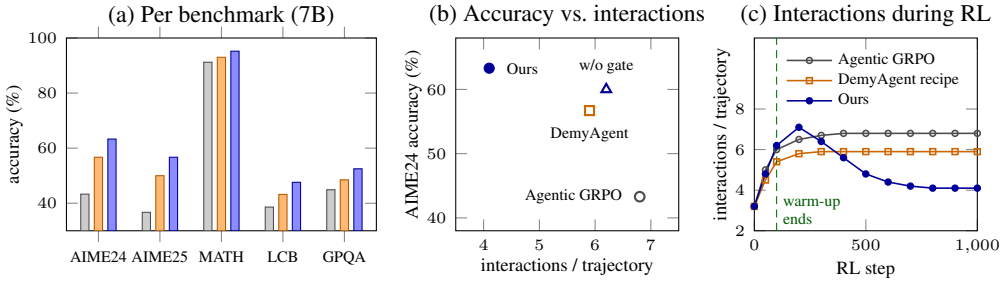


Figure 3: **Results and interaction behavior during training** (7B). (a) Accuracy per benchmark (values in Table 1). (b) AIME 2024 accuracy versus the average number of interactions per trajectory. (c) Interactions per trajectory during RL; the dashed line marks the end of the gate warm-up.

(2025b) with each backbone, run under our evaluation protocol; these rows are not numbers copied from the original paper. Numbers reported by other papers under different protocols are collected in Appendix C.1 for reference; they are not directly comparable.

5.2 Main results

Table 1 reports the results. At 7B, our model reaches a five-benchmark average of 63.1, which is 4.8 points above the DemyAgent recipe at the same scale and 12.1 points above agentic GRPO. The latter comparison is the most controlled, because only the three training signals differ. Gains appear on all five benchmarks (Figure 3a) and are largest on AIME (+6.6 and +6.7 points over the DemyAgent recipe). Text-only test-time scaling does not close the gap: SC@64 reaches an average of 46.8. At 4B, the method raises the average from 55.7 (DemyAgent recipe) to 58.6, so our 4B model matches the DemyAgent recipe at 7B (58.3) with about half the parameters.

5.3 How does the model use feedback?

Fewer interactions, mostly by removing redundant ones. Figure 4a splits the interactions in each AIME 2024 trajectory into productive and redundant ones, labeled as in Section 3. Relative to agentic GRPO, our model makes 40% fewer interactions (6.8 → 4.1) while being more accurate (Figure 3b), and the productive fraction rises from 52.3% to 82.9%. The number of productive interactions per trajectory barely changes (3.6 → 3.4), whereas the number of redundant ones falls from 3.2 to 0.7. The reduction happens in two steps. Without the gate, coherence and credit already cut redundant interactions to 2.0 while adding productive ones (4.2). The gate then removes most of the remaining redundant interactions (2.0 → 0.7) at the cost of some productive ones (4.2 → 3.4), and accuracy still rises (60.0 → 63.3). On AIME 2024 specifically, adding the gate lowers the share of redundant interactions from 31.5% to 17.1%, a 46% relative reduction. Across all five benchmarks, the mean number of redundant interactions per trajectory decreases from 1.81 to 1.07, corresponding to a 40.9% relative reduction, rounded to 41% in the abstract (Table 7).

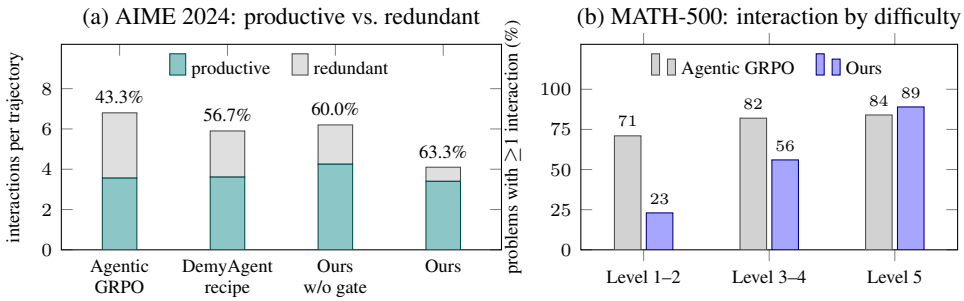


Figure 4: **Interaction behavior.** (a) Interactions per AIME 2024 trajectory, split into productive and redundant ones; accuracy is shown above each bar. (b) Fraction of MATH-500 problems on which the model interacts at least once, by difficulty level.

