Who Should Evolve? Uncertainty-Aware Role Bottleneck Inference for Multi-Agent LLM Training

Qiyu Qin[1], **Yichen Li**[1], **Haozhao Wang**[1]*, **Tianzhe Xiao**[1],
Hao Zhou[2], **Imran Razzak**[3], **Ruixuan Li**[1]
[1]School of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan, China
[2]Nanjing University of Posts and Telecommunications, Nanjing, China
[3]Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates
{qiyu_qin, ycli0204, hz_wang, rxli}@hust.edu.cn

Abstract

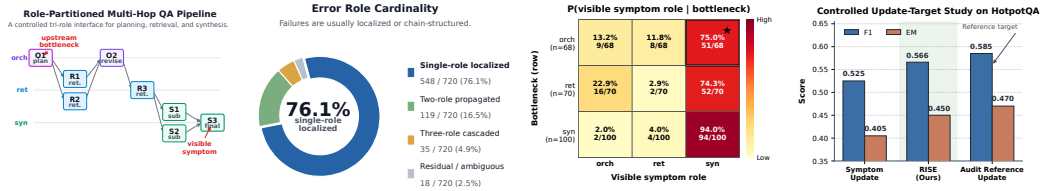
Reinforcement learning from trajectory-level outcomes has become a standard approach for training role-partitioned LLM agent pipelines, in which multiple specialized roles such as an orchestrator, a retriever, and a synthesizer collaborate to solve complex reasoning tasks. Upon trajectory failure, existing training methods do not explicitly determine which role should receive updates, applying outcome signals to all roles indiscriminately or diffusely across the pipeline. However, this paper identifies that failures can often be traced to a primary responsible role, which we term the *bottleneck role*. This suggests that updating all roles may contaminate non-bottleneck roles with irrelevant updates. Therefore, an intuitive solution is to exclusively update the bottleneck role for each failed trajectory. Unfortunately, accurately *identifying this role is highly challenging*, as we show that roles exhibiting visible symptoms of failure are often merely downstream victims of errors originating from other roles, rather than being the bottleneck. To tackle this challenge, we propose **R**ole **I**nference for **S**elective **E**volution (*RISE*), the first method that formulates per-trajectory update-target selection as role bottleneck inference. Specifically, RISE leverages Beta-binomial lower-confidence scores to estimate role responsibility, effectively discounting uncertain evidence and suppressing updates when attribution confidence is insufficient. Furthermore, RISE decouples the visible symptom locus from the support-constrained responsibility target prior to applying selective role-level training. Empirically, across HotpotQA, 2WikiMultihopQA, and MuSiQue with Qwen2.5-7B and Llama3-8B backbones, RISE consistently outperforms the strongest learning-based baseline, with up to **11.1%** relative F1 and **14.1%** relative EM improvements over GiGPO on MuSiQue with Llama3-8B. Code is available at: `https://github.com/QinWHang/RISE`.

1 Introduction

Role-partitioned LLM agent pipelines have emerged as a standard architecture for complex reasoning, with prior systems organizing LLM behavior through modular tool use, multi-agent collaboration, or explicit role specialization [13, 22, 15, 30, 11, 7]. In retrieval-intensive and multi-hop question answering, such pipelines are naturally instantiated as planning, retrieval, and answer synthesis stages, since systems must decompose a question, gather evidence, and integrate intermediate facts into a final answer [33, 14, 26, 2]. As such pipelines become trainable rather than purely prompted systems, reinforcement learning from trajectory-level outcomes offers a practical supervision signal:

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Tri-role, eight-step multi-hop QA pipeline. (b) Most failures are localized to a single role. (c) Visible symptoms often differ from bottlenecks. (d) Responsibility-aware targets improve updates.

Figure 1: Motivation for selective role evolution. The panels show the role-partitioned QA setting, primary responsibility structure, symptom–bottleneck mismatch, and the benefit of responsibility-aware update targeting.

the full trajectory can be judged as successful or failed even when intermediate role supervision is noisy or unavailable [37, 21, 3].

Existing training methods for LLMs and agents mainly address how sparse outcome feedback should be transformed into denser or lower-variance learning signals. Outcome-level policy optimization methods such as RLOO [1] and GRPO [24] reduce variance using leave-one-out baselines or group-relative advantages, but still optimize from trajectory-level rewards. Process supervision and step verifiers provide more localized feedback for intermediate reasoning steps [18, 27], while recent agent RL methods construct step-wise advantages or implicit step rewards for long-horizon trajectories [6, 20]. Despite their improved training stability and data efficiency, existing methods do not address the role-level update-target problem: after observing a trajectory outcome, the system must infer which specialized role provides the justified target for policy update.

To investigate this, we consider a fixed tri-role, eight-step QA pipeline in Figure 1a. As shown in Figure 1b, most observed failures in this pipeline are *localized to a single role*, with propagated cases typically following an upstream-to-downstream chain, motivating selective updates to supported responsibility targets rather than indiscriminate updates to all roles. Unfortunately, accurately *identifying this role is highly challenging*, as the system often exhibits both a *symptom role* where an error becomes visible, and a *bottleneck role* whose behavior provides the evidence-supported reason for updating. For example, an upstream planning or retrieval error can propagate through subsequent roles, even though the final observable failure manifests in the synthesizer. In fact, Figure 1c shows that this symptom-bottleneck divergence is common: when the orchestrator is the reference bottleneck, only 13.2% of failures are visibly localized to the orchestrator, while 75.0% surface at the downstream synthesizer; similarly, retriever bottlenecks surface at the synthesizer in 74.3% of cases. Figure 1d further shows that this mismatch affects training outcomes. In a controlled update-target study with the same backbone, rollout pipeline, and training budget, symptom-based updating reaches 0.525 F1 and 0.405 EM, whereas updating the AUDIT REFERENCE target reaches 0.585 F1 and 0.470 EM. These observations motivate formulating per-trajectory update-target selection as role bottleneck inference rather than symptom repair or joint updating.

