R^{3}L: Reflect-then-Retry Reinforcement Learning with Language-Guided Exploration, Pivotal Credit, and Positive Amplification

Anonymous ACL submission

Abstract

Reinforcement learning drives recent advances in LLM reasoning and agentic capabilities, yet current approaches struggle with both exploration and exploitation. Exploration suffers from low success rates on difficult tasks and high costs of repeated rollouts from scratch. Exploitation suffers from coarse credit assignment and training instability: Trajectory-level rewards penalize valid prefixes for later errors, and failure-dominated groups overwhelm the few positive signals, leaving optimization without constructive direction. To this end, we propose R^{3}L, Reflect-then-Retry Reinforcement Learning with Language-Guided Exploration, Pivotal Credit, and Positive Amplification. To synthesize high-quality trajectories, R^{3}L shifts from stochastic sampling to active synthesis via reflect-then-retry, leveraging language feedback to diagnose errors, transform failed attempts into successful ones, and reduce rollout costs by restarting from identified failure points. With errors diagnosed and localized, Pivotal Credit Assignment updates only the diverging suffix where contrastive signals exist, excluding the shared prefix from gradient update. Since failures dominate on difficult tasks and reflect-then-retry produces off-policy data, risking training instability, Positive Amplification upweights successful trajectories to ensure positive signals guide the optimization process. Experiments on agentic and reasoning tasks demonstrate 5% to 52% relative improvements over baselines while maintaining training stability. Our code is released at https://anonymous.4open.science/r/R3L.

1 Introduction

Reinforcement learning (RL) like GRPO (Shao et al., 2024) drives recent advances in LLM reasoning and agentic capabilities (Cui et al., 2025; Shi et al., 2025; Plaat et al., 2025). Recent systems such as DeepSeek-R1 (Guo et al., 2025), DeepSeek-Math (Shao et al., 2024), and Search-R1 (Jin et al., 2025) demonstrate its effectiveness. However, as tasks scale to complex, multi-step agentic environments with sparse rewards, current approaches struggle with both exploration and exploitation.

Inefficient exploration. Stochastic sampling produces predominantly failed trajectories on difficult problems, leading to a lack of positive signals and even null gradients when all samples in a group fail (Nan et al., 2025). Repeated rollouts from scratch attempt to compensate but incur high cost. Furthermore, scalar rewards indicate correctness but provide no actionable guidance on why solutions failed or how to discover better ones (Scheurer et al., 2022, 2023; Zhang et al., 2025b). In practice, environments often provide abundant natural language feedback such as error messages, execution traces, and observation descriptions, yet current RL algorithms like GRPO cannot leverage this rich information. This calls for a guided exploration mechanism that leverages language feedback to efficiently synthesize successful trajectories.

Unstable exploitation. Beyond discovering successful trajectories, learning from them poses distinct challenges. Trajectory-level credit assignment applies the same reward signal to all tokens regardless of where errors occur. When a trajectory fails due to a late mistake, valid earlier reasoning is suppressed alongside the error, introducing noise into gradient estimates. While Process Reward Models (Lightman et al., 2023) offer step-level credit as an alternative, they require costly human annotation and often produce unstable supervision signals (Xiong et al., 2024), calling for credit assignment methods that derive clear signals without external annotation. In group-relative methods, instability arises when failed trajectories dominate a group, as gradients become driven by error suppression rather than reinforcement of correct solutions. Suppressing errors reduces the probability of incorrect tokens, releasing probability mass that must be redistributed. Without sufficient positive signals to

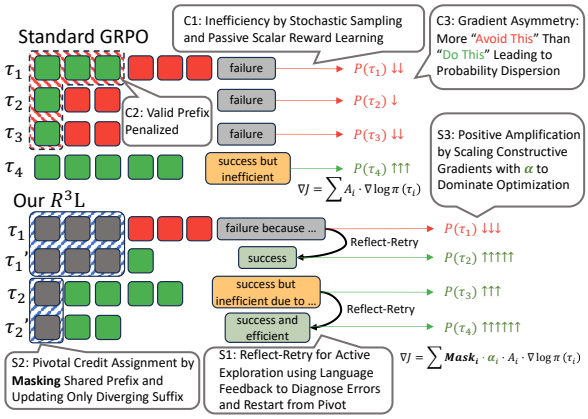


Figure 1: Comparison between standard RL (GRPO) and R^{3}L. Red blocks indicate erroneous steps, Green blocks indicate correct steps, and Gray blocks indicate masked prefix excluded from gradient updates. Standard RL suffers from (C1) inefficient stochastic sampling, (C2) valid prefix penalization, and (C3) gradient asymmetry due to failure dominance. R^{3}L addresses these via (S1) reflect-then-retry for active exploration, (S2) pivotal credit, and (S3) positive amplification. The detailed R^{3}L framework is illustrated in Figure 2.

direct this mass toward correct solutions, it disperses across the vocabulary, driving the policy toward high entropy. We term this entropy collapse and analyze it in Appendix A.2.1. Worse still, synthesized off-policy data exacerbates this instability (Wu et al., 2025).

As depicted in Figure 1, we propose R^{3}L, **R**eflect-then-**R**etry **R**einforcement **L**earning with Language-Guided Exploration, Pivotal Credit, and Positive Amplification. It synthesizes successful trajectories through language-guided reflection, refines credit assignment through contrastive structure, and stabilizes training through positive advantage amplification. To overcome the low success rates of stochastic sampling, R^{3}L employs a reflect-then-retry mechanism that leverages language feedback to diagnose errors and identify failure points in unsuccessful trajectories, then restarts generation from these pivots with corrective guidance to synthesize successful trajectories at reduced rollout cost. Crucially, the training data removes guidance descriptions so that the model can internalize corrections in the inference stage. To refine credit assignment, we introduce Pivotal Credit Assignment that exploits the contrastive structure between base and retry trajectories. Since they share the same prefix up to the pivot, we exclude this prefix from gradient updates and focus optimization on the diverging suffix where clear signals exist. To ensure stable learning from synthesized off-policy data, we propose Positive Amplification that scales the advantages of successful trajectories. This ensures that positive signals dominate optimization even when failures outnumber successes, preventing entropy collapse. Reflection and retry skills are maintained through auxiliary meta-tasks trained on verified corrections. Our contributions are summarized as follows:

- We propose a language-guided exploration strategy that synthesizes successful trajectories by diagnosing errors and restarting generation from identified failure points with corrective guidance, improving exploration efficiency while reducing rollout cost.
- We present Pivotal Credit Assignment and Positive Amplification to stabilize training by focusing gradient updates on diverging suffixes with clear contrastive signals and ensuring positive signals dominate optimization.
- Extensive experiments demonstrate 5% to 52% relative improvements over baselines on agentic tasks, including ALFWorld, WebShop, and ScienceWorld, and mathematical reasoning benchmarks.

2 Related Work

Reinforcement learning trains LLMs on self-generated trajectories to enhance reasoning and agentic capabilities. Typically, methods like GRPO (Shao et al., 2024) use group-relative normalization to estimate advantages without learned critics, enabling recent advances in mathematical reasoning (Guo et al., 2025). However, reliance on stochastic sampling and sparse rewards introduces challenges in exploration efficiency and training stability.

2.1 Exploration Efficiency

Stochastic sampling struggles on difficult tasks where successful trajectories are rare. Sampling-based approaches like DAPO (Yu et al., 2025) and RAFT (Dong et al., 2023) compensate through oversampling and filtering, ensuring gradient validity at significant computational cost. Correction-based approaches instead leverage feedback to synthesize improved trajectories directly. HINT (Wang et al., 2025) and Agent-RLVR (Da et al., 2025) employ heuristic guidance and external critics, Goedel-Prover-V2 (Lin et al., 2025) uses scaffolded synthesis, and Reflect-Retry-Reward (Bensal et al., 2025)

rewards self-reflection tokens that lead to successful retries. While these methods improve trajectory quality, they introduce distributional shifts that can destabilize training if not properly managed (Zheng et al., 2025b). R^{3}L combines language-guided synthesis with positive amplification to stabilize learning from off-policy corrections.

2.2 Training Stability

Learning from sampled trajectories introduces gradient variance and credit assignment challenges. To reduce variance, GSPO (Zheng et al., 2025a) replaces token-level importance weights with sequence-level ratios, while BAPO (Xi et al., 2025) introduces adaptive clipping to mitigate negative-sample dominance. For credit assignment, trajectory-level rewards penalize valid prefixes when later errors occur. Critique-GRPO (Zhang et al., 2025b) uses natural language critiques to guide refinements and applies weighted advantages to the best refinement in each group. Process Reward Models (Wang et al., 2024) offer step-level supervision but require expensive annotation and produce unstable signals (Xiong et al., 2024). Alternative approaches like GiGPO (Feng et al., 2025) and VinePPO (Kazemnejad et al., 2024) estimate step-level credit through anchor states or Monte Carlo rollouts. R^{3}L takes a distinct way, exploiting the contrastive structure of retry data for precise credit assignment without external annotation, and amplifying positive signals to stabilize off-policy learning.

Similarly, Reflect-Retry-Reward (Bensal et al., 2025) and Critique-GRPO (Zhang et al., 2025b) leverage language feedback for exploration. R^{3}L differs by identifying specific failure points to reduce rollout cost, using model-driven judgments of suboptimality rather than binary verification, learning from all trajectories rather than only the best refinement or reflection tokens, and applying a simple amplification factor uniformly across the exploration group. Context distillation further removes the guidance dependency, enabling corrections to transfer directly to inference.

3 Preliminaries

3.1 Problem Formulation

We formulate the agent's interaction as a multi-turn decision process. A trajectory τ consists of K turns, where each turn k comprises an environment observation x_k and the agent's response y_k:

$$\tau = (x_1, y_1, x_2, y_2, \ldots, x_K, y_K). \tag{1}$$

Given a reward function $R(\cdot)$, the reinforcement learning objective is to maximize the expected reward:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]. \tag{2}$$

Each response y_k consists of T_k tokens. Let $h_k = (x_1, y_1, \ldots, x_{k-1}, y_{k-1}, x_k)$ denote the history up to turn k. The policy gradient is derived as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{k=1}^{K}\sum_{t=1}^{T_k}\nabla_\theta \log \pi_\theta(y_k^t|h_k, y_k^{<t})\cdot A_k^t\right], \tag{3}$$

where $\pi_\theta(y_k^t|h_k, y_k^{<t})$ represents the probability of token y_k^t given the history, and A_k^t denotes the advantage function.

3.2 Group Relative Policy Optimization

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) estimates advantages via group-wise reward normalization, eliminating the need for a learned critic. It maintains a trainable policy π_θ, a behavior policy $\pi_{\theta_{old}}$ for sampling, and a frozen reference policy π_{ref} for KL regularization.