Table 2: **Ablations** on the 7B model. "ℓ_2 alignment" replaces Equation (2) with a squared distance between raw states. Avg. and Δ are computed over the four benchmarks shown.

Variant	AIME24	MATH-500	LCB v6	GPQA-D	Avg.	Δ
Full method	63.3	95.2	47.6	52.5	64.65	–
w/o counterfactual credit	56.7	92.8	43.0	48.0	60.13	−4.53
w/o coherence constraint	56.7	93.6	44.8	49.5	61.15	−3.50
w/o gate	60.0	93.8	46.2	50.5	62.63	−2.03
w/o coherence and gate	50.0	92.4	41.8	47.0	57.80	−6.85
w/o all three (= agentic GRPO)	43.3	91.2	38.6	44.9	54.50	−10.15
ℓ_2 alignment of raw states	46.7	91.5	39.2	45.4	55.70	−8.95
No gate warm-up	53.3	93.0	44.2	48.6	59.78	−4.88

Feedback-utilization precision. We define feedback-utilization precision (FUP) as the fraction of executed interactions that are labeled productive, $\mathrm{FUP} = N_{\mathrm{productive}}/(N_{\mathrm{productive}} + N_{\mathrm{redundant}})$. Holding counterfactual credit and the gate fixed, removing the coherence constraint decreases FUP from 82.9% to 67.4%. The representational constraint therefore improves feedback-utilization precision by $(82.9 - 67.4)/67.4 = 23.0\%$ relative. This suggests that preserving the active reasoning state across an interaction makes acquired feedback more likely to be used productively in subsequent reasoning.

Interaction is allocated by difficulty. Figure 4b shows how often the model interacts at each MATH-500 difficulty level. Agentic GRPO interacts on 71% of Level 1–2 and 84% of Level 5 problems; our model does so on 23% and 89%: it interacts much less where interaction is rarely needed, and slightly more where it is most useful.

Fewer integration failures. Under the annotation procedure of Section 3, the share of incorrect AIME 2024 trajectories that contain an integration failure falls from 34.5% for agentic GRPO and 28.3% for the DemyAgent recipe to 11.2% for our model; among correct trajectories, it falls from 6.0% and 5.8% to 2.1%. The learned coherence score also rises (from 0.48 to 0.71 on incorrect trajectories), but because this score is the quantity being optimized, we regard it only as a sanity check (Appendix C).

Training dynamics. Figure 3c tracks interactions during RL. In the first 200 steps, all three methods increase their number of interactions. After the gate warm-up ends at step 100, our model's interactions peak at 7.1 (step 200) and then fall to 4.1 per trajectory, whereas the two baselines plateau at 5.9 and 6.8.

5.4 Ablations

Every signal contributes. Removing counterfactual credit, the coherence constraint, or the gate lowers the four-benchmark average by 4.53, 3.50, and 2.03 points, respectively (Table 2); removing all three recovers agentic GRPO (-10.15).

Coherence and gating are complementary but partly redundant. With counterfactual credit kept, the four on/off configurations of the coherence constraint and the gate give an interaction term $I = f_{11} - f_{10} - f_{01} + f_{00} = 64.65 - 62.63 - 61.15 + 57.80 = -1.33$ points (per benchmark: -3.4, $+0.2$, -1.6, -0.5). Each signal still helps when the other is present (+3.50 for coherence, +2.03 for the gate), but their joint gain over using neither (+6.85) is smaller than the sum of their individual gains (+8.18). This agrees with Section 5.3, where both signals suppress unproductive interactions, but through different training signals.

Design choices. Replacing Equation (2) with an ℓ_2 penalty on raw states lowers the average by 8.95 points, more than removing the constraint altogether, which supports the concern of Section 4.1 that aligning raw states suppresses the use of observations. Removing the warm-up costs 4.88 points: a gate trained on early, noisy credit prematurely suppresses exploration.

