MulFeRL: Enhancing Reinforcement Learning with Verbal Feedback in a Multi-turn Loop

Anonymous ACL submission

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) is widely used to improve reasoning across domains, but outcome-only scalar rewards are often sparse and uninformative, especially for failed samples, where they indicate only failure without explaining why the reasoning breaks down. In this paper, we leverage richer verbal feedback to guide RLVR on failed samples and convert feedback-induced progress into trainable learning signals. We propose **MulFeRL** (**Mul**ti-turn **Fe**edback-guided **R**einforcement **L**earning), a multi-turn, event-triggered RLVR framework that combines *progress induction* for feedback-guided regeneration of failed samples, *progress credit assignment* for learning from verifier-confirmed progress, and *structured feedback injection* for integrating feedback into the model's reasoning process. Trained on sampled OpenR1-Math, MulFeRL outperforms supervised, self-distillation-based, and RLVR baselines in-domain and shows strong out-of-domain generalization. [1]

1 Introduction

Reinforcement Learning with Verifiable Rewards (RLVR) has become a popular approach for improving reasoning capabilities in domains such as mathematics and code generation (Shao et al., 2024; Cobbe et al., 2021). A verifier assigns each generated answer a scalar reward, e.g., correct or incorrect, which is used to optimize the policy (DeepSeek-AI, 2025). However, as model performance increases, training that relies solely on final outcome correctness yields diminishing gains and eventually struggles to further improve the model (Zhang et al., 2025; Yu et al., 2025).

We argue that this bottleneck stems from the sparsity and uninformative content of scalar rewards (Ye et al., 2025; Wang et al., 2025a). In early training stages or on harder problems, models often generate a large fraction of entirely incorrect solutions; under standard RLVR, failed solutions typically receive near-zero or negative rewards that merely indicate incorrectness, but provide little information about why the reasoning failed or how to improve it (Cai et al., 2025; Lightman et al., 2024). Consequently, the resulting learning signal suffers from poor credit assignment: gradient estimates are noisy, training becomes sample-inefficient, and progress slows or even plateaus, ultimately limiting final performance (Zheng et al., 2025).

In contrast, when errors occur, humans typically leverage verbal feedback to localize mistakes and obtain fine-grained guidance for closing the gap to the target solution (Hattie and Timperley, 2007; Kluger and DeNisi, 1996). Inspired by this, we incorporate verbal feedback into model training: beyond indicating whether an answer is correct, natural language can identify key errors, missing steps, and actionable suggestions, providing a more informative, fine-grained learning signal for unsuccessful trajectories. This enables the model to learn from failed trajectories that would otherwise be treated as uniformly uninformative under outcome-only rewards.

Recent work has explored verbal feedback as richer supervision beyond sparse outcome-only rewards. Directly optimizing verbal feedback as scalar rewards or feedback-revised supervision targets (Kim et al., 2024; Liu et al., 2024) can compress its fine-grained diagnostics into coarse reward labels or imitation targets. Self-distillation methods use feedback-conditioned or privileged policies as dense teachers (Hübotter et al., 2026; Zhao et al., 2026), but matching a privileged teacher distribution may over-compress uncertainty and self-correction traces that are important for robust reasoning (Kim et al., 2026). RL-style feedback methods inject critiques or process guidance into policy optimization (Zhang et al., 2025; Song et al., 2026),

[1] We release our code at https://anonymous.4open.science/r/MulFeRL.

showing that feedback can improve refinement and exploration. However, they typically allocate feedback broadly across samples rather than targeting failed samples, where outcome-only rewards provide no usable contrast. Since samples with a successful trajectory are already outcome-informative, broad feedback allocation can under-exploit failed samples, where feedback is most needed.

Building on these observations, we focus on a central question: **how can verbal feedback be transformed into a stable learning signal that continues to drive RL training progress on failed samples?** Our key idea is that, during training, when a sample receives uniformly zero scalar reward and thus provides no informative learning signal, we introduce verbal feedback to guide regeneration and measure the resulting feedback-conditioned improvements, using them to construct an optimizable training objective, thereby converting verbal feedback into gradient signals.

We instantiate this idea in **MulFeRL**, a multi-turn, event-triggered RLVR framework with two stages: *progress induction* and *progress credit assignment*. In the progress induction stage, MulFeRL targets uniformly failed samples, using triggered feedback as a conditioning signal for iterative regeneration. This converts failure cases that originally lack reward contrast into verifier-certified progress. In the progress credit assignment stage, MulFeRL extracts learning signal from the contrast exposed by such progress. It uses GRPO to exploit within-group contrast after regeneration, and introduces Feedback-Contrastive Optimization (FCO) to recover cross-state signal from all-failed to all-successful transitions. MulFeRL further uses *structured feedback injection* to integrate feedback into the model's reasoning process, improving its ability to use feedback during regeneration and learn from feedback-induced progress.

We train MulFeRL on a sampled subset of OpenR1-Math (Yan et al., 2026) and evaluate it on math benchmarks and OOD science/general reasoning tasks. MulFeRL consistently outperforms supervised, self-distillation-based, and RL-based baselines, with further analyses probing the roles of feedback-induced progress and credit assignment.

2 Related Work

Enhancing LLM Reasoning with Reinforcement Learning. In verifiable domains such as mathematics and code, RLVR trains LLMs with rule-based outcome signals through online policy optimization, commonly using PPO-style methods and GRPO variants for efficient reasoning post-training (Schulman et al., 2017; Shao et al., 2024; Liu et al., 2025; Yu et al., 2025). Recent work further extends RLVR-style training beyond single-turn reasoning to multi-turn and environment-interactive agents, incorporating environment feedback and turn-level credit assignment (Wang et al., 2025b; Zeng et al., 2025). Despite these advances, RLVR remains constrained by sparse scalar rewards: failed samples receive little guidance on what went wrong or how to revise, often leading to training plateaus (Uesato et al., 2022; Liu et al., 2026). We address this limitation by injecting verbal feedback during training, using it to guide regeneration and expose richer learning signals from failed samples.

Learning from Verbal Feedback. Recent work has explored verbal feedback as richer supervision beyond sparse outcome-only rewards. Early methods convert feedback into scalar rewards or revised supervised targets (Kim et al., 2024; Hancock et al., 2019; Liu et al., 2024), which eases optimization but collapses fine-grained diagnostics into coarse labels or imitation targets; self-distillation methods instead use feedback-conditioned or privileged policies as dense teachers (Hübotter et al., 2026; Zhao et al., 2026), but teacher-distribution matching may over-compress uncertainty and self-correction traces important for robust reasoning (Kim et al., 2026). RL-style feedback methods inject critiques or process guidance into policy optimization (Zhang et al., 2025; Song et al., 2026), but typically allocate feedback broadly rather than targeting uniformly failed RLVR samples, where outcome rewards provide no usable contrast. Since samples with successful trajectories are already outcome-informative, such broad allocation can under-exploit the all-fail cases where feedback is most needed. MulFeRL instead targets these cases and turns feedback-induced, verifier-certified progress into reward-grounded learning signals through multi-turn regeneration.

3 Method

Notation. Throughout this section, $x \sim \mathcal{D}$ denotes an input prompt, π_θ denotes the trainable policy, π_{old} denotes the frozen behavior policy for rollout collection, π_{ref} denotes the reference policy, and $\mathcal{V}$ denotes the verifier. For each outer update, we freeze the current policy as π_{old} and optimize

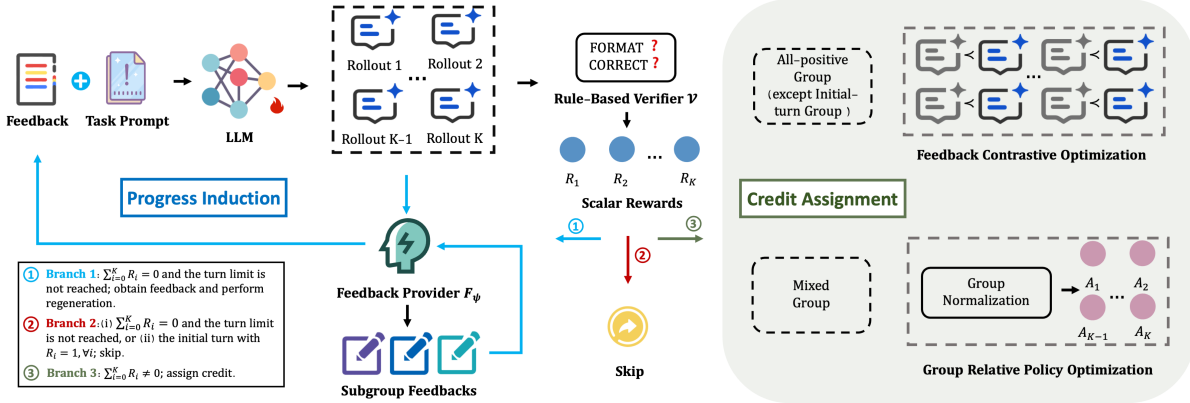


Figure 1: Illustration of MulFeRL.

π_θ with clipped importance ratios $\pi_\theta/\pi_{\text{old}}$. At turn t, π_{old} samples a rollout group $\mathcal{G}^{(t)} = \{y_i^{(t)}\}_{i=1}^K$ under context $c^{(t)}$, where $c^{(0)} = x$ and $y_i^{(t)} \sim \pi_{\text{old}}(\cdot \mid c^{(t)})$. We write $r_i^{(t)} = R_{\mathcal{V}}(x, y_i^{(t)}) \in \{0, 1\}$ for the verifier reward. We denote the feedback provider by F_ψ, which produces feedback $f^{(t)}$ at turn t and can be instantiated as a human user, a user-simulator model, or the policy model itself.

3.1 Problem Formulation

In GRPO-style RLVR, the frozen behavior policy collects K rollouts for each prompt, and a verifier assigns binary rewards indicating whether each rollout is correct. GRPO updates rely on within-group relative advantages computed from these verifier rewards. A key difficulty arises for failed samples, where all K rollouts fail verification. In this case, the verifier provides no discrimination among candidates, causing the within-group advantage signal to collapse and providing no policy-gradient learning signal, which may lead to training plateaus. Inspired by how humans use verbal feedback to localize errors and guide corrections, we introduce verbal feedback into RLVR to provide learning signal on failed samples, where the scalar verifier offers little discrimination. This raises a core question: **How can we convert verbal feedback into a stable learning signal that continues to improve the policy on failed samples?** Formally, our goal is to design a feedback-derived surrogate objective $\mathcal{J}_\theta$ that provides informative gradients and supports stable optimization:

$$\max_\theta \; \mathbb{E}_{x\sim\mathcal{D}} \, \mathbb{E}_{\mathcal{T}(x;\pi_{\text{old}})} \Big[\mathcal{J}_\theta\big(x, \mathcal{T}(x;\pi_{\text{old}})\big) \Big], \qquad (1)$$

where $\mathcal{T}(x;\pi_{\text{old}})$ denotes the collected multi-turn feedback trajectory.

3.2 Overview of MulFeRL

To tackle the low-signal regime in RLVR, we propose **MulFeRL** (**Mul**ti-turn **F**eedback-guided **R**einforcement **L**earning), an event-triggered RLVR framework that converts uniform-reward failures into verifier-certified progress events.

MulFeRL has two stages: *progress induction* and *progress credit assignment*. In the first stage, verbal feedback is used only as a conditioning intervention to help all-failed rollout groups escape the zero-reward state. In the second stage, the policy is updated only when the verifier exposes a usable discrimination signal. Thus, feedback changes the sampling condition, while learning remains gated by verifier-confirmed progress.

For a rollout group $\mathcal{G}^{(t)} = \{y_i^{(t)}\}_{i=1}^K$ under context $c^{(t)}$, with rewards $r_i^{(t)} = R_{\mathcal{V}}(x, y_i^{(t)})$, we define its reward state as:

$$z(\mathcal{G}^{(t)}) = \begin{cases} \textsc{Solved}, & \sum_{i=1}^K r_i^{(t)} = K, \\ \textsc{Contrastive}, & 0 < \sum_{i=1}^K r_i^{(t)} < K, \\ \textsc{Failed}, & \sum_{i=1}^K r_i^{(t)} = 0. \end{cases} \qquad (2)$$

The pipeline is organized around the location of verifier-exposed discrimination. For each prompt, MulFeRL first samples a feedback-free group under $c^{(0)} = x$ and verifies its reward state. If the group is CONTRASTIVE, the verifier already provides discrimination within the current state, and MulFeRL performs within-state credit assignment. If the group is SOLVED, no update is needed because the prompt is already solved without feedback. If the group is FAILED, MulFeRL triggers feedback-guided regeneration until the verifier exposes progress or the turn budget is exhausted.

Once a failed group escapes the all-zero state, MulFeRL assigns credit according to where the

new discrimination arises. A CONTRASTIVE regenerated group exposes within-state discrimination and is optimized with GRPO. A SOLVED regenerated group has no within-group contrast, but the adjacent FAILED→SOLVED transition exposes cross-state discrimination; MulFeRL therefore applies Feedback-Contrastive Optimization (FCO) to credit the transition. If all turns remain FAILED, no update is applied.

3.3 Event-Triggered Progress Induction

The key challenge in using verbal feedback for RL training is that feedback itself is not an optimizable learning signal. It is not a verifier reward and thus should not be optimized directly. Our key intuition is that effective verbal feedback should guide the model to turn an incorrect attempt into a correct one, so that the task improvements induced by feedback can be observed and used as an optimizable signal. Motivated by this intuition, the first stage of MulFeRL is *progress induction*, which uses verbal feedback as a conditioning intervention to expose verifier-certified progress from failed rollout groups. Accordingly, MulFeRL uses feedback only when the current rollout group is in the FAILED state, and treats a transition from an all-failed condition to either a CONTRASTIVE or SOLVED group as induced progress. At a failed turn t, MulFeRL obtains feedback from the feedback provider:

$$f^{(t)} \sim F_\psi\Big(\cdot \,\Big|\, x,\ \mathcal{G}^{(t)},\ \{r_i^{(t)}\}_{i=1}^{K}\Big)\,. \tag{3}$$

The feedback is a group-level, actionable summary of the current failures. It identifies the dominant failure mode and provides concrete guidance for revision[2]. We then inject the feedback into the next-turn context, $c^{(t+1)} = \mathrm{Inject}(x, f^{(t)})$, and regenerate K rollouts with $y_i^{(t+1)} \sim \pi_{\mathrm{old}}(\cdot \mid c^{(t+1)})$.

The regenerated group is verified again. If a FAILED group becomes CONTRASTIVE or SOLVED, MulFeRL treats it as induced progress and proceeds to credit assignment; otherwise, the loop continues until the turn budget is reached. By treating verbal feedback as an intervention measured by reward-state changes rather than as a direct optimization target, MulFeRL focuses extra sampling on low-signal prompts and converts feedback-induced progress into stable policy updates.

[2]Since rollouts are typically long, we use a two-step aggregation: generate g subgroup feedbacks and then summarize them into a single feedback (details in § A).

3.4 Progress Credit Assignment

Progress induction creates feedback-guided trajectories, but it does not by itself specify what should be credited. Since verbal feedback is used only as a conditioning intervention, MulFeRL treats credit assignment as an identifiability problem: a policy update is applied only when the verifier exposes a reward contrast that can be attributed to either candidates within the same state or progress across adjacent states. This view yields a unified principle: feedback may change the sampling condition, but credit is assigned only through verifier-certified discrimination. This principle leads to two complementary objectives: GRPO for within-state discrimination, and FCO for cross-state discrimination that appears only across a FAILED→SOLVED transition.

Within-state credit assignment with GRPO. For a CONTRASTIVE group $\mathcal{G}^{(t)}$, verifier outcomes distinguish successful and failed rollouts under the same conditioning context. Since all candidates share the same prompt and feedback condition, relative rewards can be attributed to response quality rather than context changes. MulFeRL therefore applies the standard clipped GRPO objective:

$$\begin{aligned}\mathcal{L}_{\mathrm{GRPO}}(x;\mathcal{G}^{(t)}) = -\frac{1}{K}\sum_{i=1}^{K}\min\Big(&\rho_i^{(t)}A_i^{(t)},\\ &\mathrm{clip}(\rho_i^{(t)}, 1-\epsilon, 1+\epsilon)A_i^{(t)}\Big)\\ &+\beta_{\mathrm{KL}}\mathrm{KL}\Big(\pi_\theta(\cdot \mid c^{(t)})\Big\|\pi_{\mathrm{ref}}(\cdot \mid c^{(t)})\Big)\,.\end{aligned} \tag{4}$$

where $A_i^{(t)}$ is the group-normalized verifier reward and $\rho_i^{(t)} = \pi_\theta(y_i^{(t)} \mid c^{(t)})/\pi_{\mathrm{old}}(y_i^{(t)} \mid c^{(t)})$. Thus, verifier-successful rollouts receive positive relative credit, while verifier-failed rollouts receive negative relative credit.