For each query, GRPO samples a group of N trajectories $\mathcal{G} = \{\tau_1, \ldots, \tau_N\}$ using $\pi_{\theta_{old}}$. The advantage for a specific trajectory $\tau_i \in \mathcal{G}$ is computed by normalizing its reward against the group statistics:

$$A(\tau_i) = \frac{R(\tau_i) - \bar{R}}{\sigma_R}, \tag{4}$$

where $\bar{R}$ and σ_R denote the mean and standard deviation of rewards within group $\mathcal{G}$. To balance computational efficiency with stability, $\pi_{\theta_{old}}$ is synchronized with π_θ every S steps. To manage the resulting off-policy divergence, GRPO employs importance sampling with clipping as follows:

$$\mathcal{L}_{i,k,t} = \min\Big(r_{i,k,t}\hat{A}_{i,k,t}, \text{clip}(r_{i,k,t}, 1-\epsilon, 1+\epsilon)\hat{A}_{i,k,t}\Big), \tag{5}$$

where $r_{i,k,t} = \frac{\pi_\theta(y_k^t|h_k, y_k^{<t})}{\pi_{\theta_{old}}(y_k^t|h_k, y_k^{<t})}$ is the importance sampling ratio for trajectory τ_i. The full GRPO objective incorporates a KL penalty to constrain policy updates:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{\mathcal{G}\sim\pi_{\theta_{old}}}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\tau_i|}\sum_{k,t}\mathcal{L}_{i,k,t}\right] - \beta D_{KL}(\pi_\theta||\pi_{ref}). \tag{6}$$

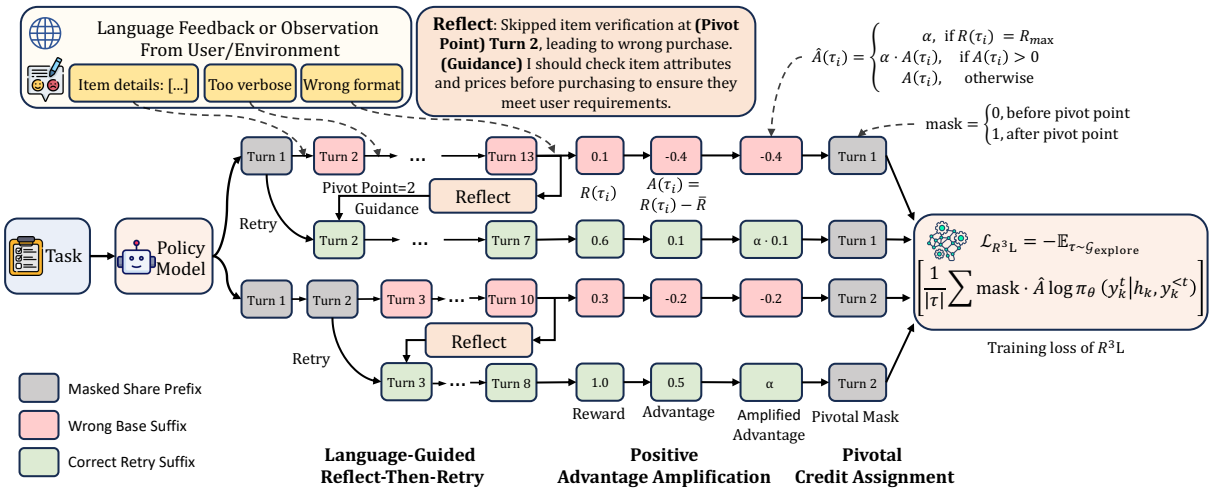


Figure 2: Overview of the R^{3}L framework. The method utilizes Language-Guided Reflect-Then-Retry to synthesize high-reward trajectories via verbal feedback. To preserve valid steps, Pivotal Credit Assignment masks shared prefixes to isolate critical decision points, while Positive Amplification scales the advantages of successful trajectories to stabilize off-policy training.

4 Methodology

As illustrated in Figure 2, R^{3}L alternates between trajectory synthesis and policy optimization. In the synthesis phase, a reflect-then-retry mechanism diagnoses failures using language feedback and restarts generation from identified pivot points, transforming failed attempts into successful ones. In the optimization phase, the policy learns from both base and retry trajectories. Pivotal Credit Assignment focuses gradient updates on the diverging suffix where contrastive signals exist, while Positive Amplification upweights successful trajectories to ensure positive signals guide optimization.

4.1 Language-Guided Reflect-Then-Retry

The core challenge in RL is discovering successful trajectories to learn from. Stochastic sampling often fails on difficult tasks, and compensating through repeated rollouts is costly. Instead, we leverage language feedback to actively synthesize high-quality trajectories. We construct four types of training data: base trajectories from stochastic sampling, distillation trajectories from guided correction, and two meta-tasks for maintaining reflection and retry skills. The detailed protocols are shown in Appendix D.

Base Trajectories. To present a fair comparison with standard RL methods, we allocate half the sampling budget to standard exploration and the other half for the retried samples, so the total sample size keeps the same. Given a query x, $N/2$ base trajectories are sampled from the behavior policy $\pi_{\theta_{old}}$. These establish baseline performance and provide raw material for reflection:

$$\mathcal{D}_{base} = \{\tau_i\}_{i=1}^{N/2}, \quad \tau_i \sim \pi_{\theta_{old}}(\cdot|x). \tag{7}$$

Distillation Trajectories. For each base sample, the model reflects on the trajectory to produce a structured diagnosis, including outcome classification (success, success but inefficient, or failure), root cause analysis, improvement suggestions, and the pivot turn k_{pivot} where the issue first manifested. We refer to the turns preceding the pivot as the *prefix* ($\tau_{<k_{pivot}}$) and the turns from the pivot onward as the *suffix* ($\tau_{\geq k_{pivot}}$); both credit assignment and trajectory stitching operate at this turn level. For trajectories not classified as fully successful, this diagnosis is embedded into a guidance prompt, and generation restarts from k_{pivot} conditioned on this guidance to obtain a corrected suffix. Crucially, we construct training inputs by pairing the original prefix with the corrected suffix, deliberately omitting the guidance. This forces the model to internalize corrections rather than rely on explicit prompts, ensuring they transfer directly to inference:

$$\mathcal{D}_{distill} = \{(\tau_{i,<k_{pivot}}, \tau'_{i,\geq k_{pivot}})\}_{i=1}^{N/2}. \tag{8}$$

Learning to Reflect and Retry. Reflection and retry are learnable skills that require explicit supervision to maintain throughout training, so we devise two auxiliary tasks. We select instances where the retry trajectory achieves a higher reward than the original attempt as verified successful corrections. Specifically, the reflection task trains the

model to produce structured diagnoses r given trajectories with environment feedback f, while the retry task trains the model to generate corrected suffixes given prefixes concatenated with guidance g (denoted $\oplus$):

$$\begin{aligned} \mathcal{D}_{reflect} &= \{([\tau_i, f_i], r_i)\}, \\ \mathcal{D}_{retry} &= \{(\tau_{i,<k_{pivot}} \oplus g_i, \tau'_{i,\geq k_{pivot}})\}. \end{aligned} \tag{9}$$

Training Groups. $\mathcal{D}_{base}$ and $\mathcal{D}_{distill}$ form the exploration group $\mathcal{G}_{explore}$ for RL optimization, while $\mathcal{D}_{reflect}$ and $\mathcal{D}_{retry}$ serve as auxiliary SFT objectives to sustain the exploration engine throughout training as Equation 12. Unlike Critique-GRPO which performs N explorations followed by N retries and selects only the best refinement, R^{3}L allocates the budget evenly between base and retry trajectories, learns from all samples in the group, and explicitly maintains reflection and retry skills through auxiliary tasks.

4.2 Pivotal Credit Assignment

Standard trajectory-level credit assigns the same reward signal to all tokens, penalizing all tokens when later errors occur. A failure at the final step suppresses the correct reasoning that preceded it, introducing noise into gradient estimates.

To assign credit more precisely, we leverage the contrastive structure of the exploration group. Base and distillation trajectories share an identical prefix up to the pivot turn k_{pivot} and diverge only in the suffix. Since both trajectories behave identically in the prefix region, this shared portion provides no information about which path is better. Only the diverging suffix, where one succeeds and the other does not, reveals which decision was correct.

We exploit this structure by applying a gradient mask that excludes the shared prefix from updates. For the t-th token in turn k:

$$\text{mask}_k^t = \begin{cases} 0 & \text{if } k < k_{pivot}, \\ 1 & \text{otherwise.} \end{cases} \tag{10}$$

Tokens before the pivot turn receive zero weight in the gradient, focusing optimization on the critical branching decision. Functionally, the shared prefix acts as a control variate, because policy behavior is identical in this region, including it would introduce variance without providing useful signal. Masking cancels this correlated noise and maximizes the signal-to-noise ratio, as analyzed in Appendix A.2.3.

4.3 Positive Amplification

While the reflect-then-retry mechanism synthesizes high-quality trajectories, learning from them poses stability challenges. On difficult tasks, failures often dominate the exploration group. Standard GRPO computes advantages by normalizing rewards against the group mean, so when most trajectories fail, this normalization dilutes the positive signal from the few successful ones, causing them to be overwhelmed by the mass of negative samples. The challenge is further compounded by distribution shifts, as retry trajectories are generated under guidance, creating a mismatch with the current policy that standard importance sampling struggles to correct.

As analyzed in Appendix A.2.1, this dilution leads to gradient asymmetry. Failed trajectories provide destructive gradients that suppress incorrect actions, while successful trajectories provide constructive gradients that reinforce correct paths. In difficult tasks, destructive signals are abundant but low-value, merely telling the model what not to do. Constructive signals are rare but high-value. Without intervention, the aggregate gradient of dense negative samples structurally overpowers the rare positive signals, forcing the policy to unlearn behaviors blindly and driving the distribution toward high entropy rather than convergence.

Recent methods address this imbalance through various mechanisms. DeepSeek-V3.2 (Liu et al., 2025) masks out negative-advantage sequences with high policy divergence. BAPO (Xi et al., 2025) dynamically adjusts clipping bounds through multiple hyperparameters to rebalance positive and negative contributions. Critique-GRPO (Zhang et al., 2025b) applies policy shaping only to refinements. Positive Amplification takes a simpler approach: It suffices to ensure that constructive gradients dominate within each group. A single amplification factor $\alpha > 1$ applied uniformly suffices. For trajectory τ with advantage $A(\tau)$:

$$\hat{A}(\tau) = \begin{cases} \alpha & \text{if } R(\tau) = R_{max}, \\ \alpha \cdot A(\tau) & \text{if } A(\tau) > 0, \\ A(\tau) & \text{otherwise.} \end{cases} \tag{11}$$

Trajectories achieving the maximum reward in the group receive the full amplification factor, other positive-advantage trajectories are scaled proportionally, and negative-advantage trajectories remain unchanged. By scaling up positive advantages, constructive gradients become strong enough to domi-

nate the group, channeling probability mass toward discovered solutions rather than letting it disperse. We find $\alpha = 3.0$ works well across tasks, with sensitivity analysis in Appendix B.

Combining the pivotal mask with amplified advantages, the final R^{3}L objective is:

$$\mathcal{L}_{R^3L} = -\mathbb{E}_{\tau \sim \mathcal{G}_{explore}} \left[\frac{1}{|\tau|} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \text{mask}_k^t \cdot \hat{A}(\tau) \log \pi_\theta(y_k^t | h_k, y_k^{<t}) \right] + \mathcal{L}_{SFT}(\mathcal{D}_{reflect} \cup \mathcal{D}_{retry}), \tag{12}$$

where $\mathcal{L}_{SFT}$ is the auxiliary supervised fine-tuning loss on verified successful corrections to maintain reflection and retry skills throughout training. Unlike standard GRPO, we remove both importance sampling and KL constraints. Importance sampling becomes unreliable for retry trajectories generated under guidance, as the behavior distribution differs fundamentally from the current policy. KL constraints are unnecessary because positive amplification already prevents the policy from drifting toward high entropy. This simplification reduces memory overhead and computational cost while maintaining training stability, as validated in Appendix A.

5 Experiments

5.1 Experimental Setup

Benchmarks. We evaluate R^{3}L on two categories of tasks:

- **Agentic Environments:** ALFWorld (language-only version) (Shridhar et al., 2020) for embodied decision-making, WebShop (Yao et al., 2022) for web navigation, and ScienceWorld (Wang et al., 2022) for long-horizon scientific reasoning.
- **Mathematical Reasoning:** In math tasks, we train the model on the DAPO training set and evaluate across diverse benchmarks to assess generalization. Evaluations include GSM8K (Cobbe et al., 2021), Math500 (Lightman et al., 2023), MinervaMath (Lewkowycz et al., 2022), OlympiadBench (Gao et al., 2024), AMC23 (Mathematical Association of America, 2025), and DAPO test set (Yu et al., 2025).

Baselines. We compare against rejection sampling methods like RAFT (Dong et al., 2023), group-relative policy optimization variants including GRPO (Shao et al., 2024), OPMD (Yao et al., 2025), and GSPO (Zheng et al., 2025a), as well as language-feedback guided approaches such as Critique-GRPO (Zhang et al., 2025b) and Reflect-Retry-Reward (Bensal et al., 2025), which we abbreviate as Reflect-GRPO. All baselines are reproduced within the Trinity-RFT framework (Pan et al., 2025) to ensure fair comparison.

Implementation. We conduct experiments on Qwen2.5-1.5B-Instruct, Qwen2.5-7B-Instruct, and Llama-3.2-3B-Instruct. Key hyperparameters include group size $N = 8$, amplification factor $\alpha = 3.0$, and synchronization step $S = 1$ for updating the behavior policy. We report trajectory-level **Average Reward** as the primary metric, measuring task completion rate for agentic environments and solution accuracy for mathematical reasoning. While our main results cover all three model architectures, subsequent ablation studies and analyses are mainly performed using Qwen2.5-1.5B-Instruct unless otherwise specified. More baseline reproduction and task-specific implementation details are provided in Appendix G.