5.5 Quality of counterfactual credit

We randomly sampled 100 trajectories, and two annotators, blinded to the predicted credit, independently labeled each executed interaction as harmful, neutral/redundant, or helpful (ordinal labels -1, 0, $+1$) according to its effect on subsequent reasoning and the final solution; disagreements were resolved by adjudication. The Spearman correlation between these labels and the predicted interaction credit is 0.73, compared with 0.41 for splitting the outcome uniformly across interactions ($R(\tau)/M$). Because the uniform split is constant within a trajectory, its correlation reflects only differences between trajectories; ranking interactions within a trajectory, and calibrating against executed replay (Zhang, 2026), would be more stringent tests.

6 Limitations

Scope. All experiments use one environment, Python execution; transfer to search, APIs, or multi-tool settings is untested. *Integration versus belief revision.* We measure integration failures but not whether the model correctly abandons a hypothesis when a reliable observation contradicts it. A constraint that preserves the line of reasoning could, in principle, make a model more resistant to such revisions; paired tests with supporting, contradicting, and corrupted observations are needed. *Credit.* Equation (4) is a heuristic that we have not calibrated against executed counterfactual replay, and leave-one-out credit is not additive when interactions complement or duplicate each other (Appendix B.3). Its quality is also bounded by that of the verifier, which may be noisy in domains with weak verification signals; self-improving verifiers (Hosseini et al., 2024) are a natural extension. *Evaluation.* All baselines were run under one protocol with a 20K-token budget; papers that report results under other protocols often report higher numbers (Appendix C.1). We did not compare against recent methods for tool-use decisions (Zhou et al., 2026; Zou et al., 2026) or branch-based credit (Zhao et al., 2026; Wang et al., 2026a) under matched budgets, nor against a GRPO baseline given the roughly 20% extra training compute that our signals require. On AIME, differences of a few points correspond to one or two problems.

7 Conclusion

We argued that reasoning from feedback, i.e., seeking, integrating, and attributing new information, deserves to be a post-training objective in its own right, and instantiated this view with three training signals optimized jointly with GRPO. In a controlled comparison, the resulting models are more accurate, interact less by removing mostly redundant interactions, and make fewer integration failures. Our analysis also delimits what each signal guarantees, which points to the next steps: behavioral tests of belief revision and credit estimates calibrated against executed replay.

Reproducibility statement

Section 4 and Algorithm 1 specify the training objective and procedure; Appendix A lists data composition and hyperparameters; Appendix B states assumptions and gives proofs for all formal statements.

ETHICS STATEMENT

This work studies post-training of language models on mathematics, programming, and science benchmarks. Model-generated code is executed only inside an isolated sandbox with time and memory limits. The trajectory annotations used in our analyses concern model outputs only and involve no personal data; they were produced by two authors of this work. We do not foresee risks specific to this work beyond those of more capable tool-using agents in general; deployment with tools that have real-world side effects requires safeguards beyond the scope of this paper.

AI USE STATEMENT

No artificial intelligence or generative AI tools were used in the preparation of this submission. All aspects of the work, including ideation, method design, implementation, experiment design, data processing, analysis, manuscript writing and revision, figure preparation, calculations, and proof development, were performed by the authors. All experimental results were obtained from the authors' own training and evaluation runs. All references were checked against their original sources. The authors take full responsibility for the content of this paper, including its text, claims, analyses, and artifacts.

REFERENCES

ByteDance Seed Foundation Code Team, Yao Cheng, Jianfeng Chen, Jie Chen, Li Chen, Liyu Chen, Wentao Chen, Zhengyu Chen, Shijie Geng, Aoyan Li, et al. FullStack Bench: Evaluating LLMs as full stack coders. *arXiv preprint arXiv:2412.00535*, 2024.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-Star: Empowering LLM-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*, 2025a.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025b.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. ReTool: Reinforcement learning for strategic tool use in LLMs. *arXiv preprint arXiv:2504.11536*, 2025.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning*, 2023.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations*, 2024.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, 2020.