To address this problem, we propose **R**ole **I**nference for **S**elective **E**volution (RISE), an uncertainty-aware framework that identifies which role should evolve after a trajectory-level outcome. RISE first converts noisy judge signals into role-level posterior-strength scores, allowing the system to form a conservative raw attribution target rather than relying on raw step scores. It then uses confidence-margin abstention to withhold updates when competing role explanations are not sufficiently separated. For failed trajectories, RISE further performs support-constrained responsibility refinement, which separates the visible symptom locus from the responsibility target and redirects repair only when upstream bottleneck evidence is sufficiently supported. Finally, RISE applies selective role-level training by updating the inferred target role or freezing all roles when attribution is uncertain. This framework turns role-level credit assignment into an explicit bottleneck inference problem, rather than treating all roles or the visibly failing role as the default update target.

We evaluate RISE on three multi-hop QA benchmarks, HotpotQA, 2WikiMultihopQA, and MuSiQue, using a controlled role-partitioned pipeline with orchestrator, retriever, and synthesizer roles. Across Qwen2.5-7B-Instruct and Llama3-8B-Instruct, RISE consistently outperforms the strongest learning baseline, with up to **11.1%** relative F1 and **14.1%** relative EM gains over GiGPO on MuSiQue with Llama3-8B. RISE also achieves these gains with fewer gradient-producing role events and lower off-target update rates, improving answer quality with more selective updates. Together, these results

support the view that role-level target selection is a central training problem in role-partitioned LLM pipelines. Our contributions are threefold:

- We formulate selective evolution in role-partitioned LLM pipelines as a role bottleneck inference problem, distinguishing the visible symptom role from the responsibility target that should receive training updates.
- We introduce RISE, an uncertainty-aware framework that combines posterior-strength role scoring, confidence-margin abstention, and support-constrained responsibility refinement to update one supported role or freeze all roles.
- We provide end-task and mechanism evidence showing that selective role evolution improves multi-hop QA performance while reducing unnecessary and off-target role updates.

2 Related Work

Reinforcement learning and credit assignment for LLM agents. Reinforcement learning from outcome-level feedback has become central to aligning and improving language models [37, 21, 3]. Outcome-level methods such as RLOO [1] and GRPO [24] optimize policies from trajectory-level rewards using leave-one-out or group-relative baselines. To provide denser supervision, process reward models introduce intermediate feedback for reasoning tasks [18, 27], while agent RL methods such as GiGPO [6] and iStar [20] construct step-level advantages or implicit step rewards for long-horizon trajectories. ECPO [16] calibrates step-level advantages using evidence counts and variance, while GEPO [36] uses state-transition graphs to assign structural credit at both step and trajectory levels. Recent multi-agent reinforcement fine-tuning methods further extend this setting to interacting LLM agents under heterogeneous roles or collaborative protocols [17, 5, 19, 9]. These works improve how sparse outcome signals are distributed across trajectories, steps, or agents, but they do not address the update-target decision in structured role pipelines: the visible failure may occur downstream while the responsible bottleneck lies upstream, so step- or agent-level credit does not determine which role should evolve. RISE treats this assignment as uncertainty-aware bottleneck inference rather than reward refinement.

Multi-agent QA and RAG pipelines. Retrieval-augmented generation grounds language model outputs in external evidence [14], and multi-hop QA requires systems to retrieve and combine evidence across multiple supporting facts [33]. Recent work improves retrieval quality and reasoning depth through interleaved retrieval [26], self-reflective critique [2], and agentic tool use [35]. Multi-agent frameworks extend the RAG decomposition to specialized agents for planning, retrieval, verification, and answer synthesis [30, 7, 4]. These systems share RISE's use of specialized interacting roles, but their focus is inference-time collaboration: how roles communicate, filter evidence, and synthesize answers. RISE addresses the complementary training-time problem of selecting the responsible role under noisy trajectory outcomes while avoiding updates to non-bottleneck roles.

Agent evolution and selective training. A growing body of work studies self-improving LLM agents that evolve through iterative interaction or experience-driven training. R-Zero [12] co-evolves a Challenger and Solver through curriculum self-play; RAGEN [29] identifies multi-turn instability phenomena such as echo traps; EvolveR [31] builds a closed-loop experience distillation lifecycle; and recent work analyzes misevolution risks across model, memory, tool, and workflow pathways [23]. Role-Agent [28] uses a single LLM as both agent and environment to improve process feedback and adapt the training task distribution. These works treat evolution as a system- or policy-level process and optimize for aggregate performance or long-term behavioral stability. RISE addresses a more localized mechanism within this setting: given a trajectory outcome in a role-partitioned pipeline, it infers which role is the supported bottleneck and freezes non-responsible roles to prevent credit contamination during iterative training.

3 Methodology

In this section, we formulate selective evolution in role-partitioned LLM pipelines and present Role Inference for Selective Evolution (RISE), an uncertainty-aware framework for inferring which role should evolve. We define the update-target problem, then describe posterior-strength scoring, margin-based abstention, support-constrained responsibility refinement, and selective role updating.

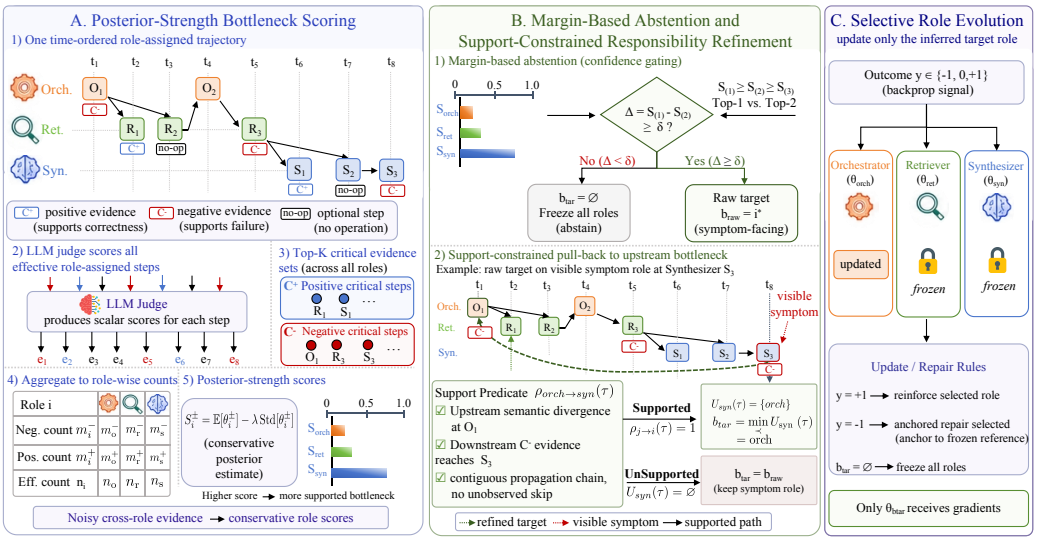


Figure 2: Overview of RISE. Given a role-partitioned trajectory, RISE converts noisy step evidence into role-wise posterior-strength scores, applies margin-based abstention to select a raw target or freeze all roles, and performs support-constrained responsibility refinement on failed trajectories to redirect downstream symptom-facing targets to supported upstream bottlenecks. The final selective-evolution step updates only the inferred target role, while all non-target roles remain frozen.