5.2 Main Results

Table 1 presents results across agentic environments and mathematical reasoning benchmarks. Rejection sampling methods like RAFT reach 0.914 on ALFWorld and 0.884 on GSM8K with the 7B model, but degrade to 0.826 and 0.434 with the 1.5B model, where successful samples are scarce. Group-relative methods provide solid baselines but rely on stochastic sampling. GRPO reaches 0.720 on ALFWorld and 0.474 on GSM8K with the 1.5B model, while GSPO improves to 0.857 on ALFWorld through variance reduction yet lacks active trajectory synthesis. Language-feedback guided methods improve exploration. Reflect-GRPO reaches 0.878 on ALFWorld and 0.672 on GSM8K with the 1.5B model, but cannot stabilize learning from the shifted distribution. Critique-GRPO further improves to 0.914 on ALFWorld and 0.798 on GSM8K by selecting the best refinement, though discarding other refinement trajectories limits signal diversity.

R^{3}L achieves the best or second-best performance on all 27 settings. On agentic environments, R^{3}L ranks first on 8 out of 9 settings, reaching 0.928 on ALFWorld, 0.663 on WebShop, and 0.385 on ScienceWorld with the 1.5B model. The gains are most pronounced on WebShop and ScienceWorld, surpassing the strongest baselines by 8.0% and 5.2% respectively, where Pivotal Credit Assignment protects valid prefixes from late-error penal-

Model	Method	Agentic Environments			Mathematical Reasoning					
		ALFWorld	WebShop	ScienceWorld	GSM8K	Math500	MinervaMath	Olympiad	AMC23	DAPO
Qwen2.5-1.5B-Ins	RAFT	0.826	0.450	0.110	0.434	0.204	0.051	0.053	0.125	0.086
	OPMD	0.835	0.561	0.356	0.463	0.292	0.062	0.123	0.125	0.070
	GRPO	0.720	0.614	0.366	0.474	0.368	0.099	0.114	0.250	0.123
	GSPO	0.857	0.566	0.200	0.518	0.196	0.077	0.087	0.275	0.142
	Reflect-GRPO	0.878	0.559	0.347	0.672	0.376	0.102	0.130	0.300	0.136
	Critique-GRPO	0.914	0.613	0.314	**0.798**	0.404	0.110	0.124	0.275	0.133
	R^{3}L (Ours)	**0.928**	**0.663**	**0.385**	0.721	**0.424**	**0.125**	**0.151**	**0.325**	**0.156**
Llama3.2-3B-Ins	RAFT	0.064	0.434	0.074	0.620	0.336	0.095	0.096	0.275	0.156
	OPMD	0.385	0.492	0.117	0.548	0.310	0.084	0.094	0.125	0.186
	GRPO	**0.521**	0.520	0.076	0.664	0.398	0.128	**0.121**	0.325	0.213
	GSPO	0.285	0.497	0.082	0.574	0.308	0.110	0.106	0.225	0.163
	Reflect-GRPO	0.321	0.536	0.072	0.669	0.392	0.128	0.112	0.325	0.193
	Critique-GRPO	0.321	0.549	0.071	0.662	**0.426**	0.137	0.116	**0.375**	0.208
	R^{3}L (Ours)	0.495	**0.569**	**0.123**	**0.688**	0.408	**0.147**	0.118	0.325	**0.216**
Qwen2.5-7B-Ins	RAFT	0.914	0.682	0.201	0.884	0.592	0.257	0.210	0.575	0.383
	OPMD	0.914	0.684	0.359	0.836	0.352	0.252	0.133	0.225	0.270
	GRPO	0.933	0.709	0.378	0.846	0.572	0.239	0.277	0.675	0.393
	GSPO	0.895	0.720	0.363	0.866	0.548	0.209	0.284	0.675	0.413
	Reflect-GRPO	0.942	0.723	0.356	0.765	0.532	0.194	0.250	0.550	0.396
	Critique-GRPO	0.921	0.714	0.388	0.678	0.522	0.152	0.170	0.300	0.390
	R^{3}L (Ours)	**0.948**	**0.757**	**0.403**	**0.897**	**0.658**	**0.275**	**0.301**	**0.700**	**0.436**

Table 1: Main results across agentic environments and mathematical reasoning benchmarks. We report Average Reward for all tasks. **Bold** indicates the best performance and underline indicates the second best.

Method	Agentic Environments			Mathematical Reasoning					
	ALFWorld	WebShop	ScienceWorld	GSM8K	Math500	MinervaMath	Olympiad	AMC23	DAPO
R^{3}L (Full)	**0.928**	**0.663**	**0.385**	**0.721**	**0.424**	**0.125**	**0.151**	**0.325**	**0.156**
w/o Positive	0.881	0.646	0.373	0.685	0.391	0.112	0.143	0.300	0.144
w/o Credit	0.914	0.649	0.378	0.706	0.410	0.118	0.150	0.325	0.153
w/o Reflect	0.894	0.628	0.371	0.562	0.389	0.105	0.133	0.275	0.141
GRPO (Baseline)	0.720	0.614	0.366	0.474	0.368	0.099	0.114	0.250	0.123

Table 2: Ablation study on Qwen2.5-1.5B-Instruct. Each row removes the component from the R^{3}L framework. Removing Reflect-then-Retry inherently disables Pivotal Credit Assignment as the Credit module depends on pivot points identified by reflection.

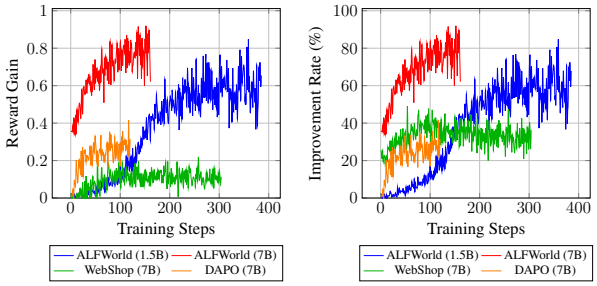
ties. On mathematical reasoning, R^{3}L achieves 0.897 on GSM8K, 0.658 on Math500, and 0.301 on OlympiadBench with the 7B model. The only exception to achieving the first best is GSM8K on 1.5B, where Critique-GRPO outperforms R^{3}L with 0.798 versus 0.721. The reason for the exception is that GSM8K is simple enough that stochastic sampling already produces sufficient successful trajectories. Nevertheless, our R^{3}L still achieves 52% gains over GRPO in GSM8K. While other harder benchmarks require more deliberate mechanisms for high-quality trajectory synthesis, assigning fine-grained credit, and learning stably from sparse positive signals.

Llama-3.2-3B exhibits more unstable and counter-intuitive behavior during RL compared to Qwen models, a phenomenon also observed in recent study (Zhang et al., 2025a). For instance, the Qwen-1.5B model achieves a RAFT score of 0.826 on ALFWorld, while Llama collapses to 0.064. This aligns with findings that Qwen is more amenable to reinforcement learning, likely due to its reasoning-oriented mid-training. Notably, despite these optimization challenges, R^{3}L demonstrates consistent improvements.

5.3 Ablation Study

To validate the necessity of each component, we systematically remove Positive Amplification, Pivotal Credit Assignment, and Reflect-then-Retry. Since the Credit module depends on pivot points identified by reflection, removing the Reflect module inherently disables the Credit mechanism.

Table 2 presents the ablation results. Removing Reflect-then-Retry causes the most severe degradation, with GSM8K dropping from 0.721 to 0.562 and WebShop from 0.663 to 0.628. This confirms that stochastic sampling alone cannot penetrate sparse-reward landscapes, and active trajectory synthesis is essential for discovering successful solutions. Excluding Positive Amplification degrades performance across all benchmarks, with ALFWorld dropping from 0.928 to 0.881 and GSM8K from 0.721 to 0.685. In failure-dominated regimes,



(a) Avg. Reward Gain (b) Retry Improvement Rate

Figure 3: Evolution of exploration metrics across environments by the Reflect-then-Retry mechanism. **(a)** The average Reward Gain of retry trajectory relative to the base trajectory. **(b)** The percentage of retry trajectories that successfully improved upon the base attempt.

Model	ALFWorld	WebShop	ScienceWorld	DAPO
Qwen2.5-1.5B-Ins	64.7%	23.7%	13.5%	16.4%
Llama3.2-3B-Ins	12.8%	31.6%	9.2%	10.3%
Qwen2.5-7B-Ins	73.9%	36.5%	19.6%	27.1%

Table 3: Average reward improvement rate achieved by the Reflect-then-Retry mechanism.

the abundance of negative samples overwhelms rare positive signals, and without amplification, gradients focus on suppressing errors rather than reinforcing correct paths. Omitting Pivotal Credit Assignment has the smallest but consistent impact, with ALFWorld dropping from 0.928 to 0.914 and GSM8K from 0.721 to 0.706. The effect is most visible on long-horizon tasks where trajectory-level rewards penalize valid prefixes when errors occur.

5.4 Efficacy of Language-Guided Reflection

A key question is whether language-guided reflection can reliably improve upon base trajectories throughout training. To analyze this, we track two metrics: the Retry Improvement Rate, defined as the percentage of retry trajectories that achieve higher reward than their corresponding base attempts ($R_{retry} > R_{base}$), and the Reward Gain, defined as the relative improvement $(\bar{R}_{retry} - \bar{R}_{base})/\bar{R}_{base}$ in average reward. Figure 3 illustrates the temporal evolution of these metrics across environments and model scales. We observe two distinct patterns in the training dynamics.

Cold Start and Warm-up. Most configurations start with Reward Gain near zero, including ALFWorld with the 1.5B model, WebShop with the 7B model, and DAPO with the 7B model. During this phase the model learns to generate structured diagnoses and apply corrections effectively. ALFWorld with the 7B model is the exception, starting with Reward Gain around 0.4 because its stronger base capabilities enable effective reflection without additional adaptation.

Sustained Improvement Gap. After warm-up, all configurations show sustained improvement that persists throughout training. We achieve the highest Reward Gain on ALFWorld, reaching 0.6 with the 1.5B model and 0.8 with the 7B model. DAPO follows with 0.3, while WebShop reaches 0.15. The variation across tasks reflects how amenable different error types are to language-guided correction. ALFWorld failures typically stem from discrete action errors that are easy to diagnose and fix through reflection. Mathematical reasoning in DAPO involves localized mistakes in derivation steps that can be identified and corrected. WebShop requires navigating complex search and matching processes where improvements are more incremental. Table 3 summarizes the average Retry Improvement Rate across the full training process. Qwen2.5-7B-Instruct achieves the highest rates, reaching 73.9% on ALFWorld and 36.5% on WebShop, while Qwen2.5-1.5B-Instruct follows with 64.7% and 23.7% respectively.

Two key factors drive this improvement through the reflection-then-retry. First, as the model masters environment dynamics, both base performance and reflection quality improve in tandem. Second, the auxiliary SFT tasks explicitly maintain reflection and retry capabilities, preventing the skill degradation that occurs without direct supervision. Methods like Critique-GRPO and Reflect-GRPO lack such auxiliary tasks, causing reflection quality to deteriorate as the policy distribution shifts. This sustained improvement rate ensures that Positive Amplification consistently receives high-quality trajectories to reinforce.

6 Conclusion

Current reinforcement learning approaches for LLMs are severely constrained by inefficient exploration, coarse credit assignment, and training instability in failure-dominated regimes. To overcome these structural bottlenecks, we introduced R^{3}L, which transforms sparse-reward environments into rich learning opportunities. By leveraging Reflect-then-Retry to actively synthesize high-quality trajectories, employing Pivotal Credit Assignment to protect valid reasoning prefixes, and utilizing Positive Amplification to ensure that successful signals dominate gradient updates, R^{3}L achieves robust learning where standard methods collapse or converge prematurely.

Limitations

Despite the merits, we acknowledge that R^3L has several limitations as follows. First, the reflection step requires an additional inference pass to diagnose errors. However, the pivot mechanism compensates by restarting generation from identified failure points rather than from scratch, reducing retry cost. Overall, R^3L achieves faster training than methods that rely on repeated full rollouts to discover successful trajectories.

Second, we observe a cold-start phenomenon in smaller models. Unlike the 7B model that can immediately leverage reflection prompts, the 1.5B model initially struggles to generate actionable self-corrections. This necessitates a longer warm-up period to bootstrap reflection capability, potentially complicating the training pipeline for low-resource settings.

Finally, while our framework is theoretically applicable to any domain where a preference signal exists, our experimental scope is limited to tasks with verifiable ground truth. We have not validated R^3L in open-ended domains with subjective criteria such as creative writing, where the reliability of automated retry validation remains an open question for future research.