Cross-state credit assignment with FCO. However, feedback does not always expose within-state discrimination. In some cases, it is sufficiently effective that the regenerated group becomes SOLVED, i.e., all current rollouts are verifier-positive. In this case, all rollouts receive reward 1, so within-group advantages collapse. The learning signal is not absent, but located across adjacent reward states: before the intervention, the same prompt produced only verifier failures; after the intervention, it produces only verifier successes. FCO is designed to credit this cross-state contrast rather than discard it as a reward-saturated group.

Fixed feedback slot

```
<thinking>
<feedback> {verbal feedback} </feedback>
rest of reasoning
</thinking>  solution
```

Figure 2: Fixed feedback slot used by MulFeRL.

Let $\widehat{\mathcal{Q}}_t$ be K sampled cross-state pairs (y^+, y^-), where y^+ comes from the current SOLVED group and y^- from the immediately preceding FAILED group. We canonicalize rollouts by removing injected feedback tokens from the likelihood objective, but not from the conditioning context used for feedback-guided regeneration. We randomly permute the failed rollouts when forming pairs, so no same-index correspondence is assumed[3]. Define the length-normalized log-ratio score $s_\theta(y; c) = |y|^{-1} \log \frac{\pi_\theta(y|c)}{\pi_{\text{ref}}(y|c)}$. The FCO loss is:

$$\mathcal{L}_{\text{FCO}}^{(t)} = -\frac{1}{|\widehat{\mathcal{Q}}_t|} \sum_{(y^+, y^-) \in \widehat{\mathcal{Q}}_t} \log \sigma\Big(\tau\big[s_\theta(y^+; c^{(t)}) - s_\theta(y^-; c^{(t-1)}) - \gamma\big]\Big). \tag{5}$$

where τ is an inverse temperature and $\gamma \geq 0$ is an optional margin.

Unified routing. Let $t^\star = \min\{t : z(\mathcal{G}^{(t)}) \neq \text{FAILED}\}$ if the set is nonempty; otherwise no update is applied. For $\mathcal{G}^\star = \mathcal{G}^{(t^\star)}$ and $z^\star = z(\mathcal{G}^\star)$, MulFeRL uses

$$\mathcal{L}(x) = \begin{cases} \mathcal{L}_{\text{GRPO}}(x; \mathcal{G}^\star), & z^\star = \text{CONTRASTIVE}, \\ \lambda\, \mathcal{L}_{\text{FCO}}^{(t^\star)}, & t^\star > 0,\ z^\star = \text{SOLVED}, \\ 0, & \text{otherwise}. \end{cases} \tag{6}$$

Thus, GRPO is used whenever the verifier exposes within-state contrast, while FCO is used only for feedback-induced FAILED→SOLVED transitions.

3.5 Structured Feedback Injection

Inspired by passage injection in retrieval-augmented generation (Tang et al., 2025), MulFeRL injects verbal feedback into a fixed `<feedback>` slot in the reasoning template, as shown in Figure 2, rather than appending it to the prompt. The slot is empty at the feedback-free turn and is filled with feedback from F_ψ at regeneration turns. This fixed-slot design treats feedback as an explicit conditioning intervention, embeds it more tightly into the model's reasoning context, and keeps injected tokens identifiable for masking and canonicalization. During training, externally injected feedback tokens are masked out; a rollout is assigned reward 1 only when both answer correctness and schema compliance hold. We prompt the feedback simulator to produce concise *Issue* and *Fix steps* fields.[4]

4 Experiment

4.1 Experiment Settings

Datasets. For training, we randomly sample 4k instances from the reorganized 45k subset of OpenR1-Math-220k following Yan et al. (2026), without using any examples, labels, or solutions from the evaluation benchmarks. We evaluate on five math benchmarks (AMC 2023 (Li et al., 2024), AIME 2024 (Li et al., 2024), OlympiadBench (He et al., 2024), MATH-500 (Hendrycks et al., 2021) and Minerva-Math (Lewkowycz et al., 2022)), spanning routine to competition-level problems. For broader generalization, we additionally test on MMLU-Pro (Wang et al., 2024), GPQA-Diamond (Rein et al., 2023) and TheoremQA (Chen et al., 2023). We report pass@1 averaged over 5 independent runs per dataset.[5]

Baselines. We consider four groups of baselines: (1) Base model: the backbone without any training; (2) Supervised finetuning baselines: *SFT* finetunes on the training set using supervised learning, *RAFT* (Dong et al., 2023) finetunes on self-generated correct responses selected by rule-based verification, and *CITL-FT* (Xi et al., 2024) finetunes on self-generated and critique-refined responses; (3) Self-distillation baselines: *SDPO* (Hübotter et al., 2026) distills feedback-conditioned rollouts from the current policy into the no-feedback policy; (4) RL-based finetuning baselines: *GRPO* (DeepSeek-AI, 2025) optimizes binary outcome rewards; *Dr.GRPO* (Liu et al., 2025) and *Critique-GRPO* (Zhang et al., 2025) are GRPO variants that respectively remove biased terms and use critiques for refinement; and *RLTF-FM* (Song et al., 2026) trains with multi-turn textual feedback and an auxiliary feedback-prediction loss, thereby internalizing feedback-driven improvements into the test-time first-turn policy.

Implementation Details. We use Qwen2.5-7B-Base (Yang et al., 2024) and Qwen3-4B-Instruct-

[3]The unbiasedness of this permutation estimator is proved in § B.8.

[4]§ A provides algorithmic details, and § B provides the theoretical analysis.

[5]Dataset details and access links are provided in § E.

Method	Math (ID)					Science & General (OOD)			Avg.
	AMC23	AIME24	Olympiad Bench	MATH 500	Minerva MATH	MMLU Pro	GPQA Diamond	Theorem QA	
Base Model									
Qwen2.5-7B-Base	34.50	13.33	28.13	55.24	18.31	45.06	27.68	19.00	30.16
Supervised Learning-based Finetuning									
SFT	38.50	12.00	27.27	57.12	21.03	47.20	28.08	24.10	31.91
RAFT	46.50	9.33	29.58	61.92	17.50	46.06	23.94	21.90	32.09
CITL-FT	41.50	15.33	30.89	63.00	19.34	47.86	27.07	23.65	33.58
Self-Distillation-based Finetuning									
SDPO	41.50	14.67	35.91	69.72	28.24	50.90	32.53	37.20	38.83
Reinforcement Learning-based Finetuning									
GRPO	42.00	16.00	36.77	70.84	28.97	51.10	33.43	37.55	39.58
Dr.GRPO	41.00	14.67	35.79	73.20	30.37	51.88	33.13	40.08	40.02
Critique-GRPO	45.50	20.67	38.64	74.96	33.82	52.34	36.97	39.75	42.83
RLTF-FM	42.00	15.33	36.05	69.88	28.46	50.76	32.93	37.65	39.13
Multi-turn Feedback-guided Reinforcement Learning									
MulFeRL	**50.00**	**24.00**	**42.49**	**78.72**	**37.87**	**54.89**	**38.99**	**43.08**	**46.26**
Base Model									
Qwen3-4B-Inst	59.00	46.00	45.13	76.68	42.43	58.56	35.05	39.85	50.34
Supervised Learning-based Finetuning									
SFT	62.50	47.33	49.17	78.12	46.32	60.46	36.26	41.93	52.76
RAFT	62.00	46.00	48.34	78.68	45.29	59.53	37.47	41.05	52.30
CITL-FT	62.00	48.00	48.55	79.64	47.13	60.18	36.87	42.65	53.13
Self-Distillation-based Finetuning									
SDPO	79.50	58.00	59.55	87.80	50.07	63.00	43.84	50.30	61.51
Reinforcement Learning-based Finetuning									
GRPO	78.50	57.33	59.05	87.84	51.25	62.18	43.94	49.85	61.24
Dr.GRPO	79.00	56.67	60.36	88.28	49.19	63.33	42.83	50.68	61.29
Critique-GRPO	84.00	62.67	62.40	88.64	50.29	64.67	45.45	51.05	63.65
RLTF-FM	82.50	60.67	61.54	88.12	49.63	64.25	44.65	50.85	62.78
Multi-turn Feedback-guided Reinforcement Learning									
MulFeRL	**90.50**	**68.00**	**68.13**	**90.24**	**55.96**	**68.08**	**50.20**	**55.75**	**68.36**

Table 1: Evaluation results (Pass@1) on mathematical reasoning (ID) and scientific/general reasoning (OOD) benchmarks. MulFeRL shows significant improvements across all datasets compared to RL baselines using paired t-test.

2507 (Team, 2025) (Qwen3-4B-Inst) as backbones, and GPT-4o (OpenAI, 2024) as the feedback simulator F_ψ. For MulFeRL, we perform one initial rollout turn followed by up to two feedback-guided turns. In each turn, we sample 8 rollouts with temperature 1. Feedback text is capped at 1k tokens when used, and each generated solution is capped at 8k tokens during both training and evaluation. All baselines use the same sampling temperature and length limits unless stated otherwise. For all feedback-based methods, e.g., MulFeRL, SDPO, RLTF-FM, and Critique-GRPO, GPT-4o is used as the feedback provider. SFT methods are trained for 600 steps; SDPO and RL-style methods for 400 steps. For each method, we report the checkpoint with the best overall validation performance. At test time, we decode with temperature 1 and evaluate in a single-pass setting without external feedback unless specified otherwise. MulFeRL adopts the structured output format in Figure 2 throughout training and evaluation; for other methods, we follow their default output formats.[6][7]

4.2 Main Results

As shown in Table 1, MulFeRL achieves the best average performance under both backbones, with consistent gains across diverse in-domain math benchmarks. It also improves out-of-domain science and general reasoning tasks despite being trained primarily on math, indicating that MulFeRL exhibits cross-domain transfer rather than merely fitting the training domain. Overall, these results support our central design: MulFeRL targets low-signal failed groups, uses progress induction to convert feedback into optimizable learning signals, and integrates these signals into policy optimization

[6] A study on output-format is provided in § F.2.1.

[7] More implementation details are provided in § D.

Method	AIME24 (Δ)	MATH500 (Δ)
MulFeRL	**24.00**	**78.72**
w/o Regeneration	16.00 (-8.00)	70.84 (-7.88)
w/o FCO	19.67 (-4.33)	76.12 (-2.60)
w/o Feedback Injection	20.67 (-3.33)	77.68 (-1.04)

Table 2: Ablation of MulFeRL on Qwen2.5-7B-Base.

through feedback-aware credit assignment, thereby improving reasoning and achieving robust gains across domains and backbones.

The comparison with feedback-based baselines further clarifies MulFeRL's advantages. We attribute part of the gain to its RL-based formulation: unlike distillation, MulFeRL optimizes the policy from feedback-derived signals rather than imitating feedback-refined outputs, which may better preserve the model's intrinsic reasoning behaviors, exploration patterns, and inductive biases. We also hypothesize that its focus on low-signal negative samples, together with multi-turn feedback and feedback-aware credit assignment, helps extract useful optimization signal from failed rollouts rather than merely exposing the model to feedback.

4.3 Ablation Study

As shown in Table 2, we ablate three core components of MulFeRL: (1) feedback-guided regeneration, where no additional feedback turns are triggered and training reduces to single-turn GRPO; (2) FCO, where regeneration is retained but cross-state credit assignment is removed, leaving only GRPO updates on contrastive groups; and (3) feedback injection, where the fixed-slot structured injection is replaced with a plain prompting variant that provides the same feedback content without the injection structure. All ablations degrade performance, showing that each component is useful. Removing feedback-guided regeneration causes the largest degradation, confirming that progress induction is crucial for recovering learning signal from all-failed rollout groups. Removing FCO substantially hurts performance, showing that regeneration alone does not fully convert feedback-induced progress into trainable signal: when feedback moves an all-failed group to an all-solved group, within-state GRPO has no reward contrast to optimize, and cross-state FCO is needed to consolidate the improvement. Finally, replacing structured feedback injection with plain prompting leads to consistent declines, suggesting that the fixed-slot injection helps the model condition on feedback more reliably. Overall, MulFeRL benefits from combining feedback-guided progress induction, cross-state credit assignment, and structured feedback utilization.

Simulator	SDPO	RLTF	Critique GRPO	MulFeRL
Qwen3-1.7B (Thinking)	57.86	58.42	58.18	61.32
Qwen3-4B-Inst (Self)	60.06	59.74	61.35	65.62
GPT-4o-mini	60.84	62.36	62.12	66.84
GPT-4o	61.51	62.78	63.65	**68.36**

Table 3: Average Pass@1 across all benchmarks when training feedback-based methods with different feedback simulators on Qwen3-4B-Inst. The *self* simulator uses the frozen base Qwen3-4B-Inst checkpoint without additional simulator training.

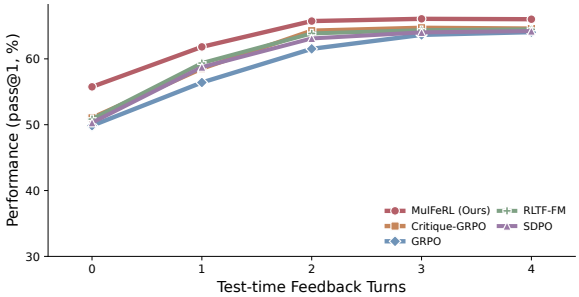


Figure 3: Test-time multi-turn feedback results on TheoremQA (Qwen3-4B-Inst).

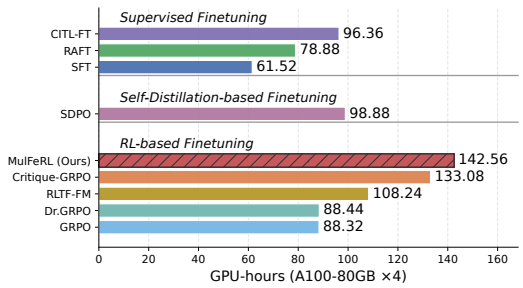
5 Analysis

5.1 Impact of Feedback Simulators

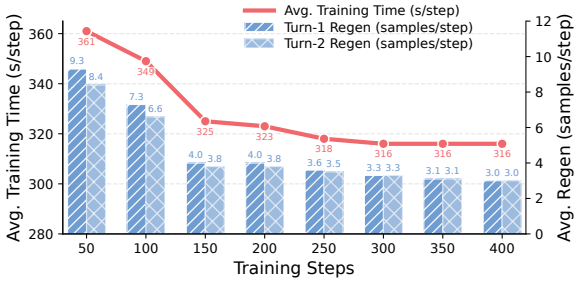
Table 3 compares MulFeRL trained with feedback from different simulators. We observe a consistent trend that higher-quality simulators lead to better downstream performance. Using the base backbone itself as the feedback simulator already yields a strong MulFeRL model (Table 13 for full results). This suggests that even without guidance from a stronger external simulator, MulFeRL can improve reasoning via self-feedback. Moreover, stronger simulators provide larger gains, with GPT-4o-mini improving over self-feedback and GPT-4o further boosting performance. We hypothesize that stronger simulators yield more informative feedback, enabling better regeneration and denser learning signals that improve generalization on challenging math reasoning tasks.

5.2 Test-time Multi-turn Feedback

We evaluate test-time multi-turn feedback on TheoremQA to assess whether MulFeRL training improves a model's ability to use external natural-language feedback for inference-time revision. GPT-4o provides feedback using the same prompts



(a) Training cost (GPU-hours).



(b) MulFeRL training efficiency across steps.

Figure 4: Training efficiency (Qwen3-4B-Inst). (a) Training cost across methods. (b) MulFeRL efficiency across training steps: average training time per step and the average number of regenerated samples per step in Turn-1/Turn-2, reported as averages over 50-step intervals.

as in training, without revealing answers. Figure 3 shows that more feedback turns improve all methods, while MulFeRL consistently achieves the best performance across turns. This indicates that MulFeRL can exploit external feedback effectively to refine its solutions and improve performance under multi-turn feedback. Performance also tends to level off in later turns, suggesting diminishing returns as the feedback simulator becomes the dominant limiting factor.