3.1 Problem Formulation

We consider a role-partitioned LLM agent pipeline with a finite role set $\mathcal{R}$, where each role corresponds to a specialized functional component. A trajectory τ consists of T role-assigned steps. At step t, the system observes a context h_t, produces an action or textual output a_t, and the step is assigned to a role $\rho_t \in \mathcal{R}$. We write the trajectory as

$$\tau = \{(h_t, a_t, \rho_t)\}_{t=1}^{T}, \qquad \rho_t \in \mathcal{R}. \tag{1}$$

Our evaluation uses $\mathcal{R} = \{\mathrm{orch}, \mathrm{ret}, \mathrm{syn}\}$ with a maximum schema of $T = 8$ slots as the primary QA configuration; §4.3 further evaluates alternative role and step constructions. Each role $i \in \mathcal{R}$ is associated with a role-conditioned policy π_{θ_i}, where θ_i denotes the trainable parameters of that role. The role-specific action distribution is $\pi_{\theta_i}(a_t \mid h_t)$, with the role identity encoded in the context or prompt. Selective evolution updates only the selected role's parameters, leaving all others fixed.

Let $\mathcal{E}_\tau \subseteq \{1, \ldots, T\}$ denote the effective steps in trajectory τ, excluding no-op steps that do not produce a non-trivial action. For each role i, the role-specific effective step set is

$$\mathcal{T}_i = \{t \in \mathcal{E}_\tau \mid \rho_t = i\}, \tag{2}$$

and the trajectory outcome is denoted by $y \in \{+1, 0, -1\}$, where $+1$ indicates success, -1 indicates failure, and 0 denotes a neutral outcome that is frozen before routing.

The objective is to improve trajectory-level outcomes while avoiding unnecessary updates to roles that are not responsible for the observed result. We therefore seek an update target

$$b \in \mathcal{R} \cup \{\varnothing\}, \tag{3}$$

where $b = i$ means that only role i is updated, and $b = \varnothing$ means that all roles remain frozen. For analysis, let $b^\star(\tau) \in \mathcal{R}$ denote the ideal responsibility target, which is not observed during training. Let $v(\tau) \in \mathcal{R}$ denote the visible symptom role, i.e., the role where the failure is most directly exposed in the trajectory. In sequential role pipelines, upstream errors can surface downstream, so $v(\tau)$ may differ from $b^\star(\tau)$. RISE therefore estimates a supported update target from noisy step evidence, rather than defaulting to all roles or to the visible symptom role.

3.2 RISE: Role Inference for Selective Evolution

RISE frames update target selection as a latent bottleneck attribution problem: given noisy step-level judge evidence, it infers which role is most responsible for the observed outcome, abstains when

attribution confidence is insufficient, and refines failure-side targets when visible symptoms point to the wrong role. Figure 2 illustrates this attribution-and-update procedure, and Algorithm 1 in Appendix B.4 summarizes the full training algorithm.

Posterior-strength bottleneck scoring. RISE uses an LLM judge as its source of step-level attribution evidence. Since judge scores are noisy proxies rather than causal labels, RISE treats role responsibility as a latent quantity and estimates it via role-wise Beta-Binomial posteriors, discounting sparse or inconsistent evidence rather than treating raw judge scores as direct supervision. Given a trajectory τ, an LLM judge (configuration and prompt in Appendix B.1) assigns a scalar evidence score $s_t \in \mathbb{R}$ to each effective step $t \in \mathcal{E}_\tau$. Because these scores are not directly comparable across trajectories, we normalize them within each trajectory and extract a critical evidence set $\mathcal{C}$ containing the highest-magnitude normalized scores:

$$\begin{aligned} z_t &= \frac{s_t - \mu_\tau}{\sigma_\tau + \epsilon}, \\ \mathcal{C} &= \mathrm{TopK}_{t \in \mathcal{E}_\tau}(|z_t|), \\ \mathcal{C}^+ = \{t \in \mathcal{C} \mid z_t > 0\}, &\qquad \mathcal{C}^- = \{t \in \mathcal{C} \mid z_t < 0\}. \end{aligned} \tag{4}$$

The trajectory-level statistics μ_τ and σ_τ are the within-trajectory mean and standard deviation of judge scores, and $\epsilon > 0$ is a numerical stability constant. The sets $\mathcal{C}^+$ and $\mathcal{C}^-$ capture the strongest positive and negative attribution evidence, rather than assigning a label to every step. RISE then aggregates critical evidence into role-wise counts:

$$m_i^+ = |\mathcal{C}^+ \cap \mathcal{T}_i|, \qquad m_i^- = |\mathcal{C}^- \cap \mathcal{T}_i|, \qquad n_i = |\mathcal{T}_i|, \tag{5}$$

where m_i^+ and m_i^- count the positive and negative critical steps assigned to role i, and n_i is the number of effective steps for that role. RISE models these counts as observations from a role-wise Beta-Binomial distribution to form per-role posterior-strength scores. Distinguishing by outcome, we define outcome-conditioned posteriors ($+$ for $y = +1$, $-$ for $y = -1$):

$$\theta_i^\pm \mid \tau \sim \mathrm{Beta}(\alpha + m_i^\pm,\, \beta + n_i - m_i^\pm), \qquad S_i^\pm = \mathbb{E}[\theta_i^\pm] - \lambda\, \mathrm{Std}[\theta_i^\pm], \tag{6}$$

with Beta prior parameters $\alpha, \beta > 0$ and an uncertainty penalty $\lambda \geq 0$; hyperparameter values are reported in Appendix C.2. These scores are posterior-strength estimates rather than normalized role probabilities: unlike raw hit ratios, they discount sparse or inconsistent evidence, and a weak margin across roles naturally triggers abstention rather than forcing an uncertain update. Because Top-K selection couples evidence across roles, the Beta-Binomial model is a working approximation for evidence rates, not a calibrated posterior over responsibility.