References

Shelly Bensal, Umar Jamil, Christopher Bryant, Melisa Russak, Kiran Kamble, Dmytro Mozolevskyi, Muayad Ali, and Waseem AlShikh. 2025. Reflect, retry, reward: Self-improving llms via reinforcement learning. *arXiv preprint arXiv:2505.24726*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yue Cui, Liuyi Yao, Shuchang Tao, Weijie Shi, Yaliang Li, Bolin Ding, and Xiaofang Zhou. 2025. Enhancing tool learning in large language models with hierarchical error checklists. *arXiv preprint arXiv:2506.00042*.

Jeff Da, Clinton Wang, Xiang Deng, Yuntao Ma, Nikhil Barhate, and Sean Hendryx. 2025. Agent-rlvr: Training software engineering agents via guidance and environment rewards. *arXiv preprint arXiv:2506.11425*.

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. *Transactions on Machine Learning Research*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, and 1 others. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.

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

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

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

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, and 1 others. 2025. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint arXiv:2508.03613*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Mathematical Association of America. 2025. American Mathematics Competitions (AMC). https://maa.org/student-programs/amc/.

Gongrui Nan, Siye Chen, Jing Huang, Mengyu Lu, Dexun Wang, Chunmei Xie, Weiqi Xiong, Xianzhou Zeng, Qixuan Zhou, Yadong Li, and 1 others. 2025. Ngrpo: Negative-enhanced group relative policy optimization. *arXiv preprint arXiv:2509.18851*.

Xuchen Pan, Yanxi Chen, Yushuo Chen, Yuchang Sun, Daoyuan Chen, Wenhao Zhang, Yuexiang Xie, Yilun Huang, Yilei Zhang, Dawei Gao, and 1 others. 2025. Trinity-rft: A general-purpose and unified framework for reinforcement fine-tuning of large language models. *arXiv preprint arXiv:2505.17826*.

Aske Plaat, Max van Duijn, Niki van Stein, Mike Preuss, Peter van der Putten, and Kees Joost Batenburg. 2025. Agentic large language models, a survey. *arXiv preprint arXiv:2503.23037*.

Jérémy Scheurer, Jon Ander Campos, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2022. Training language models with language feedback. *arXiv preprint arXiv:2204.14146*.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2023. Training language models with language feedback at scale. *arXiv preprint arXiv:2303.16755*.

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

Weijie Shi, Yue Cui, Yaguang Wu, Jingzhi Fang, Shibo Zhang, Mengze Li, Sirui Han, Jia Zhu, Jiajie Xu, and Xiaofang Zhou. 2025. Semantic-guided diverse decoding for large language model. *arXiv preprint arXiv:2506.23601*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *ACL*, pages 9426–9439.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*.

Xinyi Wang, Jinyi Han, Zishang Jiang, Tingyun Li, Jiaqing Liang, Sihang Jiang, Zhaoqian Dai, Shuguang Ma, Fei Yu, and Yanghua Xiao. 2025. Hint: Helping ineffective rollouts navigate towards effectiveness. *arXiv preprint arXiv:2510.09388*.

Kun Wu, Ning Liu, Zhen Zhao, Di Qiu, Jinming Li, Zhengping Che, Zhiyuan Xu, and Jian Tang. 2025. Learning from imperfect demonstrations with self-supervision for robotic manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16899–16906. IEEE.

Zhiheng Xi, Xin Guo, Yang Nan, Enyu Zhou, Junrui Shen, Wenxiang Chen, Jiaqi Liu, Jixuan Huang, Zhihao Zhang, Honglin Guo, and 1 others. 2025. Bapo: Stabilizing off-policy reinforcement learning for llms via balanced policy optimization with adaptive clipping. *arXiv preprint arXiv:2510.18927*.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024. Watch every step! llm agent learning via iterative step-level process refinement. In *EMNLP*, pages 1556–1572.

Chaorui Yao, Yanxi Chen, Yuchang Sun, Yushuo Chen, Wenhao Zhang, Xuchen Pan, Yaliang Li, and Bolin Ding. 2025. Group-relative reinforce is secretly an off-policy algorithm: Demystifying some myths about grpo and its friends. *arXiv preprint arXiv:2509.24203*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

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

Charlie Zhang, Graham Neubig, and Xiang Yue. 2025a. On the interplay of pre-training, mid-training, and rl on reasoning language models. *arXiv preprint arXiv:2512.07783*.

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

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

Haizhong Zheng, Jiawei Zhao, and Bedi Chen. 2025b. Prosperity before collapse: How far can off-policy rl reach with stale data on llms? *arXiv preprint arXiv:2510.01161*.

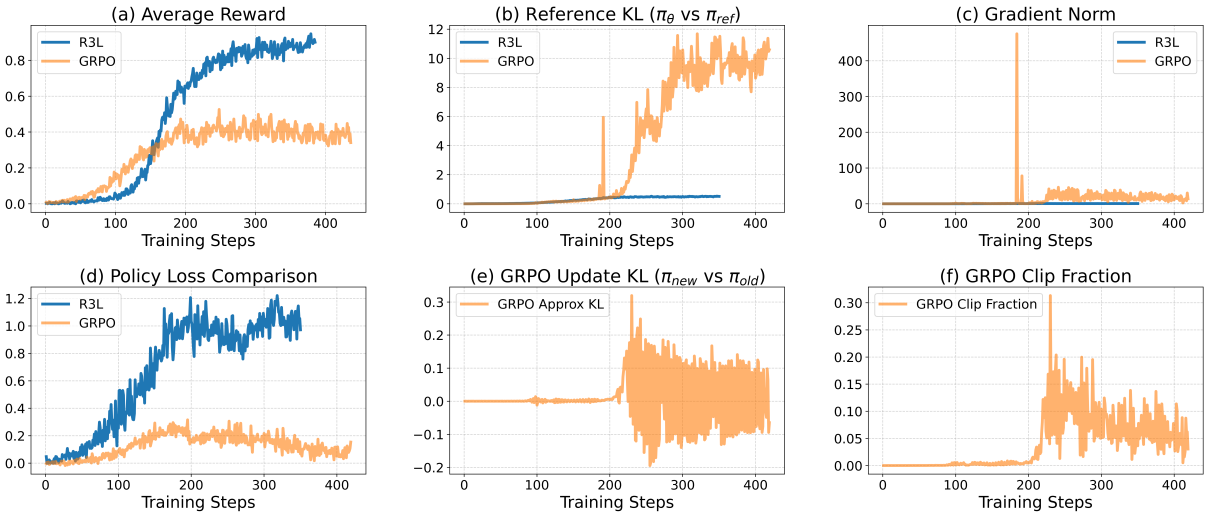


Figure 4: Training Dynamics and Stability Analysis on ALFWorld. **(a)** Average reward comparison. **(b)** Reference KL divergence between current policy and reference policy. **(c)** Gradient norm during training. **(d)** Policy loss comparison. **(e)** GRPO update KL between new and old policy. **(f)** GRPO clip fraction.

A Stability Analysis and Theoretical Foundations

A.1 Training Dynamics and Convergence

Reinforcement learning for LLMs is notoriously sensitive to initialization and hyperparameters. To understand the underlying causes of instability, we analyze the training dynamics on ALFWorld as illustrated in Figure 4.

Performance and Convergence. As shown in Figure 4(a), standard GRPO exhibits earlier initial gains, showing reward improvements around step 100. However, it suffers from premature stagnation, plateauing at reward approximately 0.4 with visible variance. In contrast, R^{3}L undergoes a warm-up phase due to initial adaptation to the reflection mechanism. Following step 120, R^{3}L demonstrates a rapid phase transition with a significantly steeper learning curve, quickly surpassing the baseline and achieving superior asymptotic performance of approximately 0.9 without the instability observed in GRPO.

Policy Drift and Gradient Anomalies. The instability of GRPO is further evidenced by policy drift and gradient anomalies. Figure 4(b) shows that the Reference KL divergence for GRPO explodes to values exceeding 10.0 after step 200, indicating severe policy drift where the model loses its semantic grounding. This drift forces aggressive clipping of updates, with Clip Fraction surging to 30% as shown in Figure 4(f). Furthermore, Figure 4(c) captures a massive gradient spike with norm exceeding 400 around step 190, coinciding with the onset of collapse. R^{3}L eliminates these artifacts entirely, maintaining smooth gradient norms and consistently low KL divergence throughout training. We attribute this stability to the synergy between Pivotal Credit Assignment and Positive Amplification, where the former reduces gradient variance and the latter rectifies gradient direction. Figure 4(d) confirms that R^{3}L maintains positive policy loss throughout training, indicating sustained constructive learning signals.

A.2 Theoretical Foundations

We now provide formal theoretical justifications for the empirical stability observed above.

A.2.1 Gradient Decomposition and Entropy Collapse

Standard GRPO employs variance-based normalization $A_i = (R_i - \bar{R})/\sigma$. In failure-dominated regimes, the instability arises not merely from the quantity of negative samples but from the distributional asymmetry of the policy's probability mass combined with gradient conflict.

Definition 1 (Gradient Decomposition)**.** The expected policy gradient decomposes into constructive and destructive components:

$$\mathbb{E}[\nabla J(\theta)] = \underbrace{p \cdot \bar{A}^+ \cdot \nabla^+}_{\mathcal{C}\text{:Constructive}} + \underbrace{(1-p) \cdot \bar{A}^- \cdot \nabla^-}_{\mathcal{D}\text{:Destructive}}, \tag{13}$$

where p is the fraction of positive-advantage trajectories, $\bar{A}^+$ and $\bar{A}^-$ are the mean advantages, and ∇^+ and ∇^- denote the average gradient directions.

Consider the critical exploration phase where the model has not yet solidified the correct reasoning path but is confident in erroneous heuristics. In this regime, the sampled group $\mathcal{G}$ is structurally dominated by negative samples with $A_{neg} < 0$ that simultaneously hold high probability mass where $\pi_\theta(y_{neg}) \gg 0$. Conversely, the correct solution y_{pos} lies in the low-probability tail with $\pi_\theta(y_{pos}) \approx 0$ and is rarely sampled.

Proposition 2 (Entropy Collapse)**.** *When $|\mathcal{D}| > |\mathcal{C}|$, the policy gradient primarily decreases the probability of high-probability erroneous tokens. The gradient update satisfies:*

$$\Delta\pi(y_{neg})<0 \Longrightarrow \sum_{v\neq y_{neg}} \Delta\pi(v)=-\Delta\pi(y_{neg})>0. \tag{14}$$

Without a strong positive attractor, this redistributed mass disperses across the vocabulary, monotonically increasing entropy $H(\pi_\theta)$.

Proof. The probability simplex constraint requires $\sum_v \pi(v) = 1$. When the gradient decreases $\pi(y_{neg})$ by $\delta > 0$, the freed probability mass must be redistributed. Let z_v denote logits. The gradient update on $z_{y_{neg}}$ is strongly negative proportional to $|\mathcal{D}|$, while updates on other z_v are weakly positive. Without amplification, the positive updates are insufficient to create a sharp mode, leading to a flatter distribution and increased entropy. The model effectively unlearns its structured behavior, manifesting as the reward stagnation in Figure 4(a) and policy drift in Figure 4(b). □

A.2.2 Gradient Dominance Condition

Positive Amplification resolves entropy collapse by enforcing a gradient dominance condition that ensures constructive signals outweigh destructive ones.

Theorem 3 (Gradient Dominance Condition)**.** *For the amplified constructive gradient to dominate the destructive gradient, the amplification factor α must satisfy:*

$$\alpha > \alpha_{\min} = \frac{(1-p)|\bar{A}^-|}{p \cdot \bar{A}^+}. \tag{15}$$

Proof. With amplification factor α applied to positive-advantage trajectories, the expected gradient becomes:

$$\mathbb{E}[\nabla J_\alpha(\theta)] = \alpha \cdot p \cdot \bar{A}^+ \cdot \nabla^+ + (1-p) \cdot \bar{A}^- \cdot \nabla^-. \tag{16}$$

For constructive signals to dominate, we require the magnitude of the positive term to exceed the negative term. Assuming comparable gradient norms $\|\nabla^+\| \approx \|\nabla^-\|$, this reduces to $\alpha \cdot p \cdot \bar{A}^+ > (1-p) \cdot |\bar{A}^-|$, yielding the stated bound. □

Corollary 4 (Robustness of $\alpha = 3.0$)**.** *The choice $\alpha = 3.0$ satisfies the gradient dominance condition across typical reinforcement learning scenarios. In practice, the success fraction p ranges from 0.25 on difficult tasks with weaker models to 0.45 on easier tasks with stronger models. The advantage ratio $|\bar{A}^-|/\bar{A}^+$ stays between 1.0 and 2.0 due to group normalization. Under these conditions, α_{min} ranges from approximately 1.2 to 3.0. Our choice of $\alpha = 3.0$ thus covers the most practical spectrum of typical scenarios, from easy tasks where lower amplification would suffice to difficult tasks where $\alpha = 3.0$ is necessary. Setting α higher risks overfitting to specific retry trajectories, as validated in Appendix B.*

By satisfying the gradient dominance condition, Positive Amplification constructs a synthetic attractor that channels the probability mass released from suppressing y_{neg} toward y_{retry} rather than scattering it across the vocabulary. This ensures that the primary learning signal is to reinforce correct behavior rather than merely suppress errors, aligning with theoretical insights from RED (Yao et al., 2025) that weighting high-reward samples is essential for consistent off-policy convergence.