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Azar, Bilal Piot, Nicolas Heess, Hado van Hasselt, Greg Wayne, Satinder Singh, Doina Precup, and Remi Munos. Hindsight credit assignment. In *Advances in Neural Information Processing Systems*, 2019.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STaR: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. *arXiv preprint arXiv:2410.01679*, 2024.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations*, 2023.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*, 2025a.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. ToRL: Scaling tool-integrated RL. *arXiv preprint arXiv:2503.23383*, 2025b.

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

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In *Proceedings of the International Joint Conference on Natural Language Processing and the Asia-Pacific Chapter of the Association for Computational Linguistics*, 2023.

Renwei Meng. Policy-conditioned counterfactual credit for verifiable reinforcement learning of long-horizon language agents. *arXiv preprint arXiv:2606.05263*, 2026.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. ART: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*, 2023.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*, 2023.

Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. SMART: Self-aware agent for tool overuse mitigation. *arXiv preprint arXiv:2502.11435*, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*, 2023.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *Conference on Language Modeling*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Ning Shang, Yifei Liu, Yi Zhu, Li Lyna Zhang, Weijiang Xu, Xinyu Guan, Buze Zhang, Bingcheng Dong, Xudong Zhou, Bowen Zhang, Ying Xin, Ziming Miao, Scarlett Li, Fan Yang, and Mao Yang. rStar2-Agent: Agentic reasoning technical report. *arXiv preprint arXiv:2508.20722*, 2025.

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

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the search capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

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

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. Acting less is reasoning more! Teaching model to act efficiently. *arXiv preprint arXiv:2504.14870*, 2025.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. MathCoder: Seamless code integration in LLMs for enhanced mathematical reasoning. *arXiv preprint arXiv:2310.03731*, 2023a.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2024.

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. In *International Conference on Learning Representations*, 2023b.

Ying Wang, Changlin Qiu, Bang Lin, Linbo Jin, Wen Jiang, Zhe Sun, and Jingli Yang. Contrastive branch policy optimization. *arXiv preprint arXiv:2608.24300*, 2026a.

Yinjie Wang, Tianbao Xie, Ke Shen, Mengdi Wang, and Ling Yang. RLAnything: Forge environment, policy, and reward model in completely dynamic RL system. *arXiv preprint arXiv:2602.02488*, 2026b.

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

Yiwei Yang, Haoxiang Zhang, Bingbing Wen, Yao Lu, Yuchen Wu, Lei Zhang, Julian McAuley, Pan Lu, and Bill Howe. Spurious tool use: When RL agents learn the wrong reason to act. *arXiv preprint arXiv:2609.16268*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025a.

Zhaochen Yu, Ling Yang, Jiaru Zou, Shuicheng Yan, and Mengdi Wang. Demystifying reinforcement learning in agentic reasoning. *arXiv preprint arXiv:2510.11701*, 2025b.

Haiyue Zhang. Credit without ground truth: Auditing step-level credit assignment in LLM agents against executed replay. *arXiv preprint arXiv:2608.19760*, 2026.

Yubao Zhao, Weiquan Huang, Sudong Wang, Ruochen Zhao, Chen Chen, Yao Shu, and Chengwei Qin. BranPO: Scalable contrastive branch sampling for long-horizon agentic reinforcement learning. *arXiv preprint arXiv:2602.03719*, 2026.

Chaoyang Zhou, Shunyu Liu, Zengmao Wang, Di Wang, Rong-Cheng Tu, Bo Du, and Dacheng Tao. Intra-trajectory consistency for reward modeling. *arXiv preprint arXiv:2506.09096*, 2025.

Yijin Zhou, Linqian Zeng, Xiaoya Lu, Wenyuan Xie, Dongrui Liu, Junchi Yan, and Jing Shao. Exploring agentic tool-calling decisions via uncertainty-aligned reinforcement learning. *arXiv preprint arXiv:2606.06976*, 2026.

Wenhao Zou, Xianglong Liu, Wendong Bi, Hanjie Wang, Simin Zhao, and Gong Zhi. CoBRA: Learning tool-use boundaries via counterfactual margins. *arXiv preprint arXiv:2609.00967*, 2026.