Margin-based abstention and support-constrained responsibility refinement. RISE first converts posterior-strength scores into a raw target through margin-based abstention. Neutral trajectories ($y = 0$) are assigned $b_{\mathrm{tar}} = \varnothing$ and skipped. For a non-neutral trajectory, let $S_i = S_i^+$ if $y = +1$ and $S_i = S_i^-$ if $y = -1$. Denote the top-scoring role as $i^* = \arg\max_{i \in \mathcal{R}} S_i$ and the top-2 margin as $\Delta = S_{i^*} - \max_{i \neq i^*} S_i$. The raw target is

$$b_{\mathrm{raw}} = \begin{cases} i^*, & \Delta \geq \delta, \\ \varnothing, & \Delta < \delta \end{cases}, \tag{7}$$

where $\delta > 0$ is a confidence threshold. When $\Delta < \delta$, RISE treats the trajectory as attribution-ambiguous and sets $b_{\mathrm{tar}} = \varnothing$, skipping directly to selective role evolution without applying a loss. We set $\delta = 0.03$ in all experiments unless otherwise specified. When $b_{\mathrm{raw}} \neq \varnothing$, RISE proceeds to determine the final target b_{tar}.

When $b_{\mathrm{raw}} \neq \varnothing$, RISE sets $b_{\mathrm{tar}} = b_{\mathrm{raw}}$ for successful trajectories. For failed trajectories, however, the raw target may correspond to the visible symptom rather than the update-worthy bottleneck. RISE therefore performs support-constrained responsibility refinement before repair. This refinement is a pull-back test rather than a universal condition for responsibility: it is used only to decide whether a downstream symptom-facing target should be reassigned to an upstream cause. If the raw target has no upstream candidate, such as an orchestrator target, refinement is skipped and the raw target is retained. Let $i = b_{\mathrm{raw}}$ be the symptom-facing raw target. For each upstream role $j \prec i$, we define a trajectory-local support predicate $\rho_{j \to i}(\tau) \in \{0, 1\}$ that is true only when visible trajectory evidence

supports role j as an upstream responsibility source for the symptom at role i. Implementation details are provided in Appendix B.2. In our implementation, this predicate requires three conditions: role j exhibits semantic divergence in its output, the downstream path from j to i contains negative evidence consistent with error propagation, and the evidence chain from j to i is contiguous and does not rely on unobserved intermediate steps. The set of supported upstream causes is

$$\mathcal{U}_i(\tau) = \{j \in \mathcal{R} : j \prec i,\ \rho_{j \to i}(\tau) = 1\}, \tag{8}$$

and the final target is

$$b_{\mathrm{tar}} = \begin{cases} \min_{\prec} \mathcal{U}_{b_{\mathrm{raw}}}(\tau), & y = -1,\ b_{\mathrm{raw}} \neq \varnothing,\ \mathcal{U}_{b_{\mathrm{raw}}}(\tau) \neq \emptyset, \\ b_{\mathrm{raw}}, & \text{otherwise.} \end{cases} \tag{9}$$

Here $\min_{\prec}$ selects the most upstream element of $\mathcal{U}_{b_{\mathrm{raw}}}(\tau)$ under the primary configuration's order $\mathrm{orch} \prec \mathrm{ret} \prec \mathrm{syn}$. Thus, refinement redirects repair from a downstream symptom to an upstream bottleneck only when the trajectory contains explicit support for that relation; otherwise, the raw target is retained.

Selective role evolution. Given the final responsibility target b_{tar}, RISE applies an outcome-conditioned update only to the selected role:

$$\mathcal{L}_{\mathrm{RISE}} = \begin{cases} \mathcal{L}_{\mathrm{rein}}(b_{\mathrm{tar}}), & y = +1,\ b_{\mathrm{tar}} \neq \varnothing, \\ \mathcal{L}_{\mathrm{rep}}(b_{\mathrm{tar}}), & y = -1,\ b_{\mathrm{tar}} \neq \varnothing, \\ 0, & b_{\mathrm{tar}} = \varnothing. \end{cases} \tag{10}$$

For successful trajectories, RISE reinforces the selected contributing role on its positive critical steps:

$$\mathcal{L}_{\mathrm{rein}}(i) = -\sum_{t \in \mathcal{C}^+ \cap \mathcal{T}_i} \hat{A}_\tau\, w_t \log \pi_{\theta_i}(a_t \mid h_t), \tag{11}$$

where $\hat{A}_\tau$ is the group-relative trajectory advantage computed over multiple rollouts of the same question, as in GRPO [24], and w_t is the softmax weight of z_t over steps in $\mathcal{C}^+ \cap \mathcal{T}_i$. We apply the reinforcement branch only when the rollout group has non-zero reward variance, so that the group-relative advantage is informative.

For failed trajectories, RISE applies a contrastive repair objective to the selected role. Repair pairs are built from negative critical steps of b_{tar} when the target is local, or from supported critical steps of the refined upstream target when responsibility has been reassigned. Let $\mathcal{P}_i$ denote the repair-pair set for role i, where each $(h, a^+, a^-) \in \mathcal{P}_i$ consists of a failure action a^- and a matched repair action a^+. The repair action is drawn preferentially from successful rollouts of the same question; when unavailable, it is matched from successful trajectories with the same role, step type, and sufficient context overlap (see Appendix B.3). We use

$$\mathcal{L}_{\mathrm{rep}}(i) = -\mathbb{E}_{(h,a^+,a^-)\sim\mathcal{P}_i} \log \sigma\left(\eta \left[\log \frac{\pi_{\theta_i}(a^+ \mid h)}{\pi_{\mathrm{ref}}(a^+ \mid h)} - \log \frac{\pi_{\theta_i}(a^- \mid h)}{\pi_{\mathrm{ref}}(a^- \mid h)}\right]\right), \tag{12}$$

where π_{ref} is a frozen reference policy and $\eta > 0$ controls the repair temperature. This contrastive objective discourages the identified failure actions while anchoring parameter movement to the reference policy. When $b_{\mathrm{tar}} = \varnothing$, Eq. (10) applies no loss, leaving all role parameters frozen.