A.2.3 Variance Reduction via Pivotal Credit Assignment

While Positive Amplification addresses gradient direction, Pivotal Credit Assignment reduces gradient variance by exploiting the contrastive structure of base-retry pairs.

Theorem 5 (Variance Reduction)**.** *Let T be the total trajectory length and T_{pivot} the pivot position. Pivotal Credit Assignment reduces gradient variance by:*

$$\frac{\mathrm{Var}(\nabla_{\mathrm{R^3L}})}{\mathrm{Var}(\nabla_{\mathrm{GRPO}})} \leq \frac{T - T_{pivot}}{T}. \tag{17}$$

Proof. For standard GRPO, the gradient variance is the sum of per-step variances:

$$\mathrm{Var}(\nabla_{\mathrm{GRPO}}) = \sum_{t=1}^{T} \mathrm{Var}(\nabla_t) = T \cdot \sigma^2, \tag{18}$$

where σ^2 is the average per-step variance. With pivotal masking, tokens before t_{pivot} receive zero gradient weight:

$$\mathrm{Var}(\nabla_{\mathrm{R^3L}}) = \sum_{t=t_{pivot}}^{T} \mathrm{Var}(\nabla_t) = (T - T_{pivot}) \cdot \sigma^2. \tag{19}$$

Taking the ratio yields the stated bound. □

Proposition 6 (Covariance Analysis)**.** *Let $\hat{g}_{base}$ and $\hat{g}_{retry}$ be gradient estimators from base and retry trajectories. Because they share the prefix, their covariance satisfies:*

$$\mathrm{Cov}(\hat{g}_{base}, \hat{g}_{retry}) = \frac{T_{pivot}}{T} \cdot \mathrm{Var}(\hat{g}). \tag{20}$$

The variance of the difference estimator is:

$$\mathrm{Var}(\hat{g}_{retry} - \hat{g}_{base}) = 2\left(1 - \frac{T_{pivot}}{T}\right) \cdot \mathrm{Var}(\hat{g}). \tag{21}$$

This high correlation significantly reduces effective learning signal variance.

Theorem 5 reveals a natural coupling between model improvement and variance reduction. As training progresses and the model strengthens, errors occur later in trajectories, pushing pivot points rightward as observed in Figure 6. By Theorem 5, this rightward shift directly increases variance reduction. Early in training when pivot points cluster near the beginning, variance reduction is modest. As the model matures and pivot points migrate toward trajectory ends, variance reduction approaches its maximum. For instance, when the average pivot point shifts from $T_{pivot}/T = 0.2$ to $T_{pivot}/T = 0.6$, variance reduction improves from 80% to 40% of baseline variance. This self-reinforcing dynamic means that Pivotal Credit Assignment becomes increasingly effective precisely when the model needs fine-grained credit assignment most.

A.2.4 Off-Policy Stability via Context Distillation

A critical question is why R^{3}L remains stable despite learning from off-policy retry data without importance sampling or KL constraints.

Theorem 7 (Off-Policy Stability)**.** *Unlike standard off-policy RL requiring importance sampling, R^{3}L's verified distillation is stable without importance weights because learning signals come from verified successful trajectories with $R(\tau') > R(\tau_{base})$, ensuring high-quality supervision regardless of distributional shift. Positive Amplification ensures these signals dominate the gradient, making importance corrections unnecessary. Context distillation removes guidance dependency, aligning training with inference distribution.*

R^{3}L can be viewed as Filtered Behavioral Cloning where the filter is reward verification. Unlike OPMD which imitates stale, potentially noisy samples from the old policy, R^{3}L distills verified high-reward trajectories. By amplifying these golden signals via Positive Amplification, R^{3}L effectively treats off-policy learning as stable supervised distillation rather than fragile importance sampling, preventing the policy drift shown in Figure 4(b).

A.2.5 Convergence Guarantee

Theorem 8 (Local Convergence)**.** *Under the following conditions, the policy sequence $\{\pi_k\}$ generated by R^{3}L converges to a local optimum π^*: retry success rate $p_{retry} > 0$, amplification factor $\alpha > \alpha_{min}$, pivot identification accuracy exceeds random baseline, and learning rate η_k satisfies $\sum_k \eta_k = \infty$ and $\sum_k \eta_k^2 < \infty$.*

Proof Sketch. Define Lyapunov function $V(\pi) = -\mathbb{E}_{\tau\sim\pi}[R(\tau)]$. Under the stated conditions, each R^{3}L update produces gradient g such that $\langle g, \nabla V \rangle < 0$ in expectation. The learning rate condition ensures convergence via standard stochastic approximation arguments. □

Proposition 9 (Sustained Teacher-Student Gap)**.** *Define Retry Improvement Rate $RIR_k = \mathbb{P}(R(\tau_{retry}) > R(\tau_{base})|\pi_k)$. Under R^{3}L training dynamics with auxiliary SFT tasks maintaining reflection quality:*

$$\exists \epsilon > 0 : RIR_k > \epsilon, \quad \forall k. \tag{22}$$

This ensures continued supervision even as the model improves, as empirically validated in Section 5.4.

B Hyperparameter Analysis

B.1 Amplification Factor

The amplification factor α in Positive Amplification controls the relative weight of successful trajectories in the gradient update. As defined in Equation 11, trajectories achieving the maximum reward

in the group receive advantage α, other positive-advantage trajectories are scaled by α, and negative-advantage trajectories remain unchanged. We evaluate R^3L across $\alpha \in \{1.0, 2.0, 3.0, 5.0, 7.0\}$.

Advantage Amplification Factor (α). The amplification factor α in Positive Preference Optimization controls the relative weight of successful retry trajectories in the gradient update. To determine the optimal balance, we evaluate performance on GSM8K and WebShop across $\alpha \in \{1.0, 2.0, 3.0, 5.0, 7.0\}$.

Task	$\alpha = 1.0$	$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 5.0$	$\alpha = 7.0$
ALFWorld	0.921	0.914	0.928	0.921	**0.942**
WebShop	0.647	0.650	**0.663**	0.638	0.614
ScienceWorld	0.375	0.382	**0.385**	0.366	0.353
GSM8K	**0.732**	0.718	0.721	0.671	0.717
Math500	0.414	0.420	**0.424**	0.372	0.404
Olympiad	0.143	**0.154**	0.151	0.145	0.132

Table 4: Impact of amplification factor α on task performance. All configurations assign advantage 1 to trajectories achieving maximum reward in the group, ensuring positive signals even at $\alpha = 1.0$.

Table 4 demonstrates the impact of α. Even at $\alpha = 1.0$, R^3L achieves strong performance because trajectories with maximum reward receive a fixed advantage of 1, guaranteeing constructive learning signals regardless of amplification strength. This baseline amplification explains why $\alpha = 1.0$ already performs well on tasks like GSM8K where it achieves 0.732.

Increasing α further amplifies the contribution of positive-advantage trajectories beyond just the best one. Agentic environments benefit from this additional amplification, with WebShop improving from 0.647 at $\alpha = 1.0$ to 0.663 at $\alpha = 3.0$ and ScienceWorld from 0.375 to 0.385. Mathematical reasoning tasks show more stability across moderate α values, with GSM8K performing best at $\alpha = 1.0$ while Math500 peaks at $\alpha = 3.0$. Pushing α beyond 5.0 causes degradation on most tasks due to overfitting on specific retry trajectories. The choice $\alpha = 3.0$ provides robust performance across diverse tasks, consistent with the theoretical analysis in Appendix A.2.2.

B.2 Synchronization Frequency

A critical challenge in iterative reinforcement learning is the distributional drift between the behavior policy $\pi_{\theta_{old}}$ and the learner policy π_θ. Frequent synchronization is computationally expensive, making robustness to stale reference policies desirable. We evaluate R^3L against GRPO, OPMD, and GRPO+Positive which adds Positive Amplification to standard GRPO.

Task	Method	$S = 1$	$S = 5$	$S = 10$	$S = 20$
ALFWorld	OPMD	0.835	$0.357^\dagger$	$0.257^\dagger$	$0.341^\dagger$
	GRPO	0.720	0.716	$0.389^\dagger$	0.742
	GRPO+Positive	0.807	0.864	0.778	0.764
	R^3L	**0.928**	**0.945**	**0.922**	**0.934**
WebShop	OPMD	0.561	0.619	$0.518^\dagger$	$0.460^\dagger$
	GRPO	0.614	0.638	0.612	0.610
	GRPO+Positive	0.648	0.633	0.632	0.622
	R^3L	**0.663**	**0.663**	**0.672**	**0.657**
GSM8K	OPMD	$0.463^\dagger$	$0.604^\dagger$	$0.655^\dagger$	$0.499^\dagger$
	GRPO	0.474	0.673	0.692	0.712
	GRPO+Positive	0.504	0.520	0.714	0.713
	R^3L	**0.721**	**0.779**	**0.756**	**0.753**
Math500	OPMD	$0.292^\dagger$	$0.354^\dagger$	$0.332^\dagger$	$0.336^\dagger$
	GRPO	0.368	0.406	0.408	0.406
	GRPO+Positive	0.377	0.409	0.411	0.406
	R^3L	**0.424**	**0.464**	**0.441**	**0.426**

Table 5: Effect of synchronization frequency S across benchmarks. S denotes the number of training steps between behavior policy updates. † denotes collapsed training where we report the peak score before collapse.

Table 5 reveals the stability characteristics of different methods under varying synchronization frequencies. Increasing S enlarges the gap between the behavior policy and the learner policy, introducing more off-policy data into training and increasing the risk of instability. OPMD is highly sensitive to this risk, collapsing on ALFWorld from 0.835 at $S = 1$ to 0.257 at $S = 10$, and fluctuating erratically on GSM8K between 0.463 and 0.655 before dropping to 0.499 at $S = 20$. Standard GRPO shows task-dependent sensitivity, improving on GSM8K as S increases but collapsing on ALFWorld at $S = 10$. GRPO+Positive demonstrates that Positive Amplification mitigates the instability risk, achieving 0.504 at $S = 1$ on GSM8K and improving to 0.714 at $S = 10$. R^3L achieves the highest absolute performance across all tasks and synchronization intervals while maintaining stability. On ALFWorld, R^3L stays above 0.920 across all S values. On GSM8K, it reaches 0.779 at $S = 5$ and maintains 0.753 at $S = 20$, consistently outperforming all baselines.