A Implementation Details

Algorithm 1 Post-training with coherence, counterfactual credit, and selective seeking

Require: warm-started policy π_θ with heads $P^{\pm}$; gate MLP_ϕ; verifier V_ψ; prompts $\mathcal{D}$; environment $\mathcal{E}$; group size G

1: **for** each RL iteration **do**
2: sample a batch of problems $\{x_i\} \subset \mathcal{D}$
3: **for** each x_i and $n = 1, \ldots, G$ **do**
4: generate $\tau_i^{(n)}$ with π_θ; at each request q_m: compute g_m (Equation (5)), sample b_m (b_m=1 during warm-up), execute q_m in $\mathcal{E}$ iff b_m=1; record $\boldsymbol{h}_m^{\pm}$
5: **end for**
6: compute $R(\tau)$, credit $\hat{c}_m$ (Equation (4)), gate rewards (Equation (6)), segment rewards r_m, and group-normalized advantages (Section 4.4)
7: update θ and $P^{\pm}$ with $\mathcal{L}_{\mathrm{PG}} + \lambda_{\mathrm{coh}}\mathcal{L}_{\mathrm{coh}}$; update ϕ with $\lambda_{\mathrm{gate}}\mathcal{L}_{\mathrm{gate}}$ (after warm-up); update the threshold μ
8: **end for**

Data. The 3K supervised demonstrations comprise 1K mathematics problems solved with the interpreter (sources: AIME, AMC, MATH), 1K code-generation problems with execution feedback (CodeContests), and 1K science problems with computational checks (GPQA, SciQ). The 30K RL prompts comprise 10K mathematics problems (MATH, historical AIME, competition mathematics), 10K code problems (LiveCodeBench, CodeContests), and 10K mixed problems (GPQA, physics, and chemistry computation), formatted with the agent instruction templates of Yu et al. (2025b). Before constructing the SFT, RL, verifier, and validation sets, we excluded all evaluation instances. Specifically, we removed AIME 2024/2025 problems, all MATH-500 problem IDs, GPQA-Diamond examples, and LiveCodeBench problems within the evaluation window. We additionally canonicalized problem text and removed exact duplicates across training and evaluation sets.

Verifier. V_ψ is initialized from the supervised checkpoint. For each partial trajectory in its training set, we sample 32 continuations and use the fraction that reach a correct answer as the regression target; it is trained with a squared loss for 2 epochs at learning rate 5×10^{-6}.

Gate threshold. The threshold μ is initialized to the median gate score over 1K rollouts, updated as an exponential moving average with decay 0.99 during training, and frozen at inference.

Table 3: Hyperparameters.

SFT epochs / learning rate	2 / 2×10^{-5}	coherence weight λ_{coh}	0.1
RL learning rate	10^{-6}	contrastive temperature κ	0.07
prompts per batch / group size G	64 / 16	credit weight η	0.3
max. response length / interactions	20,480 / 16	gate loss weight λ_{gate}	0.05
clipping ϵ_{low} / ϵ_{high}	0.2 / 0.28	gate reward scale λ_g	0.1
overlong buffer	4,096 tokens	gate sharpness β	5.0
KL penalty	none	gate warm-up	first 10% of RL steps
sandbox timeout / memory	30 s / 2 GB	verifier MC continuations	32

B Analysis of the Training Signals

B.1 Coherence constraint

Degenerate cases. For $M = 1$, the numerator and denominator of Equation (2) coincide, so the loss and its gradient vanish. For $M = 0$ the term is undefined and is skipped.

Identifiability does not imply use of the observation. Suppose the heads map every pre- and post-interaction state of the m-th interaction to the same vector e_m, where $\{e_m\}$ are orthonormal and encode only the index m. Then every positive pair has similarity 1 and every negative pair similarity 0, and

$$\mathcal{L}_{\mathrm{coh}} = \log\left(1 + (M-1)e^{-1/\kappa}\right), \tag{8}$$

which is below 10^{-5} for $\kappa = 0.07$ and $M \leq 16$, regardless of what o_m contains. The constraint therefore rules out drifting away from the request, but not ignoring the observation; this is why we assess integration behaviorally (Section 5.3) and list belief revision as an open question (Section 6).