3.3 Analysis: Credit Contamination Control

RISE controls credit contamination by masking gradients for all non-selected roles. Let $b^\star(\tau) \in \mathcal{R}$ denote the ideal responsibility target defined in §3.1. For a training rule with role-wise gradients $\mathbf{g}_i(\tau)$, we define credit contamination as the gradient magnitude assigned to non-responsible roles:

$$\Gamma(\tau) = \sum_{i \neq b^\star(\tau)} \|\mathbf{g}_i(\tau)\|_2 . \tag{13}$$

Proposition 3.1 (Contamination accounting)**.** *For any trajectory τ, RISE assigns non-zero gradients only to the final target role b_{tar}; if $b_{\mathrm{tar}} = \varnothing$, all role gradients are zero. If there exists a uniform constant $G > 0$ such that $\|\mathbf{g}_i(\tau)\|_2 \leq G$ for all roles and trajectories, then*

$$\mathbb{E}_\tau[\Gamma_{\mathrm{RISE}}(\tau)] \leq G \cdot p_{\mathrm{err}}, \tag{14}$$

where $p_{\mathrm{err}} = \mathbb{E}_\tau[\mathbb{I}[b_{\mathrm{tar}} \neq b^\star(\tau),\ b_{\mathrm{tar}} \neq \varnothing]]$ is the non-abstaining target error rate.

Proof sketch. By Eq. (10), the RISE loss depends only on the selected role policy when $b_{\text{tar}} \neq \varnothing$, and is zero when $b_{\text{tar}} = \varnothing$. Therefore,

$$\nabla_{\theta_i} \mathcal{L}_{\text{RISE}}(\tau) = \mathbf{0}, \qquad \forall i \neq b_{\text{tar}}, \tag{15}$$

and RISE can contaminate a non-responsible role only when it actively updates an incorrect target:

$$\Gamma_{\text{RISE}}(\tau) = \mathbb{I}[b_{\text{tar}} \neq b^\star(\tau),\ b_{\text{tar}} \neq \varnothing]\, \|\mathbf{g}_{b_{\text{tar}}}(\tau)\|_2 . \tag{16}$$

Taking expectation over trajectories and applying the bounded-gradient assumption gives Eq. (14). A fuller derivation is provided in Appendix B.5.

Proposition 3.1 bounds contamination by Gp_{err}; attribution must therefore control p_{err}.

Attribution condition. For failed trajectories, let p_i be the probability that an effective step of role i enters $\mathcal{C}^-$. Under rank-faithfulness, $\gamma = p_{b^\star} - \max_{i \neq b^\star} p_i > 0$. If $\max_i |S_i^- - p_i| \leq \xi$ with $\xi \geq 0$ and $\gamma > \delta + 2\xi$, then $\Delta > \delta$ and $b_{\text{raw}} = b^\star$. Beta-LCB accounts for posterior uncertainty rather than evidence ratios alone (Appendix B.5). Judge bias or downstream symptom concentration can violate rank-faithfulness; refinement then depends on valid upstream support. On the failure audit, the empirical counterpart of p_{err}, measured against the AUDIT REFERENCE, is $0.580(1-0.983) \approx 0.010$, the off-target rate in Figure 4. The condition specifies when the raw target is correct, while the audit evaluates the final routing decisions.

4 Experiments

4.1 Experiment Setup

Benchmarks. We evaluate RISE on three multi-hop QA benchmarks: HotpotQA [33], 2WikiMultihopQA [10], and MuSiQue [25]. These benchmarks require agents to retrieve and synthesize evidence across multiple supporting facts, providing natural settings where orchestrator planning or retrieval failures can propagate to downstream answer errors, the core phenomenon RISE addresses. The pipeline accommodates varying example complexity through the no-op mechanism defined in §3.1: simpler questions complete with fewer effective role decisions, while compositional questions can activate the full orchestration, retrieval, and synthesis sequence. Dataset construction and retrieval pool setup are provided in Appendix D.1.

Models and Baselines. We use Qwen2.5-7B-Instruct [32] as the primary backbone and Llama-3-8B-Instruct [8] for robustness evaluation. For external comparisons, we include MAIN-RAG [4] as a training-free multi-agent RAG reference, RLOO [1] and GRPO [24] as outcome-level RL baselines, and role-pipeline adaptations of GiGPO [6] and iStar [20] as credit-assignment baselines with group-relative or step-level signals. All learning-based baselines use matched rollout budgets; adaptation and budget details are provided in Appendices D.3 and D.4. For mechanism analysis, we additionally compare controlled update rules: NO-UPDATE performs no training, RANDOM ROLE updates a randomly selected role, ATTRIBUTION-WEIGHTED allocates updates by aggregated role scores, and JOINT ROLE updates all roles. AUDIT REFERENCE uses an offline responsibility reference unavailable to RISE during training, while SYMPTOM UPDATE selects the visible symptom role. Both serve as target-selection diagnostics rather than deployable baselines.

Evaluation Protocol. Agents answer using only the question and retrieval corpus, without gold answers, supporting-fact labels, or reference update targets. We report token-level F1, Exact Match (EM), mean parse-fallback events per rollout (PF↓), and the fraction of training trajectories with a non-empty update target (Upd). Update-efficiency metrics include trajectory counts (Traj.), gradient-producing role-level loss events (Active), and event intensity $\text{Event-Int.} = \text{Active}/(\text{Traj.} \times |\mathcal{R}|)$. Multiple events from one RISE trajectory affect only its selected role. Metric details appear in §4.3 and Appendix D.5; hyperparameters are in Appendix C.2.

4.2 Main Results

QA Performance. Table 1 reports results across three benchmarks and two backbones. RISE achieves the highest F1 and EM in all six model–dataset conditions, with up to 11.1% relative F1 and 14.1% relative EM improvements over the strongest learning baseline. The consistent ranking across Qwen2.5-7B and Llama3-8B suggests that the gains reflect the training paradigm rather than a backbone-specific effect.