5.3 Training efficiency

We measure training cost under the unified setup described in the implementation details. As shown in Figure 4a, MulFeRL incurs higher training cost than RL baselines, primarily due to its multi-turn regeneration design. When sampled candidates fail, MulFeRL performs feedback-guided regeneration, introducing additional decoding and verification overhead. Figure 4b further illustrates this effect: regeneration increases the per-step time, with the overhead being most pronounced early in training when failures are more frequent and regeneration is triggered more often. Despite the higher per-step cost, MulFeRL can reach strong performance in fewer optimization steps because feedback-guided regeneration converts low-signal failed groups into verifier-certified progress, while

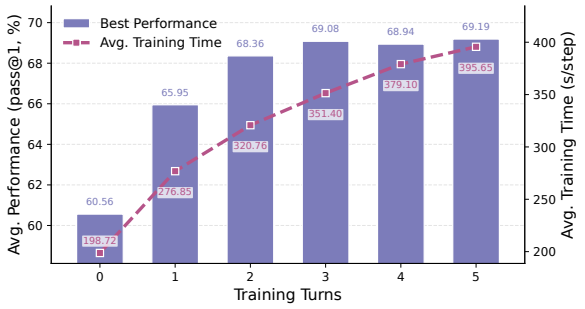


Figure 5: Impact of training turns on effectiveness and efficiency (Qwen3-4B-Inst).

the credit-assignment objective exploits identifiable discrimination, making each update more sample-efficient (see § F.4.1 for details). Thus in practice, overall cost can also be reduced by using a smaller step budget (e.g., early stopping) and optionally annealing regeneration as the policy improves, which mitigates per-step overhead while retaining substantial gains.

5.4 Impact of Multi-turn Training

Figure 5 shows how the training-time per-step regeneration turn budget affects both effectiveness and efficiency. Increasing the number of turns yields rapid initial gains, suggesting that feedback-conditioned regeneration provides stronger learning signal early on. The improvements then saturate, with diminishing returns beyond a small number of turns. In contrast, the average training time per step increases steadily as more turns introduce additional decoding and verification overhead. Overall, the figure highlights a performance–cost trade-off and motivates using a small-to-moderate turn budget that captures most of the benefit while keeping training cost manageable.

6 Conclusion

Outcome-only scalar rewards in RLVR are often sparse on failed samples, leaving all-failed rollout groups with little usable signal and causing training to plateau. We propose MulFeRL, a multi-turn feedback-guided RLVR framework that targets these low-signal groups, injects verbal feedback through a structured format, uses progress induction to turn feedback-driven revisions into optimizable learning signals, and incorporates them into policy optimization through feedback-aware credit assignment. Experiments across two backbones show consistent gains over strong feedback-based baselines, with further analyses revealing the underlying mechanisms.

Limitations

MulFeRL incurs additional training overhead compared with single-turn RLVR baselines, since multi-turn feedback-guided regeneration introduces extra decoding and verification. This cost is especially noticeable when failures are frequent, and can further increase when a strong external model is used as the feedback simulator. Nevertheless, our experiments show that MulFeRL remains effective even with weaker or self-generated feedback, suggesting that cheaper simulators, early stopping, and annealed regeneration can help reduce practical cost.

MulFeRL also inherits common limitations of learning from verbal feedback. Its effectiveness depends on reliable outcome verification and useful feedback: stronger simulators may improve results but add cost and reproducibility constraints, while weaker or noisy feedback may limit the recoverable signal. Our appendix experiments indicate that MulFeRL is relatively robust across feedback sources, but developing cheaper and more reliable feedback mechanisms remains an important future direction.

In addition, the current design uses rollout-level feedback and a conservative triggering strategy: feedback is applied only to all-failed groups, and cross-turn preference pairs are constructed with a simple pairing rule. While this targets the low-signal regime where standard RLVR struggles, it may miss finer-grained improvements in partially correct or near-miss rollouts. Future work can extend MulFeRL to step-level or process-level feedback, adaptive triggering, and richer credit-assignment schemes.

References

Xin-Qiang Cai, Wei Wang, Feng Liu, Tongliang Liu, Gang Niu, and Masashi Sugiyama. 2025. Reinforcement learning with verifiable yet noisy rewards under imperfect verifiers. *arXiv preprint arXiv:2510.00915*.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7889–7901.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

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

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3684.

John Hattie and Helen Timperley. 2007. The power of feedback. *Review of educational research*, 77(1):81–112.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *Preprint*, arXiv:2402.14008.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. 2026. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*.

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. 2026. Why does self-distillation (sometimes) degrade the reasoning capability of llms? *arXiv preprint arXiv:2603.24472*.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.

Avraham N Kluger and Angelo DeNisi. 1996. The effects of feedback interventions on performance: a historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychological bulletin*, 119(2):254.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 3843–3857. Curran Associates, Inc.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Q. Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yan Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. https://github.com/project-numina/aimo-progress-prize. Dataset and code repository.

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

Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. 2024. Chain of hindsight aligns language models with feedback. In *International Conference on Learning Representations*, volume 2024, pages 41395–41414.

Yan Liu, Feng Zhang, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, Han Liu, and Yangdong Deng. 2026. Efficient paths and dense rewards: Probabilistic flow reasoning for large language models. *arXiv preprint arXiv:2601.09260*.

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

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. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

OpenAI. 2024. GPT-4o System Card.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv: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. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Yuda Song, Lili Chen, Fahim Tajwar, Remi Munos, Deepak Pathak, J Andrew Bagnell, Aarti Singh, and Andrea Zanette. 2026. Expanding the capabilities of reinforcement learning via text feedback. *arXiv preprint arXiv:2602.02482*.

Minghao Tang, Shiyu Ni, Jiafeng Guo, and Keping Bi. 2025. Injecting external knowledge into the reasoning process enhances retrieval-augmented generation. In *Proceedings of the 2025 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 41–46.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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

Hongcheng Wang, Yinuo Huang, Sukai Wang, Guanghui Ren, and Hao Dong. 2025a. Grpo-ma: Multi-answer generation in grpo for stable and efficient chain-of-thought training. *arXiv preprint arXiv:2509.24494*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pages 95266–95290.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu,

Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Monica Lam, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. 2025b. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, Xiao Wang, Rui Zheng, Tao Ji, Xiaowei Shi, Yitao Zhai, Rongxiang Weng, Jingang Wang, Xunliang Cai, Tao Gui, and 5 others. 2024. Enhancing llm reasoning via critique models with test-time and training-time supervision. *Preprint*, arXiv:2411.16579.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Chenlu Ye, Zhou Yu, Ziji Zhang, Hao Chen, Narayanan Sadagopan, Jing Huang, Tong Zhang, and Anurag Beniwal. 2025. Beyond correctness: Harmonizing process and outcome rewards through rl training. *arXiv preprint arXiv:2509.03403*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. 2025. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*.

Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. 2025. Critique-grpo: Advancing llm reasoning with natural language and numerical feedback. *arXiv preprint arXiv:2506.03106*.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. 2026. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*.

Congmin Zheng, Jiachen Zhu, Zhuoying Ou, Yuxiang Chen, Kangning Zhang, Rong Shan, Zeyu Zheng, Mengyue Yang, Jianghao Lin, Yong Yu, and Weinan Zhang. 2025. A survey of process reward models: From outcome signals to process supervisions for large language models. *arXiv preprint arXiv:2510.08049*.

A Algorithm Details

This appendix provides the full algorithmic specification of MulFeRL, including notation, reward-state routing, feedback aggregation, structured injection, token masking, canonicalization, and FCO pair sampling.

A.1 Components and Notation

Policy, behavior policy, and reference policy. Let π_θ denote the trainable policy, π_{old} the frozen behavior policy used to sample rollouts for the current outer update, and π_{ref} a fixed reference policy. At the beginning of each outer update, we freeze the current policy as π_{old}. All rollouts used for this update are sampled from π_{old}, while the loss is optimized with respect to π_θ. This follows the standard PPO-style on-policy protocol, where trajectories are sampled from a behavior policy and then optimized using a clipped surrogate objective (Schulman et al., 2017). GRPO follows the same policy-optimization convention while replacing a learned critic with group-relative reward normalization (Shao et al., 2024). The reference policy π_{ref} is used for the KL regularization term in the GRPO branch and for the reference-normalized score in the FCO branch.

Prompt, turns, and rollout groups. For each prompt $x \sim \mathcal{D}$, MulFeRL runs a turn-based feedback trajectory for at most T turns. At turn t, the policy samples a group of K rollouts:

$$\mathcal{G}^{(t)} = [y_1^{(t)}, \ldots, y_K^{(t)}], \qquad y_i^{(t)} \sim \pi_{\text{old}}(\cdot \mid c^{(t)}),$$

where $c^{(t)}$ is the turn-specific conditioning context. The square brackets indicate an ordered list rather than a mathematical set; duplicate rollouts are preserved.

The initial context is feedback-free:

$$c^{(0)} = x.$$

For $t \geq 1$, the context is obtained by injecting the previous-turn feedback into a fixed feedback slot:

$$c^{(t)} = \text{Inject}(x, f^{(t-1)}).$$

Verifier and reward state. The verifier $\mathcal{V}$ assigns a binary reward:

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

The reward state of a group is:

$$z(\mathcal{G}^{(t)}) = \begin{cases} \textsc{Solved}, & \sum_{i=1}^{K} r_i^{(t)} = K, \\ \textsc{Contrastive}, & 0 < \sum_{i=1}^{K} r_i^{(t)} < K, \\ \textsc{Failed}, & \sum_{i=1}^{K} r_i^{(t)} = 0. \end{cases} \quad (7)$$

A CONTRASTIVE group is optimized with GRPO. A feedback-induced SOLVED group following a preceding FAILED group is optimized with FCO. A FAILED group triggers feedback-guided regeneration unless the turn budget has been exhausted.

Feedback provider. The feedback provider F_ψ can be instantiated as a human annotator, an external LLM, or the policy model itself via self-feedback. At a failed turn t, it produces group-level feedback:

$$f^{(t)} \sim F_\psi \left(\cdot \,\middle|\, x, \mathcal{G}^{(t)}, \{r_i^{(t)}\}_{i=1}^{K} \right).$$

The feedback is used only as a conditioning intervention for the next turn. It is not treated as a reward label and is not directly optimized. If the policy model itself is used as the feedback provider, the generated feedback text is treated as stop-gradient conditioning text.

A.2 Reward Design and Verifier Criteria

We use a binary verifiable reward:

$$R_{\mathcal{V}}(x, y) = \begin{cases} 1, & \text{if } y \text{ is answer-correct} \\ & \text{and schema-compliant}, \\ 0, & \text{otherwise}. \end{cases} \quad (8)$$

Answer correctness. A response is answer-correct if its final answer satisfies the task-specific verification rule implemented by $\mathcal{V}$.

Schema compliance. A response is schema-compliant if it follows the structured output template, including the required delimiters and final-answer format. Schema compliance is included only to make automatic parsing and verification well-defined. If either answer correctness or schema compliance fails, the verifier conservatively returns zero reward.

A.3 Multi-turn Feedback-guided Regeneration Loop

MulFeRL applies feedback only when the current group is FAILED. For each prompt, the algorithm optimizes the first turn that provides an informative update signal and then terminates the prompt. This prevents a single prompt from contributing multiple highly correlated updates and avoids overweighting long feedback trajectories.

At turn t, after sampling and verification, the routing rule is:

- **CONTRASTIVE.** If $\mathcal{G}^{(t)}$ contains both successes and failures, MulFeRL applies the GRPO loss on $\mathcal{G}^{(t)}$ and terminates this prompt. This case provides within-context success–failure contrast.
- **SOLVED.** If $t = 0$ and $\mathcal{G}^{(0)}$ is already SOLVED, MulFeRL skips optimization because the prompt is solved without feedback and provides no within-group contrast. If $t > 0$ and $\mathcal{G}^{(t)}$ is SOLVED, then the immediately preceding group $\mathcal{G}^{(t-1)}$ must be FAILED, because feedback is triggered only on failed groups. MulFeRL applies FCO using $\mathcal{G}^{(t-1)}$ as the all-failed anchor and $\mathcal{G}^{(t)}$ as the all-positive regenerated group.
- **FAILED.** If $\mathcal{G}^{(t)}$ is FAILED and $t < T - 1$, MulFeRL queries the feedback provider, injects the feedback into the next-turn context, and regenerates $\mathcal{G}^{(t+1)}$. If $t = T - 1$, no progress has been observed within the turn budget, and the prompt receives zero loss.

First informative turn. Define:

$$t^\star = \min\big\{t : z(\mathcal{G}^{(t)}) = \textsc{Contrastive} \\ \text{or } \big(t > 0 \wedge z(\mathcal{G}^{(t)}) = \textsc{Solved}\big)\big\},$$

if such a turn exists. MulFeRL updates only at $t^\star$. If $z(\mathcal{G}^{(t^\star)}) = \textsc{Contrastive}$, it uses GRPO. If $t^\star > 0$ and $z(\mathcal{G}^{(t^\star)}) = \textsc{Solved}$, it uses FCO anchored by $\mathcal{G}^{(t^\star - 1)}$. If no such $t^\star$ exists, the prompt receives zero loss. An initial SOLVED group at $t = 0$ is skipped rather than treated as informative because it provides neither a failure anchor nor within-group reward variation.

A.4 Feedback Aggregation for Long Rollouts

Reasoning rollouts can be long, making it difficult for a feedback provider to produce concise and reliable feedback over all candidates at once. We therefore use a two-stage feedback aggregation procedure.

Stage 1: subgroup feedback. Given an all-failed group:

$$\mathcal{G}^{(t)} = [y_1^{(t)}, \ldots, y_K^{(t)}],$$

we randomly partition it into g subgroups:

$$\mathcal{S}_1^{(t)}, \ldots, \mathcal{S}_g^{(t)}.$$

For each subgroup, we query the feedback provider:

$$f_j^{(t)} \sim F_\psi \left(\cdot \,\middle|\, \begin{array}{c} x, \\ [y_i^{(t)} : i \in \mathcal{S}_j^{(t)}], \\ [r_i^{(t)} : i \in \mathcal{S}_j^{(t)}] \end{array} \right), \quad j = 1, \ldots, g. \tag{9}$$

Each subgroup feedback identifies common failure patterns and proposes actionable revision steps.

Stage 2: group-level summarization. The subgroup feedbacks are summarized into a single group-level feedback:

$$f^{(t)} \sim \text{Summarize}\left(\cdot \,\middle|\, x, [f_1^{(t)}, \ldots, f_g^{(t)}] \right). \tag{10}$$

The final feedback follows the structured format used for injection:

Issue: root mistake or dominant failure mode,
Fix steps: actionable revision plan.

When rollouts are short, we set $g = 1$ and skip the subgroup aggregation step. The values of K, g, and T are reported in Appendix D.

A.5 Structured Feedback Injection and Token Masking

MulFeRL injects feedback into a fixed <feedback> slot inside the reasoning template. This design is motivated by prior work showing that language feedback can improve subsequent attempts through iterative refinement or verbal reflection (Madaan et al., 2023; Shinn et al., 2023). It is also related in interface design to passage injection, which explicitly places external information inside the reasoning process rather than merely appending it to the input (Tang et al., 2025). In MulFeRL, the injected content is feedback rather than retrieved evidence.

Injection interface. In the default setting, the initial feedback slot is empty:

$$c^{(0)} = x.$$

At regeneration turns, externally produced feedback $f^{(t-1)}$ is inserted into the slot as Figure 2 shows. The model then continues generation conditioned on the injected feedback. If a self-feedback variant is used, it is treated as a separate variant; the default algorithm and the theoretical analysis assume that the initial context is feedback-free.

Token masking. Externally injected feedback tokens are treated as conditioning tokens and are masked out from the optimization loss. Let $M^{(t)}(y)$ denote the set of token indices in rollout y that correspond to model-generated reasoning and answer tokens at turn t. Injected feedback tokens are excluded from $M^{(t)}(y)$. Define the canonicalized body:

$$b^{(t)}(y) = (y_\ell)_{\ell \in M^{(t)}(y)}.$$

The masked conditional log-likelihood is:

$$\log \pi_\theta(b^{(t)}(y) \mid c^{(t)}) = \sum_{\ell \in M^{(t)}(y)} \log \pi_\theta \left(y_\ell \,\middle|\, c^{(t)}, y_{<\ell} \right). \tag{11}$$

The same masking convention is used for π_{old} and π_{ref}.