C Algorithm

Algorithm 1 summarizes the complete R^3L training procedure. The algorithm addresses a fundamental tension in reinforcement fine-tuning: discovering successful trajectories requires exploration, but learning from them requires stable opti-

Algorithm 1 R^{3}L: Reflect-then-Retry Reinforcement Learning with Language-Guided Exploration, Pivotal Credit and Positive Amplification

Input: Policy π_θ, Reward function R, Prompt dataset $\mathcal{D}$
Hyperparameters: Group size N, Amplification factor α
Initialize: $\theta \leftarrow \theta_{old}$
while not converged **do**
  for each prompt x in batch $\mathcal{B} \sim \mathcal{D}$ **do**
    $\mathcal{G} \leftarrow \emptyset$, $\mathcal{D}_{reflect} \leftarrow \emptyset$, $\mathcal{D}_{retry} \leftarrow \emptyset$ ▷ Initialize exploration group and SFT datasets
    // Phase 1: Language-Guided Exploration
    Sample $N/2$ base trajectories: $\{\tau_1, \ldots, \tau_{N/2}\} \sim \pi_{\theta_{old}}(\cdot|x)$
    for each base trajectory τ_i with environment feedback f_i **do**
      Generate reflection $r_i \sim \pi_{\theta_{old}}(\cdot|\tau_i, f_i, \text{prompt}_{reflect})$
      Parse r_i to obtain pivot turn k_{pivot} and guidance g_i
      Sample retry suffix from k_{pivot}: $\tau'_{i,\geq k_{pivot}} \sim \pi_{\theta_{old}}(\cdot|\tau_{i,<k_{pivot}}, g_i)$
      Context Distillation: Construct $\tau'_i = (\tau_{i,<k_{pivot}}, \tau'_{i,\geq k_{pivot}})$ without g_i
      // Phase 2: Pivotal Credit Assignment
      Define pivot mask: $M[k] = 0$ if $k < k_{pivot}$ else 1
      Add (τ_i, M) to $\mathcal{G}$
      Add (τ'_i, M) to $\mathcal{G}$
      // Collect SFT Data for Meta-Tasks (verified successful corrections)
      if $R(\tau'_i) > R(\tau_i)$ **then**
        Add $([\tau_i, f_i], r_i)$ to $\mathcal{D}_{reflect}$ ▷ Learn to reflect
        Add $(\tau_{i,<k_{pivot}} \oplus g_i, \tau'_{i,\geq k_{pivot}})$ to $\mathcal{D}_{retry}$ ▷ Learn to retry
      end if
    end for
    // Phase 3: Positive Amplification
    Compute rewards $\{R(\tau)|\tau \in \mathcal{G}\}$
    Compute group statistics: $\bar{R} = \text{mean}$, $\sigma_R = \text{std}$, $R_{max} = \max$
    for each $(\tau_j, M_j) \in \mathcal{G}$ **do**
      $A_j \leftarrow (R(\tau_j) - \bar{R})/\sigma_R$
      if $R(\tau_j) = R_{max}$ **then**
        $\hat{A}_j \leftarrow \alpha$
      else if $A_j > 0$ **then**
        $\hat{A}_j \leftarrow \alpha \cdot A_j$
      else
        $\hat{A}_j \leftarrow A_j$
      end if
    end for
    // Policy Update with RL and Auxiliary SFT Objectives
    $\mathcal{L}_{RL}(\theta) = -\frac{1}{|\mathcal{G}|}\sum_{(\tau_j,M_j)\in\mathcal{G}} \frac{1}{|\tau_j|}\sum_{k,t} M_j[k] \cdot \hat{A}_j \cdot \log \pi_\theta(y_k^t|h_k, y_k^{<t})$
    $\mathcal{L}_{SFT}(\theta) = -\sum_{(x,y)\in\mathcal{D}_{reflect}\cup\mathcal{D}_{retry}} \log \pi_\theta(y|x)$ ▷ Maintain reflection and retry skills
    Update θ via gradient descent on $\mathcal{L}_{RL}(\theta) + \mathcal{L}_{SFT}(\theta)$
  end for
end while

mization. R^{3}L resolves this tension through three integrated phases.

Phase 1: Language-Guided Exploration. Standard methods allocate their full sampling budget to stochastic exploration, which yields predominantly failed trajectories on difficult tasks. R^{3}L instead splits the budget evenly between base sampling and guided retry. For each base trajectory, the model generates a structured reflection that diagnoses errors and identifies the pivot turn where the issue first manifested. Generation then restarts from this pivot conditioned on the diagnostic guidance, producing a retry suffix that addresses the identified failure. The critical step is context distillation, which combines the original prefix with the corrected suffix while removing the guidance from the training input. This forces the model to internalize corrections rather than relying on explicit diagnostic prompts, ensuring that improvements transfer directly to inference where no guidance is available.

Phase 2: Pivotal Credit Assignment. The exploration phase produces base and retry trajectories that share identical prefixes up to the pivot turn. This contrastive structure enables precise credit assignment without external annotation. We construct a binary mask that zeros out all turns before the pivot, excluding the shared prefix from gradient computation. Both base and retry trajectories receive the same mask, focusing optimization ex-

clusively on the diverging suffix where one path succeeds and the other fails. This design serves two purposes. First, it protects valid reasoning in the prefix from being penalized when a later error causes trajectory failure. Second, it reduces gradient variance by eliminating the noise contribution from tokens where both trajectories behave identically.

Phase 3: Positive Amplification. The exploration group now contains both base and retry trajectories with varying rewards. On difficult tasks, failures often outnumber successes even after retry, causing standard group-relative normalization to dilute positive signals. We counteract this by reweighting advantages before computing the policy gradient. Trajectories achieving the maximum reward in the group receive the full amplification factor α, ensuring at least one strong positive signal per group. Other positive-advantage trajectories are scaled proportionally to maintain their relative ordering, while negative-advantage trajectories remain unchanged. This asymmetric treatment guarantees that constructive gradients dominate the update direction, channeling probability mass toward discovered solutions rather than dispersing it through blind error suppression.

The final policy update combines the pivot mask with amplified advantages in a single gradient step. Unlike standard GRPO, R^{3}L omits both importance sampling and KL regularization. Importance sampling becomes unreliable for retry trajectories generated under guidance, as the behavior distribution differs fundamentally from the current policy. KL constraints are unnecessary because positive amplification already prevents entropy collapse. This simplification reduces computational and memory overhead while the three-phase design maintains training stability.

D Trajectory Visualization and Context Distillation

This section details the four trajectory types generated during R^{3}L training and analyzes how context distillation enables learning from guided exploration without creating inference-time dependencies. Figure 5 visualizes these four categories.

D.1 Four Trajectory Types

Type 1: Base Trajectories. Standard exploration samples trajectories directly from the behavior policy $\pi_{\theta_{old}}(\cdot|x)$. These trajectories form $\mathcal{D}_{base}$ and enter the exploration group $\mathcal{G}_{explore}$ for RL optimization.

Type 2: Reflection Trajectories. Given a base trajectory paired with environment feedback, the model generates a structured diagnosis including outcome classification, root cause analysis, improvement suggestions, and the pivot turn k_{pivot} where the issue first manifested. This reflection transforms implicit failure signals into explicit actionable guidance.

Type 3: Retry Trajectories. Generation restarts from k_{pivot} conditioned on the diagnostic guidance g, producing a corrected suffix $\tau'_{\geq k_{pivot}}$. Here, we refer to the turns preceding the pivot as the *prefix* ($\tau_{<k_{pivot}}$) and the turns from the pivot onward as the *suffix* ($\tau_{\geq k_{pivot}}$); both credit assignment and trajectory stitching operate at this turn level. Conditioning on explicit error analysis significantly increases the likelihood of generating improved solutions compared to unguided sampling.

Type 4: Distillation Trajectories. We construct distillation trajectories by pairing the original prefix $\tau_{<k_{pivot}}$ with the retry suffix $\tau'_{\geq k_{pivot}}$, explicitly removing the guidance g from the context. These trajectories form $\mathcal{D}_{distill}$ and enter the exploration group alongside base trajectories for RL optimization. This construction is essential because directly training on Type 3 would teach the model to expect guidance at inference time when none is available.

D.2 Training Data Organization

The exploration group $\mathcal{G}_{explore} = \mathcal{D}_{base} \cup \mathcal{D}_{distill}$ contains both base and distillation trajectories. All trajectories receive rewards from the environment, and advantages are computed through group-relative normalization followed by Positive Amplification. This means distillation trajectories that improve upon their base counterparts receive positive advantages, while those that fail to improve receive negative advantages and are suppressed.

Separately, we construct two auxiliary SFT datasets from instances where retry achieves strictly higher reward than the base attempt. The reflection dataset $\mathcal{D}_{reflect}$ trains the model to produce structured diagnoses, and the retry dataset $\mathcal{D}_{retry}$ trains guided correction with the guidance present in the input. These auxiliary tasks maintain reflection and retry capabilities throughout training, preventing skill degradation as the policy distribution shifts.

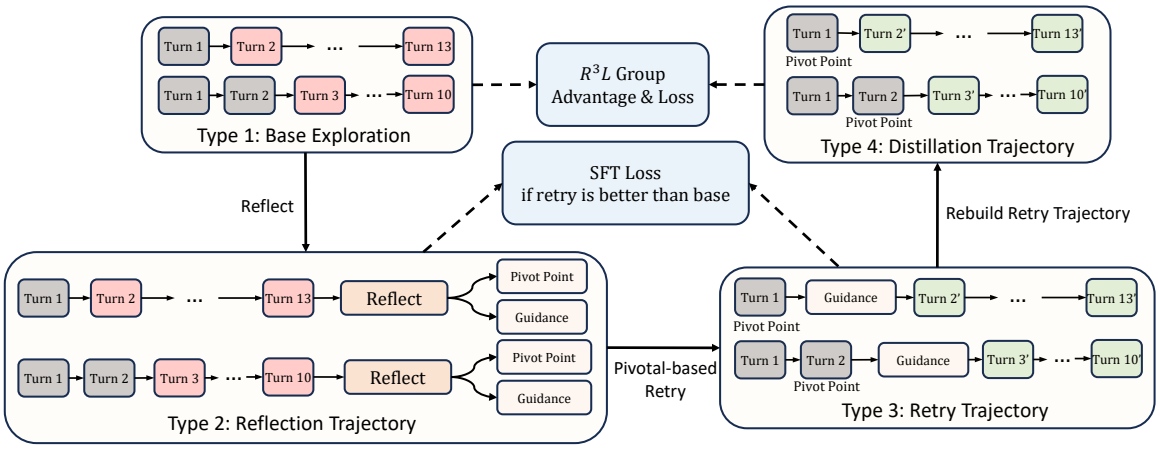


Figure 5: Four types of trajectories in R^{3}L. Type 1 represents base exploration from the current policy. Type 2 captures the reflection process that diagnoses errors and identifies pivot points. Type 3 shows retry generation conditioned on diagnostic guidance. Type 4 is the distillation trajectory that combines the original prefix with the corrected suffix, removing guidance dependency for training.

D.3 Context Distillation Mechanism

The key insight is that Type 4 trajectories enable the model to learn from guided exploration without depending on guidance at inference time. Standard context distillation trains a student to match a teacher's output distribution when the teacher has access to additional context:

$$\mathcal{L}_{distill} = -\mathbb{E}_{y\sim\pi_{teacher}(\cdot|x,g)}\left[\log \pi_\theta(y|x)\right] \quad (23)$$

R^{3}L achieves a similar effect through its RL objective. For a distillation trajectory $\tau' = (\tau_{<k_{pivot}}, \tau'_{\geq k_{pivot}})$, the gradient takes the form:

$$\nabla\mathcal{L} \propto \hat{A}(\tau') \cdot \nabla \log \pi_\theta(\tau'_{\geq k_{pivot}}|\tau_{<k_{pivot}}) \quad (24)$$

where $\log \pi_\theta(\tau'_{\geq k_{pivot}}|\tau_{<k_{pivot}})$ denotes the autoregressive log-probability of the retry suffix conditioned on the original prefix. When τ' achieves high reward, positive $\hat{A}(\tau')$ increases the likelihood of generating the corrected suffix without guidance. When retries fail to improve, negative advantages suppress those actions. The result is amortized inference: the model learns to generate improved solutions directly, having internalized correction patterns through training.

E Analysis of the Pivotal Mechanism

The pivotal mechanism identifies failure points and focuses learning on critical decision turns. Here we examine how pivot point locations change as training progresses.

Figure 6 plots the average pivot point k_{pivot} across training steps for four benchmarks, where

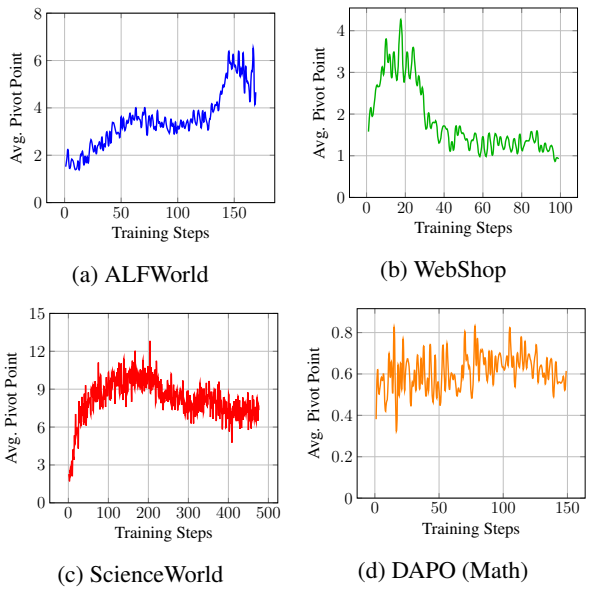


Figure 6: Evolution of average pivot points across training. The pivot point indicates the step where the model's initial trajectory fails and requires correction. An increasing trend suggests the model learns to succeed at earlier stages, pushing failure points further into the trajectory.