Table 1: Main results on multi-hop QA benchmarks. F1 and EM measure answer quality; PF reports mean parse-fallback events per rollout; Upd is the fraction of training trajectories with a non-empty update target (0 for non-learning methods). Learning methods use matched rollout budgets.

Model	Method	HotpotQA				2WikiMultihopQA				MuSiQue			
		F1	EM	PF↓	Upd	F1	EM	PF↓	Upd	F1	EM	PF↓	Upd
Qwen2.5-7B	MAIN-RAG	0.527	0.410	0.010	0.00	0.438	0.350	0.012	0.00	0.326	0.240	0.018	0.00
	RLOO	0.518	0.400	0.024	0.86	0.431	0.342	0.061	0.88	0.319	0.232	0.074	0.84
	GRPO	0.536	0.420	0.021	0.91	0.449	0.363	0.056	0.93	0.338	0.252	0.066	0.89
	GiGPO	0.542	0.425	0.018	0.94	0.456	0.370	0.052	0.95	0.344	0.258	0.061	0.92
	iStar	0.533	0.417	0.022	0.89	0.445	0.356	0.058	0.91	0.335	0.247	0.070	0.87
	RISE	**0.566**	**0.450**	**0.009**	**0.34**	**0.481**	**0.395**	**0.020**	**0.37**	**0.369**	**0.282**	**0.034**	**0.32**
Llama3-8B	MAIN-RAG	0.456	0.330	0.014	0.00	0.315	0.235	0.016	0.00	0.238	0.170	0.021	0.00
	RLOO	0.447	0.325	0.032	0.84	0.306	0.225	0.072	0.86	0.229	0.162	0.083	0.82
	GRPO	0.462	0.338	0.027	0.90	0.327	0.246	0.067	0.92	0.247	0.178	0.077	0.88
	GiGPO	0.469	0.342	0.024	0.93	0.332	0.252	0.063	0.94	0.253	0.184	0.072	0.91
	iStar	0.461	0.335	0.029	0.87	0.321	0.241	0.068	0.89	0.243	0.175	0.079	0.85
	RISE	**0.489**	**0.365**	**0.013**	**0.36**	**0.356**	**0.275**	**0.023**	**0.39**	**0.281**	**0.210**	**0.038**	**0.35**

Table 2: Selective update efficiency on HotpotQA using Qwen2.5-7B-Instruct. Role-level loss event counts are normalized to 720 trajectories.

Method	Norm. Traj.	Active	Event-Int.	Rel. Upd.
RLOO	720	1188	0.550	2.23×
GRPO	720	1188	0.550	2.23×
GiGPO	720	1332	0.617	2.50×
iStar	720	2052	0.950	3.86×
RISE	720	**532**	**0.246**	**1.00×**

Table 3: Update mechanism comparison on HotpotQA using Qwen2.5-7B-Instruct. Learning variants share the same rollout budget.

Strategy	F1	EM	PF	Upd.
No Update	0.382	0.285	0.030	0.00
Random Role	0.493	0.372	0.041	0.38
Attr.-Weighted	0.512	0.390	0.032	0.42
Joint Role	0.542	0.420	0.018	0.94
RISE	**0.566**	**0.450**	**0.009**	**0.34**

Two baseline patterns motivate selective role updating. First, RLOO underperforms the training-free MAIN-RAG baseline in all six conditions, indicating that outcome-level RL does not consistently outperform inference-time collaboration. Second, GiGPO improves over outcome-level RL but retains a consistent gap to RISE, suggesting that finer-grained credit alone does not close the performance gap. Joint-update RL baselines also incur more parse-fallback events under the same structured-output prompt, whereas RISE updates supported roles or abstains. Thus, RISE improves answer quality without trading away structured-output stability relative to these learning baselines.

Selective Update Efficiency. Table 2 compares update events on a common 720-trajectory scale. Baseline diagnostic counts are linearly rescaled from smaller runs to match the RISE diagnostic log. RISE produces 532 events, corresponding to an event intensity of 0.246; RLOO/GRPO, GiGPO, and iStar produce 2.23×, 2.50×, and 3.86× as many events, respectively. RISE therefore achieves higher QA scores with fewer gradient-producing events, although this does not imply lower total training time because attribution adds overhead.

4.3 Mechanism Analysis

Update Mechanism. Table 3 compares how update-target rules affect adaptation. Random Role and Attr.-Weighted improve over No Update but remain below RISE. Joint Role improves performance through co-adaptation, but all roles receive trajectory-level gradients regardless of responsibility. RISE achieves the strongest performance with lower update coverage, supporting selective responsibility inference from noisy evidence. These results support preventing unsupported gradient flow to non-bottleneck roles, rather than merely increasing or reweighting updates.

LLM Judge Validity. We audit whether the step-level judge provides a usable role-attribution signal. On 100 frozen HotpotQA failure trajectories from Qwen2.5-7B-Instruct, covering 700 non-noop steps, two human annotators independently label each step as severity 0, 1, or 2: no visible issue, weak failure evidence, or clear failure-relevant error. The audit marks critical steps and excludes judge scores, routing decisions, selected roles, and refinement targets (see Appendix D.6 for the annotation protocol). Figure 3 shows that judge scores are noisy but ordered: mean failure-evidence

increases monotonically with audit severity. This supports RISE's use of ranked critical evidence rather than perfectly accurate step scores. The top-ranked judge step matches an audited critical step in 98% of trajectories, supporting the judge as a strong critical-evidence retrieval signal whose remaining noise is handled by aggregation, abstention, and refinement.

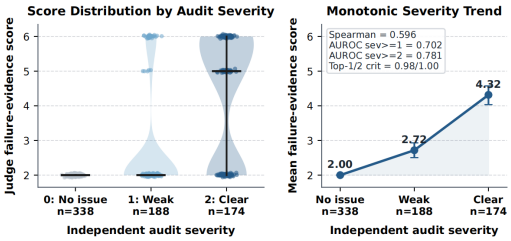


Figure 3: LLM judge validity audit on frozen HotpotQA trajectories.