Importantly, canonicalization removes injected feedback from the likelihood objective, but it does not remove feedback from the conditioning context. For $t > 0$, the probability in Eq. (11) is still computed under the original feedback-augmented context $c^{(t)}$.

A.6 Loss Interfaces

GRPO branch. For a CONTRASTIVE group $\mathcal{G}^{(t)}$, MulFeRL computes group-relative advantages:

$$\begin{aligned} \bar{r}^{(t)} &= \frac{1}{K} \sum_{i=1}^{K} r_i^{(t)}, \\ \hat{\sigma}_r^{(t)} &= \sqrt{\frac{1}{K} \sum_{i=1}^{K} \left(r_i^{(t)} - \bar{r}^{(t)} \right)^2}, \\ \hat{A}_i^{(t)} &= \frac{r_i^{(t)} - \bar{r}^{(t)}}{\hat{\sigma}_r^{(t)} + \epsilon_{\text{adv}}}. \end{aligned}$$

For compactness, write $M_i^{(t)} = M^{(t)}(y_i^{(t)})$. For an unmasked token $\ell \in M_i^{(t)}$, define the PPO-style importance ratio:

$$\rho_{i,\ell}^{(t)}(\theta) = \frac{\pi_\theta \left(y_{i,\ell}^{(t)} \,\middle|\, c^{(t)}, y_{i,<\ell}^{(t)} \right)}{\pi_{\text{old}} \left(y_{i,\ell}^{(t)} \,\middle|\, c^{(t)}, y_{i,<\ell}^{(t)} \right)}.$$

The GRPO branch uses the clipped group-relative policy loss:

$$\begin{aligned} \mathcal{L}_{\text{GRPO}}^{(t)} = -\frac{1}{K} \sum_{i=1}^{K} \frac{1}{|M_i^{(t)}|} \sum_{\ell \in M_i^{(t)}} \min\Big(& \rho_{i,\ell}^{(t)}(\theta) \hat{A}_i^{(t)}, \\ & \text{clip}\big(\rho_{i,\ell}^{(t)}(\theta), 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}\big) \hat{A}_i^{(t)} \Big) \\ & + \beta_{\text{KL}} \mathcal{R}_{\text{KL}}^{(t)}. \end{aligned} \tag{12}$$

Here $\mathcal{R}_{\mathrm{KL}}^{(t)}$ denotes the implementation's reference-policy KL regularizer computed over the same unmasked generated tokens. The exact KL estimator follows the implementation used in the main experiments.

FCO branch. FCO is used only for a FAILED→SOLVED transition. For a canonicalized response body generated under context c, define the length-normalized reference score:

$$s_\theta(c, y) = \frac{1}{|b(y)|}\left[\log \pi_\theta(b(y) \mid c) - \log \pi_{\mathrm{ref}}(b(y) \mid c)\right], \tag{13}$$

where the log-likelihoods are computed using the masking convention in Eq. (11). This reference-normalized form is analogous in spirit to pairwise preference objectives such as DPO, but here the pairwise ordering is produced by an automatic verifier and an adjacent FAILED→SOLVED transition rather than by human preference annotations (Rafailov et al., 2023).

For a positive–negative pair (y^+, y^-) with contexts (c^+, c^-), define:

$$D_\theta(y^+, y^-) = s_\theta(c^+, y^+) - s_\theta(c^-, y^-).$$

The pairwise FCO loss is:

$$\ell_{\mathrm{FCO}}(y^+, y^-) = -\log \sigma\left(\tau\left[D_\theta(y^+, y^-) - \gamma\right]\right), \tag{14}$$

where $\tau > 0$ is an inverse-temperature parameter and $\gamma \geq 0$ is a margin. Given a pair set $\mathcal{Q}_{\mathrm{FCO}}^{(t)}$, the FCO branch uses:

$$\mathcal{L}_{\mathrm{FCO}}^{(t)} = \frac{1}{|\mathcal{Q}_{\mathrm{FCO}}^{(t)}|} \sum_{(y^+, y^-) \in \mathcal{Q}_{\mathrm{FCO}}^{(t)}} \ell_{\mathrm{FCO}}(y^+, y^-). \tag{15}$$

A.7 Canonicalization for FCO

FCO compares responses generated under different conditioning contexts. To ensure that this comparison is made over model-generated content rather than externally supplied feedback, we use the canonicalized body $b^{(t)}(y)$ defined in Section A.5. The canonicalized body determines which tokens enter the likelihood objective. The turn-specific context remains unchanged.

This distinction is important because FCO compares the current all-positive group $\mathcal{G}^{(t)}$ with the immediately preceding all-failed group $\mathcal{G}^{(t-1)}$. These groups may have different feedback contexts, so the objective should compare generated solutions conditioned on their respective contexts, not the injected feedback text itself.

A.8 FCO Pair Construction and Sampling

FCO is used only for a FAILED→SOLVED transition. Let $\mathcal{G}^{(t-1)}$ be the preceding all-failed group under context $c^{(t-1)}$, and let $\mathcal{G}^{(t)}$ be the current all-positive group under context $c^{(t)}$.

We construct ordered lists of positive and negative examples:

$$\begin{aligned} \mathcal{Y}_+^{(t)} &= \left[(c^{(t)}, b^{(t)}(y_i^{(t)}))\right]_{i=1}^K, \\ \mathcal{Y}_-^{(t-1)} &= \left[(c^{(t-1)}, b^{(t-1)}(y_i^{(t-1)}))\right]_{i=1}^K. \end{aligned}$$

These are ordered multisets rather than mathematical sets. Duplicate canonicalized responses are preserved, so both lists have length K even if two rollouts yield identical bodies after canonicalization.

The full FCO pair set is the all-pair Cartesian product:

$$\mathcal{Q}_{\mathrm{FCO,full}}^{(t)} = \mathcal{Y}_+^{(t)} \times \mathcal{Y}_-^{(t-1)}. \tag{16}$$

Permutation-based sampling. Using all K^2 pairs may be expensive. By default, we use a permutation-based estimator. Write:

$$\begin{aligned} \mathcal{Y}_+^{(t)} &= [y_1^+, \ldots, y_K^+], \\ \mathcal{Y}_-^{(t-1)} &= [y_1^-, \ldots, y_K^-], \end{aligned}$$

where each y_j^+ or y_i^- denotes a context-body pair. We sample a random permutation π of $\{1, \ldots, K\}$ and construct:

$$\widehat{\mathcal{Q}}_{\mathrm{FCO}}^{(t)} = [(y_j^+, y_{\pi(j)}^-) : j = 1, \ldots, K]. \tag{17}$$

This avoids assuming any same-index correspondence between turns while using each rollout once per update.

For any pairwise loss $\ell(y^+, y^-)$, the permutation estimator is unbiased for the full all-pair average:

$$\mathbb{E}_\pi\left[\frac{1}{K}\sum_{j=1}^K \ell(y_j^+, y_{\pi(j)}^-)\right] = \frac{1}{K^2}\sum_{j=1}^K\sum_{i=1}^K \ell(y_j^+, y_i^-). \tag{18}$$

For notational simplicity, the main text writes $\mathcal{Q}_{\mathrm{FCO}}^{(t)}$ to denote either the full pair set $\mathcal{Q}_{\mathrm{FCO,full}}^{(t)}$ or its sampled estimator $\widehat{\mathcal{Q}}_{\mathrm{FCO}}^{(t)}$.

A.9 Verifier Gating and Reference Anchoring

MulFeRL does not directly optimize feedback text. Feedback only changes the next-turn conditioning context. A policy update is applied only when the verifier observes either a CONTRASTIVE group or a FAILED→SOLVED transition. If feedback fails to move the group out of the FAILED state before

the turn budget is exhausted, the prompt receives zero loss.

The GRPO branch uses a PPO-style clipped surrogate and reference-policy regularization. The FCO branch uses a reference-normalized score, which anchors pairwise comparisons to the fixed reference policy. This anchoring should not be interpreted as a formal KL trust region by itself; rather, it is a practical regularization mechanism inside the pairwise contrastive loss.

A.10 Pseudocode

Algorithm 1 summarizes the training procedure for one prompt. The behavior policy π_{old} is assumed to be frozen before rollout collection for the current outer update.

B Theoretical Analysis

We provide a theoretical analysis of MulFeRL. The goal is not to prove a global monotonic improvement guarantee, which would require stronger assumptions on optimization, exploration, verifier correctness, and model parameterization. Instead, we establish a more modest but useful claim: the routing rule in MulFeRL converts verifier-observed reward patterns into the appropriate local training signal. In particular, GRPO extracts *within-context* contrast when successes and failures coexist in the same rollout group, whereas FCO extracts *cross-context* contrast when feedback turns an all-failed group into an all-solved group.

This view is consistent with prior policy-optimization and preference-learning methods. PPO optimizes a clipped surrogate objective to stabilize policy updates (Schulman et al., 2017). GRPO, introduced in DeepSeekMath, replaces a learned value critic with group-relative normalization and has been used successfully in verifiable-reward reasoning settings (Shao et al., 2024; DeepSeek-AI, 2025). FCO is not a GRPO estimator. Rather, it is a verifier-gated pairwise contrastive surrogate, analogous in form to reference-normalized preference objectives such as DPO, which increase the relative score of preferred responses over dispreferred responses under a reference-policy normalization (Rafailov et al., 2023). The difference is that our preference labels are not human preference annotations; they are induced by an automatic verifier and a temporally adjacent Failed→Solved transition.

B.1 Preliminaries

For a prompt x and context c, define the verifier-success and verifier-failure sets:

$$S_x = \{y : R_{\mathcal{V}}(x, y) = 1\}, \qquad F_x = \{y : R_{\mathcal{V}}(x, y) = 0\},$$

The success probability under context c is:

$$p_\theta(c) = \Pr_{y \sim \pi_\theta(\cdot|c)}[y \in S_x]$$

Here c may be the original prompt or a feedback-augmented prompt. In the actual PPO/GRPO-style data-collection protocol, this rollout distribution is instantiated by the frozen behavior policy at the current outer update, i.e., $\theta = \theta_{\text{old}}$; we keep the θ notation in the analysis to study the local update direction around that point.

Assumption 1 (Conditional rollout independence)**.** *Given a fixed context c, the K rollouts in a group are sampled independently from $\pi_\theta(\cdot \mid c)$.*

Under Assumption 1, the probabilities of the three verifier-observed group states are:

$$P_{\text{SOLVED}}(c) = p_\theta(c)^K, \tag{19}$$

$$P_{\text{FAILED}}(c) = (1 - p_\theta(c))^K, \tag{20}$$

$$P_{\text{CONTRASTIVE}}(c) = 1 - p_\theta(c)^K - (1 - p_\theta(c))^K. \tag{21}$$

For hard prompts, $p_\theta(c)$ is small. More precisely, if $p_\theta(c) \ll 1$ and $Kp_\theta(c) \ll 1$, then:

$$(1 - p_\theta(c))^K = 1 - Kp_\theta(c) + O(K^2 p_\theta(c)^2).$$

The failed-state probability expands as:

$$P_{\text{FAILED}}(c) = 1 - Kp_\theta(c) + O(K^2 p_\theta(c)^2),$$

The contrastive-state probability expands as:

$$P_{\text{CONTRASTIVE}}(c) = Kp_\theta(c) - O(K^2 p_\theta(c)^2) - p_\theta(c)^K.$$

This explains why hard prompts are likely to produce all-failed groups: the same prompts for which learning is most needed are often the prompts for which vanilla group-relative RL receives no within-group contrast.

B.2 Uniform-Reward Groups Provide No Reward-Dependent GRPO Signal

Proposition 1 (Vanishing group-relative advantage)**.** *If all rewards in a rollout group are identical, then the normalized group-relative advantages are zero. Consequently, the reward-dependent component of the GRPO policy surrogate provides no discriminative learning signal.*

Algorithm 1 MulFeRL training for one prompt x

Require: Trainable policy π_θ; frozen behavior policy π_{old}; reference policy π_{ref}; verifier $\mathcal{V}$; feedback provider F_ψ; group size K; max turns T; subgroup count g; GRPO parameters $(\epsilon_{\text{clip}}, \epsilon_{\text{adv}}, \beta_{\text{KL}})$; FCO parameters (τ, γ); FCO weight λ_{FCO}.

$c^{(0)} \leftarrow x$
for $t = 0, \ldots, T-1$ **do**
  Sample $\mathcal{G}^{(t)} = [y_i^{(t)}]_{i=1}^K$ with $y_i^{(t)} \sim \pi_{\text{old}}(\cdot \mid c^{(t)})$
  Compute $r_i^{(t)} \leftarrow R_{\mathcal{V}}(x, y_i^{(t)})$ for $i = 1, \ldots, K$
  Determine $z(\mathcal{G}^{(t)})$ by Eq. (7)
  if $z(\mathcal{G}^{(t)}) = \textsc{Contrastive}$ **then**
    $\mathcal{L}(x) \leftarrow \mathcal{L}_{\text{GRPO}}^{(t)}$ by Eq. (12)
    return $\mathcal{L}(x)$
  else if $z(\mathcal{G}^{(t)}) = \textsc{Solved}$ **then**
    if $t = 0$ **then**
      return 0 {Solved without feedback; no contrastive signal}
    else
      Construct $\widehat{\mathcal{Q}}_{\text{FCO}}^{(t)}$ from $\mathcal{G}^{(t)}$ and $\mathcal{G}^{(t-1)}$ by Eq. (17)
      $\mathcal{L}(x) \leftarrow \lambda_{\text{FCO}} \mathcal{L}_{\text{FCO}}^{(t)}$ by Eq. (15)
      return $\mathcal{L}(x)$
    end if
  else
    $\{z(\mathcal{G}^{(t)}) = \textsc{Failed}\}$
    if $t = T-1$ **then**
      return 0 {No progress within turn budget}
    end if
    Obtain subgroup feedbacks and summarize them into $f^{(t)}$ using Section A.4
    $c^{(t+1)} \leftarrow \text{Inject}(x, f^{(t)})$
  end if
end for
return 0

Proof. Let $\mathcal{G} = \{y_i\}_{i=1}^K$ be a rollout group with rewards $r_i = R_{\mathcal{V}}(x, y_i)$. GRPO-style group-relative normalization computes:

$$\bar{r} = \frac{1}{K}\sum_{i=1}^{K} r_i, \qquad \hat{\sigma}_r = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(r_i - \bar{r})^2},$$

and the corresponding normalized advantage is:

$$\hat{A}_i = \frac{r_i - \bar{r}}{\hat{\sigma}_r + \epsilon},$$

where $\epsilon > 0$ is a numerical stabilizer. If all rewards are identical, then $r_i = \bar{r}$ for all i, and hence:

$$\hat{A}_i = 0, \qquad \forall i \in [K].$$

The reward-dependent GRPO/PPO surrogate contains terms of the form:

$$\min\big(\rho_i(\theta)\hat{A}_i, \text{clip}(\rho_i(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_i\big),$$

which vanish whenever $\hat{A}_i = 0$. Therefore, uniform-reward groups provide no reward-dependent discriminative signal. This applies to both all-failed groups and all-solved groups. □

Remark. Proposition 1 concerns the reward-dependent part of the policy objective. If an implementation also applies an explicit KL penalty or other regularizer on these samples, that regularizer may still produce a gradient. The proposition only states that no success–failure credit assignment can be extracted from a group whose verifier rewards are uniform.

Implication. This proposition motivates the routing rule in MulFeRL. An initial SOLVED group is skipped because the prompt is already solved and no contrast is available. A FAILED group triggers progress induction because it is unsolved and low-signal. A feedback-induced SOLVED group is

not skipped, because the preceding all-failed group provides a cross-context failure anchor.

B.3 Progress Induction Exposes Additional Update Signals

Suppose a prompt reaches a FAILED state at turn t. Feedback produces a new context:

$$c^{(t+1)} = \text{Inject}(x, f^{(t)}).$$

Let the pre- and post-feedback success probabilities be:

$$p_t = p_\theta(c^{(t)}), \qquad q_t = p_\theta(c^{(t+1)})$$

denote the success probabilities before and after injecting feedback.