B.2 What the gate learns

Proposition 1 (Gate optimum). *Fix a context and let $\rho \in [0,1]$ be the probability that the gate executes the request. Let $p = \Pr(\hat{c}_m > 0 \mid \text{context, executed})$. Under the reward of Equation (6), the expected gate reward is $J(\rho) = \rho\,\lambda_g(2p-1)$. Hence J is maximized by $\rho = 1$ if $p > 1/2$, by $\rho = 0$ if $p < 1/2$, and by any ρ if $p = 1/2$.*

Proof. Executing yields $+\lambda_g$ with probability p and $-\lambda_g$ with probability $1-p$; not executing yields 0. Thus $J(\rho) = \rho(\lambda_g p - \lambda_g(1-p)) + (1-\rho)\cdot 0 = \rho\lambda_g(2p-1)$, which is linear in ρ. □

Remark 1. *If $\hat{c}_m$ has a continuous distribution, $p > 1/2$ is equivalent to a positive conditional median. It is not equivalent to a positive conditional mean: with $\hat{c}_m = +0.1$ w.p. 0.6 and -1 w.p. 0.4, we have $p = 0.6$ but $\mathbb{E}[\hat{c}_m] = -0.34$. A reward proportional to $\hat{c}_m$ would target the mean instead.*

Remark 2. *Proposition 1 characterizes the target of the gate, not convergence to it. The training-time gate $\sigma(\beta(g_m - \mu))$ cannot reach probabilities of exactly 0 or 1, the gate is trained jointly with a changing policy and credit estimator, and stochastic gradient descent on an MLP carries no global-optimality guarantee. Since a closed gate receives zero reward, its gradient comes only from executed requests; the warm-up ensures that these cover the full range of contexts early in training.*

B.3 Counterfactual credit

Remark 3 (Unbiasedness in the idealized case). *If $\mathbb{E}[V_\psi(\tau) \mid \tau] = V^\star(\tau)$ and $\mathbb{E}[V_\psi(\tau^{\neg m}) \mid \tau_{<m}] = V^\star(\tau^{\neg m})$, then $\mathbb{E}[V_\psi(\tau) - V_\psi(\tau^{\neg m})] = c^\star_m$, by linearity of expectation, regardless of how the errors of the two terms are correlated.*

The practical estimator is not covered. Equation (4) replaces $V_\psi(\tau^{\neg m})$ with a combination of the prefix value $V_\psi(\tau_{<m})$ and a weighted value of the *factual* continuation $\tau_{>m}$, which may already reflect o_m. No identity equates this combination with $V^\star(\tau^{\neg m})$, and it need not lie in $[0,1]$. We therefore treat Equation (4) as a shaping heuristic whose usefulness is an empirical question (Sections 5.4 and 5.5).

Leave-one-out credit is not a decomposition of the outcome. If two interactions are jointly necessary, removing either one makes success impossible, and each receives credit 1 although the trajectory's success probability is 1; if two interactions are fully redundant, each receives credit 0 although the trajectory succeeds. Credit is therefore a shaping signal rather than an allocation of $R(\tau)$, and it estimates a different quantity than the uniform split $R(\tau)/M$; we make no claim that it reduces the variance of the policy gradient.

C Additional Results

Training stages. Table 4 reports AIME 2024 accuracy after each training stage. The two backbones belong to different model families, so the table should not be read as a scaling trend; same-family scaling experiments are left to future work.

Table 4: AIME 2024 accuracy (%) after each training stage.

Backbone	No training	SFT	SFT → agentic GRPO	SFT → ours
Qwen3-4B-Instruct-2507	10.0	30.0	40.0	56.7
Qwen2.5-7B-Instruct	16.7	36.7	43.3	63.3

Sensitivity. Table 5 varies one hyperparameter at a time around the default configuration. The coherence weight λ_{coh} matters most: small values give too little constraint, while $\lambda_{\text{coh}} = 0.3$ makes the model interact least (3.2 interactions) and most precisely (86.1% productive) but lowers

accuracy, consistent with over-constraining exploration. Results change little for credit weights $\eta \in [0.3, 0.5]$ and drop slightly at $\eta = 0.1$; doubling the group size from 16 to 32 changes the results only marginally.

Table 5: Sensitivity on AIME 2024 (accuracy, interactions per trajectory, productive fraction) and MATH-500 (accuracy). Default values are marked with *.