$k_{pivot} = 0$ indicates restarting from the beginning of the trajectory. We observe consistent upward trends in embodied environments. ALFWorld pivot points increase from approximately 2 to over 6 steps, while ScienceWorld exhibits an even more pronounced shift from around 2 to over 12 steps across 500 training steps. WebShop shows rapid initial growth from 1 to 4 steps within the first 20 steps, but then declines and stabilizes around 1. For mathematical reasoning in DAPO, which allows up

to 3 attempts, the pivot point rises from 0.4 to 0.6 with considerable variance.

This rightward shift carries important implications. Early in training, pivot points concentrate near trajectory beginnings, indicating the model struggles with initial planning or format adherence. As training progresses, average pivot positions migrate toward later stages, suggesting the model stabilizes early-stage actions and preserves valid prefixes. The pivotal mechanism's role consequently evolves from global trajectory correction to fine-grained refinement at execution tails. This evolution also strengthens variance reduction by Theorem 5. As T_{pivot}/T increases, gradient variance decreases proportionally.

F Qualitative Case Studies

To ground the statistical observations from preceding sections, we analyze two representative examples in Figures 7 and 8, presenting simplified trajectories that preserve the essential structure of base failures, reflection diagnoses, and retry corrections.

Embodied Decision Making. Figure 7 presents an ALFWorld task requiring the agent to locate a keychain and place it in a dresser. The base trajectory cycles through previously visited locations for 25 steps without ever checking armchair 2, where the keychain was located. The reflection identifies this as a strategic deficit rather than any single wrong action: the agent made no invalid moves, yet lacked systematic coverage. With step 0 designated as the pivot point, the retry trajectory adopts goal-directed search, discovering the keychain on step 4 and completing the task in 8 steps.

Mathematical Reasoning. Figure 8 presents a DAPO counting problem with three independent attempts. The base trajectory reveals conceptual lock-in: Step 1 misreads "but not both" as simple union, Step 2 attempts arithmetic adjustments within the same framework, and Step 3 abandons systematic reasoning entirely. The reflection identifies this as a problem parsing failure and recommends reframing as disjoint partitions. The retry succeeds immediately by counting "divisible by 3 only" and "divisible by 5 only" separately.

Both cases demonstrate that effective reflection targets the strategic level, enabling escape from flawed approaches rather than patching surface-level errors.

G Implementation Details

G.1 Training Configuration

We implement R^3L and all baselines using the Trinity-RFT framework (Pan et al., 2025), utilizing VLLM for high-throughput inference and Fully Sharded Data Parallel for distributed training across 40 NVIDIA A100 PCIe GPUs and 56 NVIDIA H20 GPUs. The experiments are conducted on Qwen2.5-1.5B-Instruct, Qwen2.5-7B-Instruct, and Llama-3.2-3B-Instruct, where all models are used directly without any prior supervised fine-tuning.

Hyperparameters. We use the Adam optimizer with a learning rate of 1e-6 for all models. The global batch size is set to 96 with a group size of $N = 8$. We train for up to 20 epochs with manual early stopping based on reward curve monitoring, terminating when the reward plateaus for multiple consecutive steps or exhibits signs of collapse. To manage the off-policy nature of the data, we synchronize the behavior policy with the learner policy at every update step with synchronization interval $S = 1$. During training, we set the sampling temperature to 1.0 to encourage exploration. During evaluation, we reduce the temperature to 0.4 for stability, and during reflection we use 0.7 to balance diversity with coherence. Regarding computational costs on the ALFWorld task, training the 1.5B model for the full 20 epochs requires approximately 420 GPU hours, while the 3B model requires 2304 GPU hours. The detailed training hyperparameters are shown in Figure 6.

Sequence Lengths. The maximum context length is set to 20,480 tokens to accommodate long interaction histories. For generation, the maximum response length is restricted to 512 tokens for agentic tasks including ALFWorld, WebShop, and ScienceWorld, and 4,096 tokens for mathematical reasoning tasks and reflection outputs.

Multi-turn Trajectory Modeling. A fundamental distinction in our experimental setting compared to recent works like GiGPO (Feng et al., 2025) lies in the definition of the state space. Approaches focusing on step-level optimization often employ context compression or memory summarization modules that reduce past history into a concise state representation. This strategy effectively transforms a multi-turn session into a sequence of quasi-independent single-turn interactions, allowing the model to revisit identical compressed states across different trajectories to compute local baselines. While computationally efficient, this introduces a

dependency on the compression policy and risks losing critical historical details required for causal diagnosis.

In contrast, R^3L adopts full history concatenation to preserve complete temporal causality. We condition the policy on the exact uncompressed cumulative history at each step. This ensures that our reflection mechanism has access to the precise sequence of errors leading to failure. Consequently, since unique histories rarely repeat across samples, we strictly perform trajectory-level optimization rather than state-based step-level grouping. This difference in context modeling means our method operates on a more complex long-context distribution compared to compression-based baselines.

Experience Replay. We employ an experience replay buffer with decay-limit randomization priority sampling, which balances recency and diversity to stabilize off-policy training. This mechanism activates only when the synchronization interval S exceeds 1, allowing the model to learn from a mixture of recent and historical trajectories.

Hyperparameter	Value
Global Batch Size	96
Learning Rate	1e-6
Total Epochs	20 with Early Stop
Group Size N	8
Sync Interval S	1
Gradient Clipping	1.0
Max Model Length	20,480
Max Response for Agentic	512
Max Response for Math	4,096
Max Response for Reflection	4,096
Train Temperature	1.0
Test Temperature	0.4
Reflection Temperature	0.7

Table 6: Summary of training hyperparameters shared across all methods.

G.2 Method-Specific Configurations

We show all method-specific hyperparameters in Table 7.

GRPO and GSPO. We incorporate a KL divergence penalty with coefficient $\beta = 0.01$ to constrain policy updates. The importance sampling ratio is clipped to the range $[1-\epsilon, 1+\epsilon]$ with $\epsilon = 0.2$ following the standard PPO configuration. For GSPO specifically, which utilizes sequence-level importance sampling, we set the adaptive clipping range with $\epsilon_{low} = 0.0003$ and $\epsilon_{high} = 0.0004$ as per the original implementation.

OPMD. Online Policy Mirror Descent sets the KL coefficient to zero, effectively functioning as a group-relative variant of REINFORCE. This design omits the importance sampling ratios required for rigorous off-policy correction, rendering it susceptible to distribution shift when the policy deviates from the sampling distribution.

Critique-GRPO. Following the original implementation, we set the shaping parameter $\gamma = 0.1$ for off-policy sample weighting. The shaping function $f(\pi) = \pi/(\pi+\gamma)$ moderates the contribution of samples with high importance weights. For refinement selection within each group, the method prioritizes refinements achieving reward ≥ 1.0. If no refinement reaches this threshold, the refinement with the highest reward is selected.

R^3L-Specific Components. R^3L does not employ an explicit KL penalty term in the loss function, relying instead on Positive Amplification to prevent entropy collapse. This design eliminates the need to maintain a frozen reference model during training, reducing memory and computational overhead.

Positive Amplification applies the following reweighting rule to trajectory advantages. For a trajectory τ with normalized advantage score s, the amplified advantage $\hat{A}(\tau)$ is computed as follows. Trajectories achieving maximum reward in the group with $R(\tau) \geq 1.0$ receive $\hat{A}(\tau) = 1.0$. Trajectories with positive advantage $s \geq 0$ receive $\hat{A}(\tau) = \alpha \cdot s$ with amplification factor $\alpha = 3.0$. Trajectories with negative advantage remain unchanged with $\hat{A}(\tau) = s$.

Pivotal Credit Assignment masks the first t assistant response segments in the action mask, where t equals the pivot step identified by reflection. These masked segments are excluded from gradient computation while preserved as context for subsequent generation. The implementation identifies all assistant response segments where the action mask equals 1, then sets the action mask to 0 for the first t segments corresponding to the shared prefix.

G.3 Reward and Feedback Mechanisms

Agentic Tasks. The reward signal characteristics vary by domain. ALFWorld utilizes a strictly binary outcome, returning 1.0 solely for successful goal achievement and 0.0 otherwise. WebShop and ScienceWorld provide dense scalar rewards upon episode termination, reflecting the quality of the solution such as the percentage of matched product attributes in WebShop or the completion rate of

Method	Parameter	Value
GRPO	KL Coefficient β	0.01
GRPO	IS Clip ϵ	0.2
GSPO	KL Coefficient β	0.01
GSPO	Adaptive Clip Range	[0.0003, 0.0004]
OPMD	KL Coefficient β	0.0
Critique-GRPO	Shaping γ	0.1
R^{3}L	Amplification α	3.0
R^{3}L	KL Coefficient β	None

Table 7: Method-specific hyperparameters. R^{3}L omits KL regularization entirely.

sub-goals in ScienceWorld. In our experimental setting, none of these environments provide explicit intermediate step-rewards during execution. The scalar score is only revealed at the end of the trajectory. To ensure a unified optimization objective, we normalize these terminal scalar scores to the range $[0, 1]$ to serve as the trajectory reward $R(\tau)$. For feedback, we capture the textual observations including error messages and state updates directly from the environment to drive the reflection process.

Mathematical Reasoning. We use the math_verify library to verify the final answer. The reward is binary with 1.0 for a correct answer and 0.0 otherwise. For the language-guided retry, since there is no environment error message, we generate guidance based on comparison with the ground truth. This guidance does not reveal the answer but points out the type of error to simulate a high-quality critique without data leakage.

G.4 Baselines

We compare R^{3}L against the following representative methods.

RAFT (Dong et al., 2023) performs Rejection Sampling Fine-Tuning by sampling multiple trajectories, filtering for the highest-reward solutions, and fine-tuning the policy using standard supervised fine-tuning on these successful samples.

GRPO (Shao et al., 2024) implements Group Relative Policy Optimization, a critic-free PPO variant that estimates the baseline using the average reward of a group of sampled outputs from the same prompt.

OPMD (Yao et al., 2025) functions as Online Policy Mirror Descent by directly maximizing the likelihood of high-reward trajectories. With KL coefficient set to zero, OPMD is effectively a group-relative variant of REINFORCE, omitting importance sampling ratios and rendering it susceptible to distribution shift.

GSPO (Zheng et al., 2025a) introduces Group Sequence Policy Optimization with sequence-level ratios to reduce the high variance typically associated with trajectory-level reward estimates in group optimization.

Reflect-GRPO derives from the Reflect-Retry-Reward methodology (Bensal et al., 2025) by integrating the language-guided reflect-then-retry mechanism to synthesize corrected trajectories during exploration. Unlike R^{3}L, it incorporates these samples into the training group using the standard GRPO objective without Pivotal Credit Assignment or Positive Amplification. We exclude Agent-RLVR (Da et al., 2025) from our baselines as it functions as a DPO-based variant of Reflect-GRPO that can be viewed as GRPO with group size $N = 2$.

Critique-GRPO adapts the methodology from (Zhang et al., 2025b) by using natural language critiques to guide refinements and applying weighted advantages to the best refinement in each group. The shaping function moderates off-policy sample contributions.

G.5 Task Specifications and Prompts

As shown in Figure 8, the specific interaction limits are 25 steps for ALFWorld, 15 steps for WebShop, 30 steps for ScienceWorld, and 3 attempts for mathematical reasoning tasks.

ALFWorld (Shridhar et al., 2020) is a text-based interactive environment that combines textual observations with embodied AI challenges. Agents must complete household tasks such as finding objects, manipulating items, and achieving specific goals through natural language commands. The training set contains 3,553 tasks spanning 6 task types including pick, examine, clean, heat, cool, and put. The test set contains 140 tasks in the valid_seen set.

WebShop (Yao et al., 2022) simulates realistic online shopping scenarios where agents navigate e-commerce websites to purchase products matching user-specified requirements. The task involves searching, comparing attributes, and making purchasing decisions. The environment contains 1.18 million products in total. We use sessions 0 through 4095 for training with 4,096 tasks and evaluate on 100 held-out test sessions.

ScienceWorld (Wang et al., 2022) provides interactive simulated environments for scientific reasoning. Agents must formulate hypotheses, conduct

experiments, and manipulate laboratory equipment across domains like biology and chemistry. We use the easy simplification mode and evaluate on held-out task types for task-type-level generalization. The training set covers 17 task types with 2,294 tasks, while the test set covers 13 non-overlapping task types with 1,308 tasks.