Lemma 1 (Escape probability after feedback)**.** *Conditioned on the feedback-augmented context $c^{(t+1)}$, the probability that the regenerated group contains at least one verifier-successful rollout is:*

$$P_{\text{escape}}(c^{(t+1)}) = 1 - (1 - q_t)^K.$$

Moreover, if the feedback improves the single-rollout success probability, namely:

$$q_t > p_t,$$

then the escape probabilities satisfy:

$$1 - (1 - q_t)^K > 1 - (1 - p_t)^K.$$

Proof. Under Assumption 1, the regenerated group remains all-failed with probability $(1 - q_t)^K$. Therefore, the probability of escaping the all-failed state is:

$$P_{\text{escape}}(c^{(t+1)}) = 1 - (1 - q_t)^K.$$

The function $h(z) = 1 - (1 - z)^K$ is strictly increasing for $z \in [0, 1)$ and $K \geq 1$, because:

$$h'(z) = K(1 - z)^{K-1} \geq 0,$$

with strict positivity for $z < 1$. Therefore, $q_t > p_t$ implies $h(q_t) > h(p_t)$. □

Lemma 1 is intentionally conditional. It does not claim that any feedback string necessarily improves the policy. It states that whenever feedback increases the success probability from p_t to q_t, the probability of observing a non-failed group increases accordingly. This distinction is important because poor feedback may leave $q_t \leq p_t$.

After feedback, the regenerated group can be either CONTRASTIVE, SOLVED, or FAILED. Their probabilities are:

$$P_{\text{CONTRASTIVE}}(c^{(t+1)}) = 1 - q_t^K - (1 - q_t)^K, \tag{22}$$

$$P_{\text{SOLVED}}(c^{(t+1)}) = q_t^K, \tag{23}$$

$$P_{\text{FAILED}}(c^{(t+1)}) = (1 - q_t)^K. \tag{24}$$

Vanilla GRPO can use the CONTRASTIVE case because successes and failures coexist in the same group. However, by Proposition 1, it cannot extract reward-dependent success–failure credit from the SOLVED case because all rewards are equal. MulFeRL additionally uses the feedback-induced SOLVED case through FCO. Hence, conditioned on entering a feedback turn, the probability that MulFeRL obtains a verifier-based update signal is:

$$\begin{aligned} P_{\text{update}}^{\text{MulFeRL}} &= P_{\text{CONTRASTIVE}}(c^{(t+1)}) \qquad (25) \\ &\quad + P_{\text{SOLVED}}(c^{(t+1)}) \\ &= 1 - (1 - q_t)^K. \qquad (26) \end{aligned}$$

The corresponding probability for GRPO after the same feedback turn is:

$$\begin{aligned} P_{\text{update}}^{\text{GRPO}} &= P_{\text{CONTRASTIVE}}(c^{(t+1)}) \\ &= 1 - q_t^K - (1 - q_t)^K. \end{aligned}$$

Therefore, the update probability recovered by FCO is:

$$\begin{aligned} P_{\text{update}}^{\text{MulFeRL}} - P_{\text{update}}^{\text{GRPO}} \\ = q_t^K. \end{aligned} \tag{27}$$

Equation (27) characterizes the precise probability mass that is lost by a purely within-group contrastive method but recovered by FCO: the feedback-induced all-solved groups.

B.4 GRPO Provides Within-Context Success–Failure Credit Assignment

We next analyze the local reward-gradient direction of binary-reward GRPO on a CONTRASTIVE group. Consider a group:

$$\mathcal{G} = \mathcal{G}^+ \cup \mathcal{G}^-,$$

where the group sizes are:

$$\begin{aligned} \mathcal{G}^+ &= \{y_i : R_{\mathcal{V}}(x, y_i) = 1\}, \\ \mathcal{G}^- &= \{y_i : R_{\mathcal{V}}(x, y_i) = 0\}. \end{aligned}$$

Define:

$$m = |\mathcal{G}^+|, \qquad n = |\mathcal{G}^-|, \qquad K = m + n,$$

with $m > 0$ and $n > 0$.

For binary rewards:

$$\bar{r} = \frac{m}{K}, \qquad \hat{\sigma}_r = \sqrt{\frac{m}{K}\frac{n}{K}} = \frac{\sqrt{mn}}{K}.$$

With numerical stabilizer $\epsilon > 0$, define:

$$d_\epsilon = \frac{\sqrt{mn}}{K} + \epsilon.$$

Then the normalized advantage for a successful rollout is:

$$\hat{A}^+ = \frac{1 - \bar{r}}{d_\epsilon} = \frac{n/K}{d_\epsilon},$$

and the normalized advantage for a failed rollout is:

$$\hat{A}^- = \frac{0 - \bar{r}}{d_\epsilon} = -\frac{m/K}{d_\epsilon}.$$

Proposition 2 (Local GRPO contrast direction)**.** *At $\theta = \theta_{\text{old}}$, ignoring clipping saturation and explicit regularization terms, the reward-dependent GRPO gradient on a binary-reward* CONTRASTIVE *group is a positive scalar multiple of:*

$$\frac{1}{m} \sum_{y^+ \in \mathcal{G}^+} \nabla_\theta \log \pi_\theta(y^+ \mid c) - \frac{1}{n} \sum_{y^- \in \mathcal{G}^-} \nabla_\theta \log \pi_\theta(y^- \mid c).$$

Proof. The unclipped reward-dependent local surrogate has gradient:

$$\nabla_\theta J_{\text{GRPO}} = \frac{1}{K}\left[\sum_{y^+ \in \mathcal{G}^+} \hat{A}^+ \nabla_\theta \log \pi_\theta(y^+ \mid c) + \sum_{y^- \in \mathcal{G}^-} \hat{A}^- \nabla_\theta \log \pi_\theta(y^- \mid c)\right],$$

because $\nabla_\theta \rho_i(\theta)|_{\theta=\theta_{\text{old}}} = \nabla_\theta \log \pi_\theta(y_i \mid c)$. Substituting the binary advantages gives:

$$\nabla_\theta J_{\text{GRPO}} = \frac{1}{K}\left[\frac{n/K}{d_\epsilon} \sum_{y^+ \in \mathcal{G}^+} \nabla_\theta \log \pi_\theta(y^+ \mid c) - \frac{m/K}{d_\epsilon} \sum_{y^- \in \mathcal{G}^-} \nabla_\theta \log \pi_\theta(y^- \mid c)\right].$$

Equivalently:

$$\nabla_\theta J_{\text{GRPO}} = \frac{mn}{K^2 d_\epsilon}\left[\frac{1}{m} \sum_{y^+ \in \mathcal{G}^+} \nabla_\theta \log \pi_\theta(y^+ \mid c) - \frac{1}{n} \sum_{y^- \in \mathcal{G}^-} \nabla_\theta \log \pi_\theta(y^- \mid c)\right].$$

Since $m, n, K, d_\epsilon > 0$, the scalar multiplier is positive. □

Proposition 2 shows that, locally and on the reward-dependent part of the objective, GRPO increases the average log-likelihood of verifier-successful samples relative to verifier-failed samples under the same context. This is a within-context credit-assignment signal. The statement is local: clipping, KL regularization, parameter sharing, finite step sizes, and optimization noise can affect the realized update.

B.5 FCO Provides Cross-Context Success–Failure Credit Assignment

We now analyze the case that motivates FCO. Consider a FAILED→SOLVED transition. Let $\mathcal{G}^{(t-1)}$ be the all-failed group sampled under context $c^{(t-1)}$, and let $\mathcal{G}^{(t)}$ be the all-solved group sampled under the feedback-augmented context $c^{(t)}$. Thus, for all $y^- \in \mathcal{G}^{(t-1)}$ and $y^+ \in \mathcal{G}^{(t)}$:

$$R_{\mathcal{V}}(x, y^-) = 0, \qquad R_{\mathcal{V}}(x, y^+) = 1.$$

FCO defines the length-normalized reference score:

$$s_\theta(c, y) = \frac{1}{|y|}\left[\log \pi_\theta(y \mid c) - \log \pi_{\text{ref}}(y \mid c)\right],$$

where π_{ref} is fixed. For a positive–negative pair (y^+, y^-), define the cross-context score gap:

$$D_\theta(y^+, y^-) = s_\theta(c^{(t)}, y^+) - s_\theta(c^{(t-1)}, y^-).$$

The pairwise FCO loss is:

$$\ell_{\text{FCO}}(y^+, y^-) = -\log \sigma\left(\tau\left[D_\theta(y^+, y^-) - \gamma\right]\right),$$

where $\tau > 0$ is an inverse-temperature parameter and $\gamma \geq 0$ is a margin.

Proposition 3 (Local FCO contrast direction)**.** *For any verifier-certified pair (y^+, y^-) from a* FAILED→SOLVED *transition, the negative gradient direction of the FCO loss is:*

$$-\nabla_\theta \ell_{\text{FCO}}(y^+, y^-) = \alpha_\theta(y^+, y^-)\left[\frac{1}{|y^+|}\nabla_\theta \log \pi_\theta(y^+ \mid c^{(t)}) - \frac{1}{|y^-|}\nabla_\theta \log \pi_\theta(y^- \mid c^{(t-1)})\right],$$

where the scalar coefficient is:

$$\alpha_\theta(y^+, y^-) = \tau\sigma\left(-\tau\left[D_\theta(y^+, y^-) - \gamma\right]\right)$$

satisfies:

$$0 < \alpha_\theta(y^+, y^-) < \tau.$$

Proof. Let the scalar logit be:

$$z_\theta = \tau \left[D_\theta(y^+, y^-) - \gamma \right].$$

Since:

$$\ell_{\text{FCO}} = -\log \sigma(z_\theta),$$

we have:

$$\begin{aligned} -\nabla_\theta \ell_{\text{FCO}} &= \sigma(-z_\theta) \nabla_\theta z_\theta \\ &= \tau \sigma(-z_\theta) \nabla_\theta D_\theta(y^+, y^-). \end{aligned}$$

Because π_{ref} is fixed:

$$\nabla_\theta s_\theta(c, y) = \frac{1}{|y|} \nabla_\theta \log \pi_\theta(y \mid c).$$

Therefore, the score-gradient is:

$$\begin{aligned} \nabla_\theta D_\theta(y^+, y^-) = &\frac{1}{|y^+|} \nabla_\theta \log \pi_\theta(y^+ \mid c^{(t)}) \\ &- \frac{1}{|y^-|} \nabla_\theta \log \pi_\theta(y^- \mid c^{(t-1)}). \end{aligned}$$

Finally, since $\sigma(\cdot) \in (0, 1)$ and $\tau > 0$, the scalar coefficient $\alpha_\theta(y^+, y^-)$ lies in $(0, \tau)$. □

Proposition 3 shows that FCO reinforces verifier-successful responses under the feedback-augmented context while suppressing verifier-failed responses under the preceding failed context. This is a cross-context contrastive signal. Unlike GRPO, it is not an unbiased estimator of a same-context policy gradient, because the positive and negative samples are conditioned on different contexts. Its role is instead to convert a verifier-certified temporal transition into a pairwise training signal.

B.6 Unified View: Where the Contrast Is Observable

The preceding propositions show that GRPO and FCO have the same local sign structure but operate on different observable contrasts:

increase verifier-successful samples
relative to verifier-failed samples.

The distinction is where this success–failure contrast is observed.

In a CONTRASTIVE group, successes and failures coexist under the same context. GRPO can therefore use within-context group-relative advantages. In a FAILED→SOLVED transition, the current group is uniformly positive and has no within-group reward variation. GRPO is uninformative by Proposition 1. However, the immediately preceding all-failed group provides a verifier-certified negative anchor, and FCO uses the adjacent cross-context contrast.

Thus, MulFeRL does not combine unrelated losses. It routes each prompt according to the location of the verifier-observed discrimination signal:

$$\begin{cases} \text{CONTRASTIVE}: & \text{GRPO within-context,} \\ \text{FAILED} \to \text{SOLVED}: & \text{FCO cross-context,} \\ \text{FAILED} \to \text{FAILED}: & \text{no positive update,} \\ \text{SOLVED AT INITIAL TURN}: & \text{skip.} \end{cases}$$

This routing is also consistent with the broader feedback-refinement literature: language feedback can change the conditioning context and improve subsequent attempts without itself being treated as a reward label (Madaan et al., 2023; Shinn et al., 2023). MulFeRL uses feedback only to induce a new context; the training signal is still gated by the verifier.

B.7 Why FCO Is Needed for Reward-Saturated Progress

A feedback-induced SOLVED group is a reward-saturated progress event. It is useful behaviorally because all regenerated candidates are correct. However, it is unusable for reward-dependent GRPO because all normalized advantages vanish. If such groups were skipped, the algorithm would discard precisely the cases in which feedback was most effective.

FCO recovers this otherwise discarded signal. Since feedback is triggered only after a FAILED group, every feedback-induced SOLVED group has an adjacent all-failed anchor:

$$\mathcal{G}^{(t-1)} : \text{all failed}, \qquad \mathcal{G}^{(t)} : \text{all solved}.$$

The verifier therefore certifies every cross pair:

$$(y^+, y^-) \in \mathcal{G}^{(t)} \times \mathcal{G}^{(t-1)}$$

as ordered in the desired direction:

$$R_{\mathcal{V}}(x, y^+) = 1 > 0 = R_{\mathcal{V}}(x, y^-).$$

FCO turns this ordered transition into a pairwise contrastive objective. It is therefore not an auxiliary preference loss over arbitrary model generations; it is the mechanism that makes reward-saturated progress learnable.

B.8 Unbiasedness of Permutation-Based FCO Pair Sampling

The full FCO objective over a transition averages over all K^2 positive– negative pairs:

$$\mathcal{L}_{\text{full}} = \frac{1}{K^2} \sum_{j=1}^{K} \sum_{i=1}^{K} \ell_{\text{FCO}}(y_j^+, y_i^-).$$

To reduce computation, MulFeRL samples a random permutation π over $\{1, \ldots, K\}$ and uses:

$$\mathcal{L}_{\text{perm}} = \frac{1}{K} \sum_{j=1}^{K} \ell_{\text{FCO}}(y_j^+, y_{\pi(j)}^-).$$

Proposition 4 (Unbiased pair sampling)**.** *If π is sampled uniformly at random from all permutations of $\{1, \ldots, K\}$, then:*

$$\mathbb{E}_\pi[\mathcal{L}_{\text{perm}}] = \mathcal{L}_{\text{full}}.$$

Moreover, whenever differentiation and expectation can be interchanged:

$$\mathbb{E}_\pi[\nabla_\theta \mathcal{L}_{\text{perm}}] = \nabla_\theta \mathcal{L}_{\text{full}}.$$

Proof. For any fixed j, $\pi(j)$ is uniformly distributed over $\{1, \ldots, K\}$. Therefore:

$$\mathbb{E}_\pi \left[\ell_{\text{FCO}}(y_j^+, y_{\pi(j)}^-)\right] = \frac{1}{K} \sum_{i=1}^{K} \ell_{\text{FCO}}(y_j^+, y_i^-).$$

Averaging over j gives:

$$\mathbb{E}_\pi[\mathcal{L}_{\text{perm}}] = \frac{1}{K} \sum_{j=1}^{K} \frac{1}{K} \sum_{i=1}^{K} \ell_{\text{FCO}}(y_j^+, y_i^-) = \mathcal{L}_{\text{full}}.$$

The gradient statement follows by linearity of expectation under the standard regularity condition that the gradient can be exchanged with expectation. □

This result shows that permutation-based FCO preserves the full all-pair objective in expectation while reducing the number of evaluated pairs from K^2 to K.

B.9 Verifier Gating and Regularization

MulFeRL is designed to avoid directly optimizing noisy feedback text. Feedback only changes the next-turn conditioning context. A policy update is applied only when the verifier observes either a CONTRASTIVE group or a FAILED→SOLVED transition. Therefore, unsuccessful feedback does not create positive labels by itself.

The update is also regularized in two ways. First, the GRPO branch follows PPO-style clipped policy optimization and, in common implementations, includes a reference-policy KL term (Schulman et al., 2017; Shao et al., 2024). These mechanisms are intended to limit large policy deviations during on-policy optimization. Second, FCO uses the reference-normalized score:

$$s_\theta(c, y) = \frac{1}{|y|} \left[\log \pi_\theta(y \mid c) - \log \pi_{\text{ref}}(y \mid c)\right].$$

This score anchors the pairwise comparison to changes relative to the reference policy: if $\pi_\theta = \pi_{\text{ref}}$, then $s_\theta(c, y) = 0$ for all (c, y). Hence FCO prefers a positive response over a negative response only to the extent that the current policy improves their relative normalized score beyond the reference baseline.

This reference normalization should not be interpreted as a formal KL trust region by itself. It is an anchoring mechanism inside a pairwise logistic loss. A formal global stability guarantee would require additional assumptions, such as explicit KL constraints, bounded gradients, sufficiently small step sizes, or convexity-like conditions that do not hold for neural language models. Our claim is therefore limited to verifier-gated, reference-anchored local optimization.