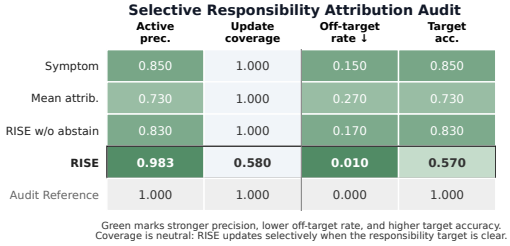


Figure 4: Responsibility target audit on frozen HotpotQA trajectories.

Responsibility Target Audit. Figure 4 audits whether each method identifies a supported responsibility target against the AUDIT REFERENCE, rather than merely updating a role. *Active precision* measures correctness among executed updates, *update coverage* measures how often a method updates, *off-target rate* measures wrong-role updates, and *target accuracy* counts abstentions as missed targets. RISE does not maximize unconditional target accuracy. By abstaining on uncertain cases, it reduces coverage to 0.580 but raises active precision to 0.983 and lowers off-target rate to 0.010, far below symptom updating. This asymmetry matters: abstention forgoes one update opportunity, whereas a wrong-role update injects contaminating gradients into a non-bottleneck role, making off-target rate the direct evidence for RISE's contamination-control claim. Notably, RISE w/o abstention performs worse than symptom updating in active precision (0.830 vs. 0.850). This shows that refinement alone is insufficient: margin-based abstention improves update reliability.

Role/step Construction Sensitivity. Table 4 tests whether RISE depends on the exact canonical 3-role/8-step decomposition. *Action Agree.* measures whether the routing action agrees with the canonical setting, where the action is repair, reinforce, or freeze rather than an exact role name; *Active Rate* measures how often the router triggers a non-freeze update. All variants achieve at least 0.975 in action agreement, with active rates shifting by at most 0.033. The larger drop under retrieval splitting reflects that retrieval steps carry the bulk of failure evidence in multi-hop QA, so finer-grained retrieval boundaries increase attribution ambiguity. Thus, RISE is robust to reasonable role/step constructions while relying on enough evidence to identify responsibility. Additionally, sensitivity to the margin threshold δ is reported in Appendix E.1.

Table 4: Role/step sensitivity on HotpotQA using Qwen2.5-7B-Instruct. 4R orch split separates planner and controller roles, 4R ret split separates primary and follow-up retrievers, and 3R-6S uses six effective steps.

Setting	Roles/Steps	Action Agree.	Active Rate
3R-8S canonical	3 / 8	1.000	0.900
4R orch split	4 / 8	0.992	0.892
4R ret split	4 / 8	0.983	0.875
3R-6S projected	3 / 6	0.975	0.867

Table 5: Ablation study on HotpotQA using Qwen2.5-7B-Instruct.

Variant	F1	EM	PF↓	Upd.
w/o target refinement	0.512	0.385	0.019	0.43
w/o anchored repair	0.526	0.402	0.036	0.34
w/o Beta-LCB	0.538	0.417	0.048	0.39
w/o repair role gate	0.545	0.425	0.033	0.48
w/o margin abstention	0.552	0.434	0.021	0.86
RISE	**0.566**	**0.450**	**0.009**	**0.34**

Ablation Study. Table 5 shows that RISE's components address distinct failure modes. Removing target refinement causes the largest F1 drop, highlighting the importance of support-constrained target refinement. Removing anchored repair or Beta-LCB raises PF substantially (to 0.036 and 0.048), showing that attribution calibration and repair anchoring improve structured-output stability. Removing margin abstention has the smallest F1 impact but more than doubles the update rate, confirming that abstention mainly controls training selectivity. Together, RISE selects the responsible role, supports structured-output stability, and filters unsupported trajectories.

5 Conclusion

We introduced RISE, an uncertainty-aware framework that recasts selective role evolution in role-partitioned LLM pipelines as a bottleneck inference problem. By inferring the supported responsibility role, abstaining under attribution uncertainty, and confining each update to a single inferred target, RISE improves multi-hop QA performance while reducing credit contamination. Experiments and mechanism audits indicate that role-level target selection is a central lever for training structured LLM agent systems.

Acknowledgments and Disclosure of Funding

This work was supported by the National Key Research and Development Program of China under Grant 2024YFC3307900; the National Natural Science Foundation of China under Grants 62302184, 62376103, 62436003, 62472332, and 62672205; the Major Science and Technology Project of Hubei Province under Grants 2025BAB011, 2024BAA008, and 2024BAB039; the Hubei Science and Technology Talent Service Project under Grant 2024DJC078; the Science and Technology Innovation Talent Program under grant 2023CB005; and Wuhan Dameng Database Co., Ltd. The authors declare no competing interests beyond the funding disclosed above.

References

[1] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024.

[2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2023.

[3] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[4] Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, et al. Main-rag: Multi-agent filtering retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2607–2622, 2025.

[5] Guanzhong Chen, Shaoxiong Yang, Chao Li, Wei Liu, Jian Luan, and Zenglin Xu. Heterogeneous group-based reinforcement learning for llm-based multi-agent systems. *arXiv preprint arXiv:2506.02718*, 2025.

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

[7] Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

[8] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[9] Zhitao He, Zijun Liu, Peng Li, Yi R Fung, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. Advancing language multi-agent learning with credit re-assignment for interactive environment generalization. *arXiv preprint arXiv:2502.14496*, 2025.

[10] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, 2020.

[11] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

[12] Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

[13] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, et al. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*, 2022.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[15] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

[16] Yuanfan Li, Qi Zhou, Wenjing Duan, and Lu Chen. When denser credit is not enough: Evidence-calibrated policy optimization for long-horizon llm agent training. *arXiv preprint arXiv:2606.05885*, 2026.

[17] Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. Marft: Multi-agent reinforcement fine-tuning. *arXiv preprint arXiv:2504.16129*, 2025.

[18] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

[19] Muhan Lin, Shuyang Shi, Yue Guo, Vaishnav Tadiparthi, Behdad Chalaki, Ehsan Moradi Pari, Simon Stepputtis, Woojun Kim, Joseph Campbell, and Katia Sycara. Speaking the language of teamwork: Llm-guided credit assignment in multi-agent reinforcement learning. *arXiv preprint arXiv:2502.03723*, 2025.