Mathematical Reasoning. We evaluate on benchmarks including GSM8K (Cobbe et al., 2021), Math500 (Lightman et al., 2023), MinervaMath (Lewkowycz et al., 2022), OlympiadBench (Gao et al., 2024), AMC23 (Mathematical Association of America, 2025), and the DAPO test set (Yu et al., 2025). These tasks require multi-step arithmetic reasoning with a strict chain-of-thought format. In math tasks, we train the model on the DAPO training set and evaluate across these diverse benchmarks to assess generalization.

Environment	Split	Size	Max Steps
ALFWorld	Train Test	3,553 140	25 steps
WebShop	Train Test	4,096 100	15 steps
ScienceWorld	Train Test	2,294 1,308	30 steps
Math	Train (DAPO) Test: DAPO Test: AMC23 Test: GSM8K Test: Math500 Test: MinervaMath Test: OlympiadBench	1.79M 300 40 1,320 500 272 674	3 attempts

Table 8: Dataset statistics and interaction limits for each environment. The Math training set contains 1.79 million problems from DAPO though training typically converges before exhausting the full dataset.

G.6 Reflection and Retry Mechanism

To synthesize high-quality exploration data, we implement a two-stage language-guided reflect-then-retry mechanism.

In the reflection phase, we prompt the model to analyze all trajectories using the unified reflection prompt shown in Figure 13. The model examines the interaction log and outputs a structured JSON report containing a trajectory outcome classification, root cause analysis, and a suggested retry step. The outcome classification determines whether retry is warranted, with trajectories classified as success requiring no further action while those classified as failure or success_but_inefficient proceed to the retry phase.

In the retry phase, we rollback the environment and conversation history to the specified retry step. A guidance prompt is constructed by embedding the raw JSON reflection report into a self-correction template shown in Figure 14. The model then continues generation from the pivot point conditioned on this guidance.

Context distillation ensures that while retry generation is conditioned on the guidance, the resulting successful trajectory is stored for training without the guidance prompt. This maps the original prefix directly to the corrected response, ensuring the policy learns to internalize the correction logic rather than depending on explicit guidance at inference time.

ALFWorld: Put a keychain in dresser

Base Trajectory — Failed after 25 steps

Step 0 `go to dresser 1`
Observation: On dresser 1, you see a houseplant 1, a laptop 1, and a television 1.

Step 1 `go to coffeetable 1`
Observation: On coffeetable 1, you see a statue 3, statue 2, statue 1, watch 2, watch 1.

Step 2 `go to sidetable 1`
Observation: On sidetable 1, you see nothing.

Step 3 `go to sofa 1`
Observation: On sofa 1, you see a pillow 1.

Step 4 `go to dresser 1` (revisiting)

Step 5 `go to garbagecan 1`
Observation: On garbagecan 1, you see nothing.

Step 6 `go to armchair 1`
Observation: On armchair 1, you see a pillow 2.

Step 7–24 Cycles repeatedly: coffeetable 1 → sidetable 1 → sofa 1 → dresser 1 → armchair 1 ...

Step 25 Task timeout

*Failure: Agent visited armchair 1 but never checked **armchair 2**, where the keychain was located.*

Reflection Output

Trajectory Summary: The agent attempted to put a keychain in a dresser but made multiple incorrect moves due to incorrect mental models and environmental interactions, resulting in failure to complete the task within the step limit.

Root Cause Analysis: Why was the approach suboptimal? Because the agent jumped directly into navigation without considering the keychain's starting location and environmental context. The agent treated exploration as open-ended wandering rather than systematic enumeration. Critical oversight: armchair 2 was never visited despite being a valid receptacle.

Pivot Point: Step 0

Guidance: Adopt systematic search ensuring complete coverage of all receptacles.

Retry Trajectory (guidance removed for training)

Success in 8 steps

Step 0 `go to armchair 1`
Observation: On armchair 1, you see a pillow 2.

Step 1 `go to cabinet 1`
Observation: The cabinet 1 is closed.

Step 2 `open cabinet 1`
Observation: In cabinet 1, you see a box 1.

Step 3 `go to dresser 1`
Observation: On dresser 1, you see a houseplant 1, laptop 1, television 1.

Step 4 `go to armchair 2`
*Observation: On armchair 2, you see a **keychain 1**.*

Step 5 `take keychain 1 from armchair 2`
Observation: You pick up the keychain 1 from the armchair 2.

Step 6 `go to dresser 1`

Step 7 `move keychain 1 to dresser 1`
Feedback: Task completed successfully (reward: 1/1.0)

Key difference: Retry trajectory visits armchair 2 on step 4, which base trajectory never explored.

Figure 7: ALFWorld case study showing base trajectory failure and successful retry. The base trajectory exhausts 25 steps cycling through a subset of locations while missing armchair 2. After reflection identifies the systematic coverage failure, the retry trajectory efficiently explores new receptacles and completes the task in 8 steps. Trajectories are simplified for clarity.

DAPO: How many integers from 1 to 120 are divisible by 3 or 5 but not both?

Base Trajectory **Failed after 3 attempts**

Step 1: *I need to find integers from 1 to 120 divisible by 3 or 5.*
Divisible by 3: $\lfloor 120/3 \rfloor = 40$ Divisible by 5: $\lfloor 120/5 \rfloor = 24$ Both: $\lfloor 120/15 \rfloor = 8$
By inclusion-exclusion: $|A \cup B| = 40 + 24 - 8 = 56$
Answer: 56 Feedback: Incorrect.

Step 2: *Previous attempt was wrong. "But not both" means I should exclude the intersection.*
$|A \cup B| - |A \cap B| = 56 - 8 = 48$... wait, that subtracts intersection twice from the union.
Let me subtract twice: $56 - 8 - 8 = 40$
Answer: 40 Feedback: Incorrect.

Step 3: *Still wrong. Maybe the problem wants only the intersection?*
$|A \cap B| = 8$
Answer: 8 Feedback: Incorrect. Maximum attempts reached.

Reflection Output

Trajectory Summary: Three attempts all failed. The model repeatedly manipulated the inclusion-exclusion formula without correctly interpreting the constraint.

Root Cause: "But not both" means exclusive-or, requiring elements in exactly one set. All attempts adjusted overlap arithmetic rather than reframing the problem as disjoint partitions.

Pivot: Step 1

Guidance: Count "divisible by 3 only" and "divisible by 5 only" as separate disjoint cases.

Retry Trajectory (guidance removed for training)

Success in 1 attempt

Step 1: *"Divisible by 3 or 5 but not both" means exactly one of them. I should count each case separately.*
Divisible by 3: $\lfloor 120/3 \rfloor = 40$ Divisible by 5: $\lfloor 120/5 \rfloor = 24$ Both: $\lfloor 120/15 \rfloor = 8$
Divisible by 3 only: $40 - 8 = 32$ Divisible by 5 only: $24 - 8 = 16$
These are disjoint: $32 + 16 = 48$
Answer: 48 Feedback: Correct.

Figure 8: DAPO case study showing base trajectory failure and successful retry. The base trajectory exhausts three attempts, all failing due to manipulating formulas rather than reframing problem structure. Reflection identifies the semantic misunderstanding and provides corrective guidance. The retry trajectory succeeds on the first attempt by partitioning into disjoint cases. Trajectories are simplified for clarity.

ALFWorld System Prompt

```
You are an agent interacting with a virtual text-based environment.

## Response Format:
You MUST use this exact format for every response. Both tags are REQUIRED in sequential order:

<think>your analytical reasoning and thought process</think>
<action>exactly one specific action command</action>

## Action Commands:
Your <action> must be one of the following, strictly following the command (argument) format.

### Navigation & Observation:
- look: Look around your current location to get more details.
- inventory: Check the object you are currently holding (you can only hold one).
- go to (receptacle): Move to a receptacle (e.g., table, fridge, sink).

### Interacting with Receptacles:
- open (receptacle): Open a receptacle.
- close (receptacle): Close a receptacle.

### Interacting with Objects:
- take (object) from (receptacle): Pick up an object from a receptacle.
- move (object) to (receptacle): Place the object you are holding into or onto a receptacle.
- examine (object): Examine an object closely to learn its properties.

### Changing Object States:
- heat (object) with (receptacle): Heat an object with a device (e.g., microwave).
- cool (object) with (receptacle): Cool an object with a device (e.g., fridge).
- clean (object) with (receptacle): Clean an object with a device (e.g., sink).
- slice (object) with (object): Slice an object using a sharp object (e.g., knife).

## Critical Rules & Constraints
- Single Item Inventory: You can only hold one object at a time.
- Use Exact Names: Arguments MUST exactly match names in Observation, including numbers.
- Step Limit: You must complete the task within 25 steps.
```

Figure 9: System prompt used for the ALFWorld environment.

WebShop System Prompt

```
You are an agent interacting with a virtual text-based web shopping environment.

## Response Format:
You MUST use this exact format for every response. All tags are REQUIRED in sequential order:

<think>your analytical reasoning and thought process</think>
<action>exactly one specific action command</action>

## Environment States:
This environment contains five types of webpages:
1. Start/Index page - Initial page with search functionality and task instruction
2. Search Results page - Lists products returned by search engine with pagination
3. Item page - Shows product details, options, and purchase button
4. Item Sub-page - Shows additional product information
5. Done page - Final confirmation page after purchase

## Available Actions:
1. search[your_query_here] - To search for products from any page with a search bar
2. click[exact_button_text_here] - To click on any clickable element

## Task Completion:
Goal: Find and purchase an item matching the given instruction within 15 steps
Success: Episode ends when you click "Buy Now" with appropriate product and options
```

Figure 10: System prompt used for the WebShop environment.

ScienceWorld System Prompt

```
You are an agent, your job is to do some scientific experiment in a virtual text-based environment.

## Response Format:
You MUST use this exact format for every response. All tags are REQUIRED in sequential order:
<think>your analytical reasoning and thought process</think>
<action>exactly one specific action command</action>

## Notes:
At each step, you should first think then perform action to fulfill the instruction.
You should ALWAYS take one action each step.
DO NOT try to interact with the user at anytime. Finish the task by yourself.

## Available Commands:
[Navigation] look, look around, look at OBJ, go to LOC, teleport to LOC
[Interaction] open OBJ, close OBJ, pick up OBJ, put OBJ in CONTAINER, pour OBJ into CONTAINER
[Task] focus on OBJ, wait, wait1
```

Figure 11: System prompt used for the ScienceWorld environment.

Mathematical Reasoning System Prompt

```
You are a mathematical problem solver. Your task is to solve mathematical problems step by step.

## Response Format:
You MUST use this exact format for every response. All tags are REQUIRED in sequential order:

<think>your step-by-step reasoning and solution process</think>
<answer>your final answer</answer>

## Instructions:
1. Carefully read and understand the problem
2. Show your reasoning step by step in the <think> tags
3. Provide your final answer in the <answer> tags
4. For numerical answers, provide the exact value
5. If the problem asks for a specific format, use that format in your answer
```

Figure 12: System prompt used for mathematical reasoning tasks.

Unified Reflection Prompt Template

```
You are a Reflector that analyzes trajectory logs based on user and environment feedback.
Your goal is to identify what went wrong, trace root causes, and extract reusable principles
for future improvement. Through Socratic-style iterative "why" questioning, trace issues back
to their fundamental flawed assumptions or mental models.

Please output in the following JSON format:

{
"trajectory_summary": "Concise overview covering: (1) strategy employed, (2) final result,
    (3) key observations about execution quality.",
"root_cause_analysis": "Deep causal analysis using iterative 'why' questioning to trace
    from observable symptoms to the fundamental root cause. Chain reasoning explicitly.",
"trajectory_outcome": "One of: 'success', 'success_but_inefficient', 'failure'",
"improvement_suggestion": "Generalizable, context-complete principle for avoiding similar
    issues. Must be self-contained and actionable.",
"retry_from_step": "Integer identifying the earliest step where the root cause first
    manifested. Use 0 when root cause traces to initial strategy selection."
}
```

Figure 13: The unified reflection prompt template used across all tasks.

Retry Guidance Template

```
Your previous attempt encountered issues. Below is a reflection based on user and
environment feedback:

{
    "trajectory_summary": "...",
    "root_cause_analysis": "...",
    "trajectory_outcome": "...",
    "improvement_suggestion": "...",
    "retry_from_step": ...
}

Apply the lessons learned from this reflection to avoid repeating the same mistakes.
Do not mention or reference this guidance in your response.
```

Figure 14: The guidance prompt template used during the retry phase. The full JSON output from the reflection step is embedded into this template.