B.10 Summary of the Theoretical Justification

The analysis establishes the following claims.

First, uniform-reward groups provide no reward-dependent GRPO credit assignment: their group-relative advantages vanish. This explains why all-failed and all-solved groups are uninformative for purely within-group contrastive RL.

Second, feedback-guided regeneration exposes additional update opportunities. Conditioned on a feedback-induced success probability q_t, the probability of escaping an all-failed state is $1-(1-q_t)^K$. If feedback improves the single-rollout success probability from p_t to $q_t > p_t$, this escape probability strictly increases. Among the escaped cases, FCO recovers exactly the all-solved probability mass q_t^K that GRPO cannot use.

Third, GRPO and FCO share the same local sign structure: both increase verifier-successful samples relative to verifier-failed samples. The difference is where the contrast is observed. GRPO uses within-context contrast in a CONTRASTIVE group. FCO uses adjacent cross-context contrast in a FAILED→SOLVED transition.

Finally, FCO should be viewed as a verifier-certified cross-context contrastive surrogate, not as an unbiased estimator of the original GRPO objective. This interpretation is both mathemati-

cally accurate and sufficient to justify the algorithmic role of FCO: it turns reward-saturated progress events into usable training signals while preserving verifier gating and reference anchoring.

C Prompts for MulFeRL

We use a small set of prompts throughout training. Concretely, we include three system prompts (for training rollouts, feedback generation, and feedback merging) and one prompt template for feedback injection and regeneration. We provide the exact prompt texts in Figures 6, 7, 8, and 9, respectively. The corresponding user prompts are simple and self-explanatory (e.g., the problem statement, and when needed, the model's sampled rollouts and/or multiple feedback comments), so we omit them to avoid redundancy.

During evaluation, for experiments that require feedback, we use the same input system prompt, feedback-generation prompt, and feedback-injection/regeneration prompt for MulFeRL on math benchmarks as those used in training, ensuring consistency with the training setup and facilitating reproducibility. Feedback merging is not required in this setting because these evaluation experiments do not aggregate feedback from multiple sources or multiple feedback comments; instead, they use a single-turn, feedback-guided regeneration protocol. For OOD evaluations, we only adjust the output-format constraints to match each task's requirements, while keeping the overall prompt structure and interaction protocol the same. In the appendix, we present the prompts for math tasks as a representative example.

C.1 Training System Prompt

This system prompt specifies the base role and output schema for policy rollouts during training: the model must wrap all reasoning inside a single `<thinking>` block and include a single `<feedback>` block at the beginning of `<thinking>` to standardize the response structure. This schema is also used by the verifier for format compliance. See Figure 6.

C.2 Feedback Simulator System Prompt

This system prompt is used by the feedback simulator $\mathcal{F}_\psi$ and is only triggered in the FAILED regime (Algorithm 1, lines 6–15). Given the original prompt x and a set of failed rollouts (we pass the *full* model outputs, including tags such as `<thinking>`, `<feedback>`, and the final answer field), the feedback simulator produces one concise `<feedback>` block that (i) pinpoints the earliest/root mistake, (ii) quotes where the reasoning first goes wrong, and (iii) provides step-by-step correction suggestions. Importantly, it must not provide a full corrected solution or reveal the final numeric answer. In our two-step aggregation (Appendix A.4), we apply this prompt to each subgroup to obtain subgroup feedbacks $\{f_j^{(t)}\}_{j=1}^{g}$. See Figure 7.

C.3 Merge Feedback System Prompt

This system prompt implements the second step of feedback aggregation (Appendix A.4): it summarizes and merges multiple subgroup feedback comments $\{f_j^{(t)}\}_{j=1}^{g}$ into a single concise and actionable feedback $f^{(t)}$ with the same (Issue, Fix steps) structure used for injection. It deduplicates repeated points and keeps only the earliest/root failure modes and their immediate consequences, without providing a full solution or revealing the final answer. See Figure 8.

C.4 Feedback Injection and Regeneration Prompt

This prompt template is used for feedback-guided regeneration when transitioning from turn t to $t{+}1$ (Algorithm 1, lines 12–14). It prepends the external feedback verbatim and forces the model to continue generation from the exact prefix `<thinking><feedback>{feedback}</feedback>`, ensuring the next rollout conditions on $f^{(t)}$ under standard next-token prediction. The template also forbids generating a new feedback block or paraphrasing the provided feedback, so that the injected feedback span is well-defined for token masking during optimization (Appendix A.5). See Figure 9.

D Implementation Details

This section reports the key implementation choices and key hyperparameters used in our experiments. Our implementation is built on the `verl` (Sheng et al., 2025) framework. We only report the most influential hyperparameters here; the full configuration can be found in the code repository referenced in the main text.

Our experiments use publicly available reasoning benchmarks, including math, science, and general reasoning tasks, and do not involve collect-

ing new personal data. These benchmarks are not intended to contain personally identifying information. The generated feedback is restricted to task-level error analysis and does not require or introduce personal information.

D.1 Training Details

D.1.1 RL-Training Details

Table 4 summarizes the key hyperparameters for MulFeRL. We mainly employ GPT-4o as the feedback simulator ψ. For fair comparison, all reinforcement learning-based finetuning baselines share the same backbone, training data, rollout group size, sampling temperature, length limits, number of training steps, and feedback provider whenever applicable. Method-specific objectives are kept faithful to the original algorithms.

For Critique-GRPO (Zhang et al., 2025), we also use GPT-4o as the feedback simulator and follow the original setup, sampling 7 rollouts in the first turn and 1 rollout in the second turn. For RLTF-FM (Song et al., 2026), we follow the feedback-modeling variant of RL from Text Feedback. RLTF assumes that text feedback is available during training but not at inference, so the model must internalize feedback into its single-turn behavior. RLTF-FM implements this by adding an auxiliary objective that predicts the feedback text, i.e., the critique, conditioned on the prompt and sampled response, in addition to the standard RL objective. We instantiate this objective with GPT-4o-generated feedback and use the same rollout budget and feedback budget as other feedback-based baselines. The feedback-modeling loss is applied only to the feedback tokens, with coefficient $\lambda_{\mathrm{FM}} = 0.2$. At test time, RLTF-FM is evaluated in the same single-pass setting without external feedback unless explicitly stated otherwise.

We detail MulFeRL's prompting pipeline as follows: the initial rollout prompt is given in Figure 6; the feedback-request prompt used to query ψ is in Figure 7, the feedback-merge (application) prompt is in Figure 8; and the prompt used for feedback injection and feedback-conditioned regeneration is in Figure 9. For other RL baselines, including GRPO, Dr.GRPO, Critique-GRPO, and RLTF-FM, we do not enforce MulFeRL's structured output format; instead, we follow their default generation interface and, when applicable, use the prompting setup from previous work (Zhang et al., 2025). During training, MulFeRL receives reward 1 only when both the output format and the final answer are correct (and 0 otherwise), whereas for other RL methods the reward is 1 if the final answer is correct (and 0 otherwise).

Parameter	Value
Optimization	
Global batch size	16
Per-device micro-batch	2
Learning rate	1×10^{-6}
Total steps	400
GRPO / KL / Entropy	
KL coef β_{KL}	0.001
KL type	`low_var_kl`
Entropy coefficient	0
FCO (cross-state)	
FCO weight λ_{FCO}	0.01
FCO inverse temperature τ	0.005
Sampling / Multi-turn Regeneration	
Group size K	8
Temperature	1.0
Max prompt length	2048
Max response length	8192
Max feedback length	1024
Max regeneration turns	2
Feedback subgroup size K_{fb}	2
Feedback subgroups g	$\lceil 8/2 \rceil = 4$

Table 4: Key hyperparameters for MulFeRL.

D.1.2 Self-Distillation Details

We implement SDPO (Hübotter et al., 2026) as a self-distillation baseline with the same rollout and feedback budget as the RL baselines. SDPO converts textual feedback into dense token-level supervision by comparing the no-feedback policy with a feedback-conditioned teacher distribution. In the original formulation, SDPO treats a feedback-conditioned policy as a self-teacher and distills its feedback-informed next-token predictions back into the no-feedback policy; this provides token-level credit assignment without requiring an external reward model or an additional refinement rollout.

In our implementation, we maintain an EMA-updated copy of the student as the feedback-conditioned teacher. Concretely, for each training prompt x, the trainable student policy first samples an initial response y under the no-feedback context. We then query GPT-4o to obtain textual feedback c for the sampled response. The EMA teacher is conditioned on the feedback-augmented context (x, y, c) and re-scores the same response tokens in y; gradients are stopped through this teacher

Parameter	Value
Feedback provider	GPT-4o
Initial rollout group size K	8
Temperature	1.0
Max prompt length	2048
Max response length	8192
Max feedback length	1024
Total RL steps	400
RL reward	final-answer correctness
Feedback-modeling coefficient λ_{FM}	0.2
Feedback-modeling target	GPT-4o critique tokens
Structured MulFeRL format	no
FCO objective	no
Test-time external feedback	disabled by default

Table 5: Additional hyperparameters for RLTF-FM. Shared RL optimization and sampling settings follow Table 4.

distribution. The trainable student is optimized under the original no-feedback context to match the feedback-conditioned teacher distribution on the sampled response tokens. After each optimization step, the teacher parameters are updated as $\theta_{\mathrm{tea}} \leftarrow \mu\theta_{\mathrm{tea}} + (1-\mu)\theta_{\mathrm{stu}}$.

SDPO uses the same training prompts, rollout group size, sampling temperature, maximum prompt length, maximum response length, maximum feedback length, and total training steps as the other feedback-based baselines. Unlike MulFeRL, SDPO does not perform feedback-guided regeneration, does not use the structured MulFeRL output format, and does not use the FCO objective. At test time, SDPO is evaluated in the same single-pass setting without external feedback unless explicitly stated otherwise.

D.1.3 SFT-Training Details

We include several supervised finetuning baselines (e.g., SFT, RAFT, and CITL-FT), which are trained with a standard supervised objective. Table 7 summarizes the key hyperparameters used for all SFT-style runs. Figure 10, Figure 11, and Figure 12 summarize the prompts used for the supervised finetuning baselines.

D.2 Evaluation Details

We evaluate with a decoding temperature of 1.0 and keep the maximum input/output lengths identical to those used during training. For each dataset, we report the mean performance over 5 independent runs.

Parameter	Value
Feedback provider	GPT-4o
Student initialization	backbone checkpoint
Teacher initialization	backbone checkpoint
Teacher update	EMA after each optimizer step
EMA decay μ	0.999
Initial rollout group size K	8
Temperature	1.0
Max prompt length	2048
Max response length	8192
Max feedback length	1024
Total training steps	400
Distillation target	feedback-conditioned teacher logits
Teacher context	prompt + sampled response + feedback
Student context	prompt only
Extra refinement sampling	no
Structured MulFeRL format	no
FCO objective	no
Test-time external feedback	disabled by default

Table 6: Key hyperparameters for SDPO. We use an EMA-updated copy of the student as the feedback-conditioned teacher. Shared sampling and length settings follow Table 4.

Parameter	Value
Max Prompt Length	2048
Max Response Length	8192
Per-device micro-batch	4
Learning rate	1×10^{-5}
Total steps	600

Table 7: Key hyperparameters for SFT training (if applicable).

D.2.1 Shared Inference Setup

Backbones and feedback simulator. We use Qwen3-4B-Instruct-2507 (Team, 2025) and Qwen2.5-7B-Base (Yang et al., 2024) as backbones. When external feedback is enabled, GPT-4o (OpenAI, 2024) serves as the feedback simulator ψ unless otherwise stated.

Decoding and length limits. We use a unified decoding configuration across methods for fair comparison. We cap the maximum generation length to 8k tokens and keep the stopping criteria consistent across all evaluated models.

Scoring and evaluation. We report Pass@1 using a single model output per instance, and report

results using the best checkpoint selected on the validation set for each method. Unless explicitly stated, evaluation is *single-pass* and does not use any external feedback at test time. We use each benchmark's official or commonly adopted evaluation scripts/format checks for verification. Unless explicitly stated, for MulFeRL we enforce the fixed output format shown in Figure 2: when feedback is available, it is explicitly placed inside a dedicated `<feedback>` tag within the `<thinking>` block; when feedback is absent, the model may either generate self feedback in the same tag or leave it empty (depending on the model's behavior). This structured format is used to remain consistent with the output convention during training. For other RL methods, we follow their default generation interface and evaluate them using the benchmark-provided evaluation scripts.

D.2.2 Main Results (Table 1)

All methods are evaluated in the single-pass setting described in § D.2.1. No external feedback is used at test time. We keep decoding, maximum length identical across methods. For all methods, we employ the benchmark-provided (or commonly used) answer verification scripts; in addition, MulFeRL applies a training-consistent formatting constraint.

D.2.3 Ablation Study (Table 2)

We evaluate all ablated variants under the same single-pass setup. The ablations differ only in the algorithmic components enabled during finetuning:

- **w/o Regeneration:** disables multi-turn regeneration; the learned policy is evaluated as usual in a single pass.
- **w/o FCO:** removes the cross-state FCO branch; evaluation remains single-pass.
- **w/o Feedback injection:** replaces structured injection with a plain prompting variant that provides the same feedback content; evaluation remains single-pass.

D.2.4 Impact of Feedback Simulators (Table 3)

For all methods, we study the effect of the feedback simulator by changing only ψ during training while keeping the remaining training and evaluation settings fixed. At evaluation time, we use the same single-pass protocol without external feedback, so the reported differences reflect how simulator choice affects the learned policy.

D.2.5 Test-time Multi-turn Feedback (Figure 3)

To evaluate test-time feedback utilization, we enable multi-turn interaction during inference. For MulFeRL, we query ψ for feedback using the template in Figure 7 (single rollout; no merging), and perform feedback-conditioned regeneration using the prompt in Figure 9. For GRPO and Critique-GRPO, we follow the prompting setup in previous work (Zhang et al., 2025) for both feedback elicitation and critique-guided regeneration. In all cases, feedback is constrained to identify errors and suggest repair steps without revealing the correct answer. We report Pass@1 after each revision turn.

D.2.6 Training Efficiency Measurement

All methods are trained under the same hardware/-software stack and data pipeline described in the main implementation details.

D.2.7 Impact of Multi-turn Training (Turn Budget Ablation)

We vary the per-step regeneration turn budget $T_{\text{train}} \in \{0, 1, 2, 3, 4, 5\}$, while keeping other hyperparameters fixed.

E Datasets Details

Table 8 summarizes the datasets used in our experiments and provides the corresponding source links.

F Additional Experiments

We organize the additional experiments around four questions: whether MulFeRL's gains can be explained by formatting or teacher access, which components and feedback sources matter, how robust the method is to imperfect feedback, and whether the gains persist under larger compute and across independent runs. Unless otherwise stated, we follow the evaluation protocol in Appendix D.2.

F.1 Run-to-run Stability

Table 9 reports mean and standard deviation over independent runs for the main comparison. The improvements of MulFeRL are consistent across runs and across both backbones, rather than being driven by a small number of favorable seeds. The standard deviations are comparable to the surrounding baselines, while the average performance remains higher on the main in-domain and out-of-domain evaluations. This supports the statistical stability of the reported gains.

Domain	Dataset	Link
Training	OpenR1-Math-220k	https://huggingface.co/datasets/open-r1/OpenR1-Math-220k
Math eval	MATH-500	https://huggingface.co/datasets/HuggingFaceH4/MATH-500
Math eval	Minerva-Math	https://huggingface.co/datasets/math-ai/minervamath
Math eval	OlympiadBench	https://huggingface.co/datasets/Hothan/OlympiadBench
Math eval	AIME 2024	https://huggingface.co/datasets/HuggingFaceH4/aime_2024
Math eval	AMC 2023	https://huggingface.co/datasets/AI-MO/aimo-validation-amc
Science & General eval	MMLU-Pro	https://huggingface.co/datasets/TIGER-Lab/MMLU-Pro
Science & General eval	TheoremQA	https://huggingface.co/datasets/TIGER-Lab/TheoremQA
Science & General eval	GPQA-Diamond	https://huggingface.co/datasets/Idavidrein/gpqa

Table 8: Datasets used in our experiments.