Parameter	Value	AIME24	MATH-500	Interactions	Productive (%)
λ_{coh}	0.01	56.7	92.8	5.8	64.2
	0.05	60.0	94.2	4.8	74.5
	0.10*	63.3	95.2	4.1	82.9
	0.30	60.0	94.0	3.2	86.1
η	0.1	60.0	94.4	4.4	78.2
	0.3*	63.3	95.2	4.1	82.9
	0.5	63.3	94.8	4.0	83.6
G	8	56.7	93.6	4.6	76.8
	16*	63.3	95.2	4.1	82.9
	32	63.3	95.4	4.0	83.2

Interaction efficiency. Table 6 gives the AIME 2024 numbers behind Figure 4a and Figure 3b.

Table 6: Interaction efficiency on AIME 2024.

Method	Interactions / trajectory	Productive (%)	Redundant (%)	Accuracy
Agentic GRPO	6.8	52.3	47.7	43.3
DemyAgent recipe	5.9	61.2	38.8	56.7
Ours w/o gate	6.2	68.5	31.5	60.0
Ours	4.1	82.9	17.1	63.3

Redundant interactions across benchmarks. Table 7 reports the per-benchmark values underlying the aggregate reduction cited in Section 5.3.

Table 7: Mean number of redundant interactions per trajectory with and without the gate. The macro average gives the 40.9% relative reduction reported as 41% in the abstract.

Benchmark	w/o gate	Full
AIME 2024	1.95	0.70
AIME 2025	1.80	0.82
MATH-500	1.40	1.00
LiveCodeBench v6	2.60	1.75
GPQA-D	1.30	1.07
Macro avg.	1.81	1.07

Integration failures and learned coherence. Table 8 gives the full numbers behind Section 5.3. The coherence score is the mean cosine similarity between $\boldsymbol{z}_m^-$ and $\boldsymbol{z}_m^+$, i.e., the quantity that Equation (2) optimizes, so it is reported only as a sanity check.

Interaction by difficulty. Table 9 reports the MATH-500 interaction rates plotted in Figure 4b.

C.1 Results reported in prior work

Table 10 lists results reported by other papers. They use different evaluation protocols (e.g., number of samples per problem, generation budget, tool interface) and, in several cases, different or larger backbones, so they are not comparable to Table 1. We include them for transparency; several exceed our numbers. In particular, the released DemyAgent-4B is trained from the authors' own SFT checkpoint (Qwen3-4B-RA-SFT) with a larger upper clipping ratio (0.315) and is evaluated as average@32 at temperature 1.0 and top-p 0.6 (Yu et al., 2025b), whereas the DemyAgent-recipe

Table 8: Integration failures on AIME 2024 (share of trajectories with at least one failure, %) and learned coherence score, split by correctness.

	Integration failures (%)		Coherence score	
Method	correct	incorrect	correct	incorrect
Agentic GRPO	6.0	34.5	0.72	0.48
DemyAgent recipe	5.8	28.3	0.75	0.52
Ours	2.1	11.2	0.89	0.71

Table 9: MATH-500 interaction rate by difficulty level: share of problems with at least one interaction (%).

Level	Agentic GRPO	Ours
1–2	71	23
3–4	82	56
5	84	89

rows in Table 1 are our matched-budget runs; the two sets of numbers are therefore not directly comparable.

Table 10: Results reported in prior work (accuracy, %). Rows marked [a] are copied from Table 2 of Yu et al. (2025b); rows marked [b] from Table 1 of Feng et al. (2025). "–": not reported.

Method	Params	AIME24	AIME25	GPQA-D	LCB v6
ToRL (Li et al., 2025b)[a]	7B	43.3	30.0	–	–
ARPO (Dong et al., 2025b)[a]	7B	30.0	30.0	53.0	18.3
DemyAgent-4B (Yu et al., 2025b)[a]	4B	72.6	70.0	58.5	26.8
rStar2-Agent (Shang et al., 2025)[a]	14B	80.6	69.8	60.9	–
ReTool, Qwen2.5-32B-Instruct (Feng et al., 2025)[b]	32B	67.0	49.3	–	–
ReTool, R1-Distill-Qwen-32B (Feng et al., 2025)[b]	32B	72.5	54.3	–	–