Method	Math (ID)					Science & General (OOD)			Avg.
	AMC23	AIME24	Olympiad Bench	MATH 500	Minerva Math	MMLU Pro	GPQA Diamond	Theorem QA	
Qwen2.5-7B-Base									
Base Model	34.50 ± 1.12	13.33 ± 2.36	28.13 ± 0.94	55.24 ± 0.67	18.31 ± 1.25	45.06 ± 0.76	27.68 ± 1.03	19.00 ± 1.16	30.16 ± 0.76
SFT	38.50 ± 1.37	12.00 ± 1.83	27.27 ± 1.09	57.12 ± 0.50	21.03 ± 0.89	47.20 ± 0.91	28.08 ± 0.78	24.10 ± 1.12	31.91 ± 0.60
RAFT	46.50 ± 1.37	9.33 ± 1.49	29.58 ± 1.08	61.92 ± 0.73	17.50 ± 1.30	46.06 ± 0.79	23.94 ± 1.09	21.90 ± 0.96	32.09 ± 0.78
CITL-FT	41.50 ± 1.37	15.33 ± 1.83	30.89 ± 0.87	63.00 ± 0.54	19.34 ± 1.11	47.86 ± 0.77	27.07 ± 0.85	23.65 ± 1.00	33.58 ± 0.62
SDPO	41.50 ± 1.37	14.67 ± 1.83	35.91 ± 0.98	69.72 ± 0.72	28.24 ± 1.08	50.90 ± 0.80	32.53 ± 0.97	37.20 ± 1.14	38.83 ± 0.66
GRPO	42.00 ± 1.12	16.00 ± 1.49	36.77 ± 1.04	70.84 ± 0.76	28.97 ± 0.87	51.10 ± 0.66	33.43 ± 1.03	37.55 ± 1.19	39.58 ± 0.68
Dr.GRPO	41.00 ± 1.37	14.67 ± 1.83	35.79 ± 0.80	73.20 ± 0.50	30.37 ± 1.16	51.88 ± 0.93	33.13 ± 0.84	40.08 ± 0.87	40.02 ± 0.64
Critique-GRPO	45.50 ± 1.12	20.67 ± 1.49	38.64 ± 0.97	74.96 ± 0.63	33.82 ± 0.82	52.34 ± 0.72	36.97 ± 0.97	39.75 ± 1.08	42.83 ± 0.60
RLTF-FM	42.00 ± 1.12	15.33 ± 1.83	36.05 ± 0.95	69.88 ± 0.70	28.46 ± 1.02	50.76 ± 0.78	32.93 ± 0.96	37.65 ± 1.13	39.13 ± 0.65
MulFeRL	**50.00 ± 1.77**	**24.00 ± 1.49**	**42.49 ± 0.82**	**78.72 ± 0.55**	**37.87 ± 1.08**	**54.89 ± 0.78**	**38.99 ± 0.84**	**43.08 ± 1.03**	**46.26 ± 0.65**
Qwen3-4B-Inst									
Base Model	59.00 ± 1.37	46.00 ± 1.49	45.13 ± 0.97	76.68 ± 0.57	42.43 ± 0.75	58.56 ± 0.76	35.05 ± 0.81	39.85 ± 0.94	50.34 ± 0.60
SFT	62.50 ± 1.77	47.33 ± 1.49	49.17 ± 0.93	78.12 ± 0.45	46.32 ± 0.98	60.46 ± 0.72	36.26 ± 0.60	41.93 ± 0.79	52.76 ± 0.54
RAFT	62.00 ± 1.12	46.00 ± 1.49	48.34 ± 1.08	78.68 ± 0.50	45.29 ± 0.72	59.53 ± 0.57	37.47 ± 0.85	41.05 ± 0.91	52.30 ± 0.58
CITL-FT	62.00 ± 1.12	48.00 ± 1.83	48.55 ± 0.77	79.64 ± 0.53	47.13 ± 0.96	60.18 ± 0.68	36.87 ± 0.78	42.65 ± 0.67	53.13 ± 0.55
SDPO	79.50 ± 1.12	58.00 ± 1.83	59.55 ± 0.86	87.80 ± 0.47	50.07 ± 0.98	63.00 ± 0.72	43.84 ± 0.80	50.30 ± 0.88	61.51 ± 0.56
GRPO	78.50 ± 1.37	57.33 ± 1.49	59.05 ± 0.89	87.84 ± 0.52	51.25 ± 1.03	62.18 ± 0.55	43.94 ± 0.72	49.85 ± 0.99	61.24 ± 0.62
Dr.GRPO	79.00 ± 1.37	56.67 ± 2.36	60.36 ± 0.71	88.28 ± 0.34	49.19 ± 1.12	63.33 ± 0.84	42.83 ± 0.95	50.68 ± 0.76	61.29 ± 0.57
Critique-GRPO	84.00 ± 1.37	62.67 ± 1.49	62.40 ± 0.79	88.64 ± 0.49	50.29 ± 0.77	64.67 ± 0.67	45.45 ± 0.69	51.05 ± 0.95	63.65 ± 0.54
RLTF-FM	82.50 ± 1.77	60.67 ± 1.49	61.54 ± 0.82	88.12 ± 0.46	49.63 ± 0.90	64.25 ± 0.70	44.65 ± 0.75	50.85 ± 0.91	62.78 ± 0.55
MulFeRL	**90.50 ± 1.12**	**68.00 ± 1.83**	**68.13 ± 0.73**	**90.24 ± 0.43**	**55.96 ± 0.95**	**68.08 ± 0.69**	**50.20 ± 0.80**	**55.75 ± 0.93**	**68.36 ± 0.59**

Table 9: Results reported as mean ± standard deviation over 5 independent runs, where each run is evaluated at its best validation checkpoint.

F.2 Controlled Baseline Comparisons

F.2.1 Unified Output Format

To control for confounding effects caused by differences in output formatting, we align the prompts and output constraints of GRPO and Critique-GRPO to exactly match those used by MulFeRL. Specifically, both baselines use the same input prompt (Figure 6) as MulFeRL; for Critique-GRPO, the prompts for getting feedback (Figure 7) and for feedback injection (Figure 9) are also identical to MulFeRL. Finally, we enforce the same structured output-format constraint as in Figure 2 for all methods. The corresponding results are reported in Table 10.

Under this controlled setting, GRPO shows a slight performance drop compared to the main-table results. A plausible explanation is that the structured format introduces additional generation overhead (consuming token budget) and partially limits the model's freedom to organize reasoning and intermediate steps, reducing solution flexibility on harder instances. In addition, strict formatting can increase the risk of format violations or truncation, which lowers the fraction of valid outputs.

Critique-GRPO also underperforms its original numbers. Although it employs feedback injection, the amount of injectable feedback is relatively limited in this setting, preventing the model from fully benefiting from critique signals. Meanwhile, the structured template and strict formatting further constrain how the model organizes reasoning and intermediate steps, and can increase the likelihood of format violations/truncation. Together, these effects can shrink the effective solution space on complex problems and reduce the valid-output rate, thereby offsetting and potentially outweighing the

gains from feedback. Overall, this alignment study suggests that a unified output format alone does not account for our improvements; MulFeRL's gains are more fundamentally driven by feedback-guided iterative refinement and its associated optimization objectives.

F.2.2 Same-budget GPT-4o Distillation

Table 11 compares MulFeRL against a GPT-4o distillation baseline under approximately matched training-time GPT-4o API cost. Across both backbones, MulFeRL is consistently stronger than the distillation baseline on both in-domain mathematical reasoning and out-of-domain reasoning benchmarks. This suggests that the improvement is not simply due to querying GPT-4o or exposing the student to stronger model outputs. Instead, the benefit comes from how MulFeRL uses external feedback: the model diagnoses and revises its own failed reasoning, and the verifier then turns successful feedback-guided revisions into RL supervision. In contrast, distillation mainly encourages imitation of teacher-generated outputs, which can be less aligned with preserving and improving the student's own reasoning behavior.

F.3 Component and Simulator Analyses

F.3.1 Component Ablations

The complete ablation results are reported in Table 12. The conclusions are consistent with the main experiments. Performance degrades across all ablations, indicating that each component contributes materially to MulFeRL. Disabling multi-turn regeneration yields the largest drop, consistent with the observation that single-turn scalar rewards provide a weak learning signal on failed samples. Removing the FCO branch while keeping regeneration also leads to a clear decline, suggesting that regeneration alone does not reliably convert feedback-induced solved-state transitions into a stable training signal: when feedback moves an all-failed group to an all-solved group, within-state GRPO has no reward contrast, and cross-state FCO is needed to consolidate the improvement direction. Finally, ablating feedback injection consistently hurts performance, implying that structured injection conditions the model's reasoning on feedback more faithfully than plain prompting, improving feedback utilization and stabilizing training. We omit Qwen2.5-7B-Base as a simulator because, without instruction tuning, it frequently violates the required output schema, making feedback extraction difficult.

F.3.2 Feedback Simulators

The complete results are given in Table 13. The same takeaway as in the main paper holds: simulator quality strongly correlates with the final performance of MulFeRL. As shown in Table 13, replacing the feedback source with a more capable simulator consistently improves downstream accuracy. Notably, even using the training backbone itself as the simulator is competitive and surpasses a weaker small model, indicating that MulFeRL can effectively leverage reasonably good self-generated critiques and is not tied to a specific simulator. Moving to stronger external simulators yields additional gains: GPT-4o-mini provides a clear improvement over self-feedback, and GPT-4o further increases performance. A plausible explanation is that higher-quality simulators produce more precise failure localization and more targeted, executable suggestions, which in turn guide regeneration toward better candidates and provide cleaner learning signals, leading to better generalization on difficult mathematical reasoning benchmarks.

F.3.3 Matched Train-time and Test-time Feedback Variants

Table 14 studies matched train-time and test-time feedback variants. For each feedback variant, methods are trained under the same setup as in the main experiments except for the feedback type, and are evaluated with one round of feedback-guided regeneration using the same feedback type as in training. Outcome-only feedback provides a much weaker learning and revision signal than full verbal feedback, showing that binary verdicts alone do not capture the diagnostic information needed for robust improvement. Full-verbal feedback is consistently strongest, and MulFeRL benefits more from it than Critique-GRPO, indicating that the MulFeRL feedback format and optimization objectives make better use of rich diagnostic feedback during training and regeneration.

For compactness, the feedback-variant tables use four short names. *Outcome-only* denotes binary verdict feedback only, *Wrong-verdict* denotes verbal feedback induced by a fully flipped verdict, *Corrupted-diagnostics* denotes verbal feedback with fully corrupted diagnostics, and *Full-verbal* denotes full issue-and-fix feedback. We use "w/o FI" for the ablation without structured feed-

Method	Math (ID)					Science & General (OOD)			Avg.
	AMC23	AIME24	Olympiad Bench	MATH 500	Minerva MATH	MMLU Pro	GPQA Diamond	Theorem QA	
Qwen2.5-7B-Base									
GRPO	39.50	16.00	33.68	67.40	26.32	48.40	30.20	34.60	37.01
SDPO	38.00	14.00	34.51	66.72	26.69	47.80	30.00	33.95	36.46
Critique-GRPO	43.50	20.67	36.23	72.44	31.47	50.09	34.44	37.45	40.79
RLTF-FM	38.50	14.67	34.75	66.80	26.25	48.10	30.51	34.35	36.74
MulFeRL	**50.00**	**24.00**	**42.49**	**78.72**	**37.87**	**54.89**	**38.99**	**43.08**	**46.26**
Qwen3-4B-Inst									
GRPO	75.50	57.33	55.58	84.72	48.53	59.13	41.31	46.15	58.53
SDPO	76.50	57.33	55.99	84.20	47.50	59.60	40.61	46.80	58.57
Critique-GRPO	82.00	62.67	59.53	86.20	48.01	61.92	43.03	48.50	61.48
RLTF-FM	79.00	59.33	57.63	85.36	47.57	60.98	42.32	47.95	60.02
MulFeRL	**90.50**	**68.00**	**68.13**	**90.24**	**55.96**	**68.08**	**50.20**	**55.75**	**68.36**

Table 10: Evaluation results (Pass@1) on mathematical reasoning (ID) and scientific/general reasoning (OOD) benchmarks under a unified MulFeRL-style feedback-injection output format across all methods.

Method	Math (ID)					Science & General (OOD)			Avg.
	AMC23	AIME24	Olympiad Bench	MATH 500	Minerva MATH	MMLU Pro	GPQA Diamond	Theorem QA	
Qwen2.5-7B-Base									
GPT-4o Distillation	47.61	21.33	40.52	77.58	35.27	52.71	37.11	40.76	44.11
MulFeRL	**50.00**	**24.00**	**42.49**	**78.72**	**37.87**	**54.89**	**38.99**	**43.08**	**46.26**
Qwen3-4B-Inst									
GPT-4o Distillation	88.18	63.67	65.74	89.33	53.76	66.49	48.88	53.71	66.22
MulFeRL	**90.50**	**68.00**	**68.13**	**90.24**	**55.96**	**68.08**	**50.20**	**55.75**	**68.36**

Table 11: Comparison between GPT-4o distillation and MulFeRL when the distillation baseline is constrained to use approximately the same *training-time GPT-4o API cost in USD* as MulFeRL. All results are reported as 5-run means, where each run is evaluated at its best validation checkpoint.

back injection.

F.3.4 Robustness Without Structured Feedback Injection

All methods use the same trained checkpoints as in the main experiments; only at test time do we provide one round of feedback-guided regeneration with the specified feedback variant. Table 15 repeats the compact feedback-variant analysis for the ablation without structured feedback injection. Full-verbal feedback still provides the strongest revisions, but wrong-verdict and corrupted-diagnostics feedback cause larger degradation than clean feedback. This supports the role of structured injection: placing feedback inside the reasoning interface helps the policy use useful diagnostics more faithfully and reduces sensitivity to severely misleading feedback. The result also clarifies that robustness does not come merely from applying another test-time revision round; it depends on how feedback was integrated during training and inference.

F.3.5 Test-time Feedback Variants

All methods use the same trained checkpoints as in the main experiments; only at test time do we provide one round of feedback-guided regeneration with the specified feedback variant. Table 16 isolates the effect of test-time feedback by keeping the trained checkpoints fixed and varying only the feedback available during one regeneration round. Full-verbal feedback gives the strongest revisions, whereas outcome-only verdicts provide much weaker guidance because they do not explain how to repair the failed solution. Wrong-verdict feedback consistently hurts, and fully corrupted diagnostics also degrade MulFeRL, confirming that feedback quality matters. Under clean full-verbal feedback, MulFeRL remains strongest across backbones and evaluation groups.

F.4 Training Dynamics and Compute

F.4.1 Convergence and Validation Curves

Figure 13 plots the average performance on five math benchmarks as training progresses. We ob-

Method	Math (ID)					Science & General (OOD)			Avg.
	AMC23	AIME24	Olympiad Bench	MATH 500	Minerva MATH	MMLU Pro	GPQA Diamond	Theorem QA	
Qwen2.5-7B-Base									
MulFeRL	**50.00**	**24.00**	**42.49**	**78.72**	**37.87**	**54.89**	**38.99**	**43.08**	**46.26**
w/o Regeneration	42.20	16.00	36.77	70.84	28.97	51.10	33.43	37.55	39.61
w/o FCO	46.20	19.67	39.38	75.40	32.13	52.24	35.15	39.45	42.45
w/o Feedback injection	47.10	20.67	40.39	76.56	34.34	53.09	36.87	41.18	43.78
Qwen3-4B-Inst									
MulFeRL	**90.50**	**68.00**	**68.13**	**90.24**	**55.96**	**68.08**	**50.20**	**55.75**	**68.36**
w/o Regeneration	78.60	57.33	59.05	87.84	51.25	62.18	43.94	49.85	61.26
w/o FCO	87.70	65.67	65.79	88.08	53.16	66.36	47.68	53.20	65.96
w/o Feedback injection	88.50	66.33	66.62	88.76	54.04	66.97	48.79	54.10	66.76

Table 12: Ablation of MulFeRL components on Qwen3-4B-Inst and Qwen2.5-7B-Base. Numbers are Pass@1 (%).

Feedback simulator	Math (ID)					Science & General (OOD)			Avg.
	AMC23	AIME24	Olympiad Bench	MATH 500	Minerva Math	MMLU Pro	GPQA Diamond	Theorem QA	
Backbone: Qwen2.5-7B-Base									
Qwen3-1.7B	47.10	20.00	38.78	76.92	33.75	53.09	36.06	41.08	43.35
GPT-4o-mini	49.20	21.67	41.57	78.08	36.18	54.31	37.78	42.25	45.13
GPT-4o	**50.00**	**24.00**	**42.49**	**78.72**	**37.87**	**54.89**	**38.99**	**43.08**	**46.26**
Backbone: Qwen3-4B-Inst									
Qwen3-1.7B	85.60	62.33	58.81	87.76	48.75	65.86	45.15	52.65	63.36
Self (base checkpoint)	88.00	64.00	63.62	88.68	51.84	66.92	46.77	53.80	65.45
GPT-4o-mini	89.30	66.33	66.94	89.40	52.65	67.58	48.79	54.75	66.97
GPT-4o	**90.50**	**68.00**	**68.13**	**90.24**	**55.96**	**68.08**	**50.20**	**55.75**	**68.36**

Table 13: MulFeRL trained with different feedback simulators on Qwen3-4B-Inst and Qwen2.5-7B-Base.

serve that MulFeRL improves more rapidly in the early stage and stabilizes at a strong level in substantially fewer optimization steps than GRPO and Critique-GRPO across both backbones. We attribute this faster convergence to two design choices. First, feedback-guided regeneration can often refine otherwise unsuccessful rollouts into higher-quality candidates before parameter updates, thereby reducing the extent to which the learning signal is dominated by low-quality samples. Second, cross-state FCO extracts additional supervision from verifier-certified feedback-induced transitions, providing denser and more informative learning signals per prompt.

Method	Math (ID)					Science & General (OOD)			Avg. (mean ± std.)
	AMC23	AIME24	Olympiad Bench	MATH 500	Minerva MATH	MMLU Pro	GPQA Diamond	Theorem QA	
Qwen2.5-7B-Base									
GRPO + Full-verbal	44.42	22.67	39.18	71.68	31.54	53.64	35.52	51.96	43.83 ± 0.45
Critique-GRPO + Outcome-only	45.16	21.33	38.64	74.90	33.70	52.50	36.76	40.92	42.99 ± 0.46
Critique-GRPO + Full-verbal	47.98	25.33	41.12	75.84	35.74	54.42	38.68	54.37	46.69 ± 0.54
MulFeRL + Outcome-only	44.98	20.67	39.74	74.02	32.88	53.95	36.61	41.55	43.05 ± 0.47
MulFeRL + Full-verbal	**54.08**	**30.67**	**46.72**	**79.82**	**41.44**	**58.77**	**43.81**	**58.42**	**51.72 ± 0.66**
Qwen3-4B-Inst									
GRPO + Full-verbal	81.18	64.67	62.18	88.88	53.76	64.86	46.12	56.41	64.76 ± 0.38
Critique-GRPO + Outcome-only	83.82	62.67	62.42	88.64	50.24	64.62	45.34	51.46	63.65 ± 0.43
Critique-GRPO + Full-verbal	86.62	68.00	65.76	89.48	52.98	66.84	48.18	58.52	67.05 ± 0.50
MulFeRL + Outcome-only	82.34	62.00	62.88	89.01	53.74	64.98	46.92	53.18	64.38 ± 0.43
MulFeRL + Full-verbal	**93.75**	**73.33**	**71.89**	**91.09**	**58.98**	**71.33**	**54.21**	**61.82**	**72.05 ± 0.53**

Table 14: Compact matched train-time and test-time feedback variants. For MulFeRL and Critique-GRPO, the feedback type used during training is matched to the feedback type used during one feedback-guided regeneration round at test time. GRPO has no feedback-conditioned training and is included with full-verbal test-time feedback as an anchor. Variant names are defined in the text. Values are 5-run means, with standard deviations reported for averages.

Method	Math (ID)					Science & General (OOD)			Avg. (mean ± std.)
	AMC23	AIME24	Olympiad Bench	MATH 500	Minerva MATH	MMLU Pro	GPQA Diamond	Theorem QA	
Qwen2.5-7B-Base									
MulFeRL w/o FI + Outcome-only	47.28	21.33	40.58	76.92	34.36	53.18	36.98	46.12	44.59 ± 0.61
MulFeRL w/o FI + Wrong-verdict	43.24	17.33	37.42	75.74	31.62	50.24	34.28	34.86	40.59 ± 1.04
MulFeRL w/o FI + Corrupted-diagnostics	44.04	18.67	38.32	75.92	32.32	50.86	34.88	37.08	41.51 ± 0.97
MulFeRL w/o FI + Full-verbal	**52.18**	**24.67**	**43.96**	**79.10**	**37.02**	**55.92**	**39.52**	**56.60**	**48.62 ± 0.66**
Qwen3-4B-Inst									
MulFeRL w/o FI + Outcome-only	88.98	67.33	66.90	88.94	54.34	67.14	49.10	54.88	67.20 ± 0.55
MulFeRL w/o FI + Wrong-verdict	85.96	63.33	63.88	88.22	52.12	64.44	46.82	46.20	63.87 ± 1.07
MulFeRL w/o FI + Corrupted-diagnostics	86.88	64.67	65.02	88.38	52.78	65.22	47.42	47.40	64.72 ± 0.98
MulFeRL w/o FI + Full-verbal	**92.18**	**70.67**	**69.82**	**90.68**	**56.72**	**69.88**	**51.42**	**59.82**	**70.15 ± 0.65**

Table 15: Compact test-time feedback variants for **MulFeRL w/o feedback injection**. We write "w/o FI" for removing structured feedback injection; variant names are defined in the text. All feedback-conditioned variants use one round of feedback-guided regeneration at test time. Values are 5-run means, with standard deviations reported for averages.

Training system prompt

```
You are a reasoning assistant.\n
Solve the problem step by step.\n\n
You MUST follow this exact output format for every problem:\n
1) Wrap ALL your reasoning inside a single <thinking>...</thinking> block.\n
2) At the very start of <thinking>, output exactly ONE <feedback>...</feedback> block.\n
- The <feedback> must be concise and actionable, and use this structure:\n
<feedback>\n
Issue:\n
1. Likely pitfalls: ...\n
2. Step-by-step plan:\n
- (1) ...\n
- (2) ...\n
- (3) ...\n\n
Fix steps:\n
1. ...\n
2. ...\n
3. ...\n
</feedback>\n
- Give guidance / a repair plan, but do NOT give a full solution inside <feedback>.\n
- Do NOT output any expression inside <feedback> that directly equals the final result.\n
(e.g., do NOT write something like '... = <final answer>' inside <feedback>.)\n
- You MAY include tiny snippets (a short identity, a one-line correction),\n
but avoid long derivations or long equations in <feedback>.\n
3) After </thinking>, on a new line, output the final numeric answer in the format:\n
\boxed{answer}\n
Do NOT add any extra text after the boxed answer.\n\n
Example (format only):\n
<thinking>\n
<feedback>\n
Issue:\n
1. Likely pitfalls: Misreading quantities; forgetting to combine changes.\n
2. Step-by-step plan:\n
- (1) Identify the initial quantity and each change.\n
- (2) Choose the correct operation (add/subtract/etc.).\n
- (3) Compute carefully.\n\n
Fix steps:\n
1. Extract numbers and what they represent.\n
2. Write the operation clearly.\n
3. Recompute the final arithmetic once.\n
</feedback>\n
Alice starts with 3 apples.\n
She buys 2 more apples.\n
Total apples = 3 + 2 = 5.\n
</thinking>\n
\boxed{5}\n
```

Figure 6: Training system prompt.

Obtain Feedback system prompt

```
You are a strict reviewer of an incorrect solution.\n
Please briefly explain (step-by-step) the solution's mistakes and provide your correction
suggestions.\n\n
Task:\n
- Identify the earliest/root mistake (the first wrong step).\n
- Explain the errors step-by-step (concise, specific).\n
- Give concrete correction suggestions (how to fix the process).\n\n
Rules:\n
- Do NOT provide a full correct solution.\n
- Do NOT provide or reveal the final numeric answer.\n
- Avoid generic advice; point to specific steps or lines.\n\n
Output ONLY:\n
<feedback>\n
Issue:\n
1. Earliest/root mistake: ...\n
2. Where it first goes wrong (quote 1-2 lines): ". . . "\n
3. Step-by-step errors:\n
- (1) ...\n
- (2) ...\n
- (3) ...\n\n
Fix steps:\n
1. ...\n
2. ...\n
3. ...\n
</feedback>
```

Figure 7: System prompt for obtaining feedback.

Merge feedback system prompt

```
You merge multiple feedback comments on an incorrect solution.\n\n
Task:\n
- Combine the feedback into ONE concise, actionable feedback.\n
- Deduplicate repeated points.\n
- Keep ONLY the earliest/root failure modes (the first wrong step and its consequences).\n\n
Rules:\n
- Do NOT provide a full correct solution.\n
- Do NOT provide or reveal the final numeric answer.\n
- Keep it short and specific.\n\n
Output ONLY:\n
<feedback>\n
Issue:\n
1. Earliest/root mistake: ...\n
2. Where it first goes wrong (quote 1-2 lines): ". . . "\n
3. Step-by-step errors:\n
- (1) ...\n
- (2) ...\n
- (3) ...\n\n
Fix steps:\n
1. ...\n
2. ...\n
3. ...\n
</feedback>
```

Figure 8: System prompt for merging feedback.

Feedback injection and regeneration rollout prompt

```
You are a problem reasoning assistant.\n
Re-solve the problem from scratch. Use the provided feedback only as hints.\n
Do NOT create another <feedback> block.\n
Do NOT modify, repeat, or paraphrase the provided feedback text.\n\n
Output format (must follow exactly):\n
1) Start with EXACTLY: <thinking><feedback>{feedback}</feedback>\n
2) Continue your reasoning, then close: </thinking>\n
3) The final answer MUST be written as: \boxed{{answer}}\n
Do NOT output anything else.\n\n
Problem:\n
{question}\n
Start your new solution by continuing from the following prefix exactly:\n
<thinking><feedback>{feedback}</feedback>
```

Figure 9: Prompt for feedback injection and regeneration rollout.

SFT/RAFT system prompt and CITL-FT initial prompt

```
You are a reasoning assistant.\n
Solve the problem step by step.\n\n
Output format (must follow exactly):\n
1) Wrap ALL reasoning inside a single <thinking>...</thinking> block.\n
2) After </thinking>, on a new line, output the final numeric answer in the format:\n
\boxed{answer}\n
Do NOT output any extra text after the boxed answer.\n
```

Figure 10: SFT/RAFT prompt and CITL-FT initial prompt

CITL-FT critique system prompt

```
You are a strict reviewer of a solution.\n
Given a problem and a proposed solution, identify mistakes and provide actionable repair
suggestions.\n\n
Rules:\n
- Do NOT provide a full correct solution.\n
- Do NOT reveal the final numeric answer.\n
- Be specific (point to the first wrong step and its consequence).\n\n
Output ONLY:\n
<critique>\n
1. Earliest/root mistake: ...\n
2. Why it is wrong: ...\n
3. Fix plan (high-level):\n
- (1) ...\n
- (2) ...\n
- (3) ...\n
</critique>\n
```

Figure 11: Critique prompt for CITL-FT

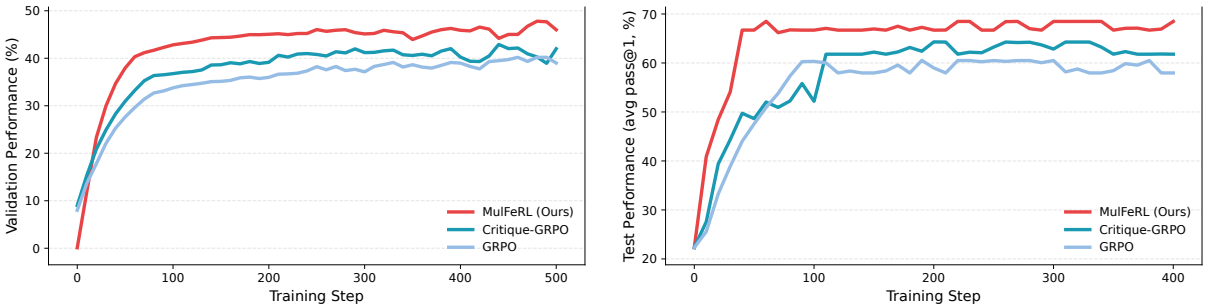
CITL-FT refinement system prompt

```
You are a reasoning assistant.\n
Re-solve the problem, using the provided critique as guidance.\n
Do NOT repeat the critique verbatim.\n\n
Output format (must follow exactly):\n
1) <thinking> ... </thinking>\n
2) \boxed{answer}\n
Do NOT output anything else.\n\n
Inputs:\n
Problem: {question}\n
Initial solution: {initial_response}\n
Critique: {critique}\n
```

Figure 12: Refinement prompt for CITL-FT

Method	**Math (ID)**					**Science & General (OOD)**			**Avg. (mean ± std.)**
	AMC23	**AIME24**	**Olympiad Bench**	**MATH 500**	**Minerva MATH**	**MMLU Pro**	**GPQA Diamond**	**Theorem QA**	
Qwen2.5-7B-Base									
GRPO + Outcome-only	42.56	20.00	37.62	71.04	30.42	52.26	34.46	43.12	41.44 ± 0.52
GRPO + Wrong-verdict	39.14	14.67	34.98	69.96	27.44	49.53	32.22	33.89	37.73 ± 0.79
GRPO + Full-verbal	44.42	22.67	39.18	71.68	31.54	53.64	35.52	51.96	43.83 ± 0.45
Critique-GRPO + Outcome-only	45.42	21.33	38.88	74.98	33.88	52.72	36.98	41.20	43.17 ± 0.48
Critique-GRPO + Wrong-verdict	42.02	17.67	36.06	73.96	31.22	50.14	34.66	36.83	40.32 ± 0.74
Critique-GRPO + Full-verbal	47.98	25.33	41.12	75.84	35.74	54.42	38.68	54.37	46.69 ± 0.52
MulFeRL + Outcome-only	50.66	25.33	43.36	78.96	38.48	55.68	40.02	45.94	47.30 ± 0.54
MulFeRL + Wrong-verdict	47.56	20.00	40.12	78.02	35.44	52.57	36.21	40.30	43.78 ± 0.90
MulFeRL + Corrupted-diagnostics	48.46	21.33	41.04	78.30	36.28	53.51	37.29	42.20	44.80 ± 0.83
MulFeRL + Full-verbal	**54.08**	**30.67**	**46.72**	**79.82**	**41.44**	**58.77**	**43.81**	**58.42**	**51.72 ± 0.66**
Qwen3-4B-Inst									
GRPO + Outcome-only	79.64	60.67	60.56	88.18	52.44	63.72	44.92	51.30	62.68 ± 0.48
GRPO + Wrong-verdict	77.12	54.67	57.98	87.16	49.72	60.48	42.66	39.84	58.70 ± 0.81
GRPO + Full-verbal	81.18	64.67	62.18	88.88	53.76	64.86	46.12	56.41	64.76 ± 0.38
Critique-GRPO + Outcome-only	84.02	63.33	62.66	88.70	50.48	64.82	45.56	51.76	63.92 ± 0.44
Critique-GRPO + Wrong-verdict	81.82	58.00	59.78	87.94	48.24	62.56	43.54	45.14	60.88 ± 0.78
Critique-GRPO + Full-verbal	86.62	68.00	65.76	89.48	52.98	66.84	48.18	58.52	67.05 ± 0.48
MulFeRL + Outcome-only	90.84	69.33	68.86	90.39	56.48	68.86	51.02	56.92	69.09 ± 0.51
MulFeRL + Wrong-verdict	88.21	64.67	66.01	89.73	54.16	66.43	47.73	52.60	66.19 ± 0.94
MulFeRL + Corrupted-diagnostics	89.01	66.00	66.79	89.99	54.80	67.19	48.67	53.84	67.04 ± 0.86
MulFeRL + Full-verbal	**93.75**	**73.33**	**71.89**	**91.09**	**58.98**	**71.33**	**54.21**	**61.82**	**72.05 ± 0.53**

Table 16: Compact test-time feedback variants for MulFeRL, GRPO, and Critique-GRPO. Variant names are defined in the text; all feedback-conditioned variants use one round of feedback-guided regeneration at test time. Values are 5-run means, with standard deviations reported for averages.



(a) Qwen2.5-7B-Base.

(b) Qwen3-4B-Inst.

Figure 13: Average Pass@1 over the five mathematical reasoning benchmarks across training checkpoints. MulFeRL is evaluated on both answer correctness and format compliance, while GRPO and Critique-GRPO are evaluated on answer correctness only.