[20] Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. Agentic reinforcement learning with implicit step rewards. *arXiv preprint arXiv:2509.19199*, 2025.

[21] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[22] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

[23] Shuai Shao, Qihan Ren, Chen Qian, Boyi Wei, Dadi Guo, Jingyi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, Dongrui Liu, et al. Your agent may misevolve: Emergent risks in self-evolving llm agents. *arXiv preprint arXiv:2509.26354*, 2025.

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

[25] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

[26] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037, 2023.

[27] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

[28] Xucong Wang, Ziyu Ma, Shidong Yang, Tongwen Huang, Pengkun Wang, Yong Wang, and Xiangxiang Chu. Role-agent: Bootstrapping llm agents via dual-role evolution. *arXiv preprint arXiv:2606.10917*, 2026.

[29] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

[30] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.

[31] Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

[32] Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024.

[33] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380, 2018.

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

[35] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[36] Jiazhen Yuan, Zhike Gong, Jinquan Hang, Zhengbiao Bai, and Wei Zhao. Graph-enhanced policy optimization in llm agent training. *arXiv preprint arXiv:2510.26270*, 2025.

[37] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Limitations

RISE studies selective updates in role-partitioned pipelines with known upstream dependencies and observable support paths. The current evaluation centers on multi-hop QA; dynamic role creation, cyclic collaboration, and open-ended tool-use tasks remain untested. Attribution quality depends on the judge's evidence rankings and on the support predicates; systematic judge bias or support rules that do not fit a new domain can route updates to the wrong role. The AUDIT REFERENCE targets used in our audits are operational references rather than uniquely identifiable causal ground truth. Finally, judge calls introduce training-time overhead, so reduced update-event counts do not directly imply shorter wall-clock time. The attribution judge is not required during evaluation or deployment.

B Method Details

B.1 LLM Judge and Step Evidence

Judge configuration. In the main experiments, the judge is a separate frozen copy of the corresponding policy backbone: Qwen2.5-7B-Instruct for the Qwen experiments and Llama-3-8B-Instruct for the Llama experiments. It uses no trainable role adapters and receives no gradient updates. Judge decoding uses temperature 0 and at most 64 output tokens. The judge is invoked only during training, not during evaluation or deployment.

Judge input and output. RISE uses an LLM judge to assign step-level evidence scores before role-level attribution. For each non-neutral trajectory, the judge scores each non-noop step separately using only visible trajectory information. If the trajectory outcome is neutral, all step scores are set to 0. No-op steps are also assigned score 0 and excluded from effective evidence statistics. For a target step, the judge input contains the trajectory summary, target step index, role id, visible context JSON, visible action JSON, visible next-state JSON, and the final outcome bucket, i.e., success, failure, or neutral. The judge is required to output a JSON object:

```
{"better_matters": true, "worse_matters": false, "score": 3}
```

Here `score` is a signed integer in $[-10, 10]$. A positive score indicates evidence that the step helped the observed outcome, a negative score indicates evidence that the step may be a failure bottleneck or that replacing it could improve a failed trajectory, and zero indicates neutral, uncertain, invalid, or directionless evidence.

Judge prompt. The formal judge prompt follows a counterfactual step-evaluation lens: it asks whether the target step appears to push the rollout toward success, or whether replacing that step with a better action could plausibly improve the final outcome. The prompt restricts the judge to observable trajectory evidence and requires JSON-only output. We use the following prompt template:

LLM judge prompt template

```
You are JudgePrompt-v1c for RISE.
Evaluate one target step with a counterfactual lens, but only from
observable trajectory evidence.

Trajectory summary:
{trajectory_summary}

Target step payload:
- index: {target_step_idx}
- role: {role_id}
- visible context: {context_json}
- visible action: {action_json}
- visible next state: {next_state_json}
- final outcome bucket: {outcome_short}

Required reasoning shape:
1. Did this step appear to push the run toward success?
2. Could swapping this step with a better one plausibly improve the ending?
3. Emit one signed integer score from -10 to 10.

Return JSON only:
{{"better_matters": true, "worse_matters": false, "score": 3}}
```

Output validation and leakage control. Judge outputs are strictly validated before being used as evidence. A valid response must be a JSON dictionary containing `better_matters`, `worse_matters`, and `score`; the two flags must be Boolean; the score must be an integer within $[-10, 10]$; and the direction must be consistent with the flags. Specifically, positive scores require `better_matters=true` and `worse_matters=false`, while negative scores require the reverse. Invalid or direction-inconsistent responses are mapped to neutral evidence with score 0. Apart from the final outcome bucket, judge inputs are recursively checked to exclude oracle or gold fields, including oracle answers, oracle documents, oracle document identifiers, supporting-fact overlap, retrieval-quality signals, answer-quality signals, and completeness signals. If any forbidden field appears in the visible context, action, or next state, prompt construction fails rather than silently leaking hidden supervision.

Step evidence construction. After collecting raw judge scores, RISE converts them into trajectory-local critical evidence using Eq. (4). The normalization statistics are computed only over non-noop steps. No-op steps receive normalized score 0 and are excluded from role-level evidence counts.

If the within-trajectory standard deviation is zero, all normalized scores are set to 0. The critical evidence set uses

$$K = \min(|\mathcal{E}_\tau|, \max(2, \mathrm{round}(0.4|\mathcal{E}_\tau|))) , \tag{17}$$

so each trajectory selects at least two effective steps when available and otherwise selects all effective steps. The selected critical steps are then split into $\mathcal{C}^+$ and $\mathcal{C}^-$ as defined in Eq. (4), and only these signed critical steps are aggregated into the role-level posterior-strength estimator.

B.2 Support-Constrained Responsibility Refinement

Inputs and role order. Responsibility refinement is applied only after the router has selected a failure-side repair route. Its inputs include the router-selected raw target b_{raw}, role-level posterior-strength scores, raw judge scores, normalized step evidence, the global absolute critical-step set, and the fixed role-step mapping. In the primary QA configuration, steps are mapped to roles as

$$\mathrm{orch} : \{1, 4\}, \qquad \mathrm{ret} : \{2, 3, 5\}, \qquad \mathrm{syn} : \{6, 7, 8\}.$$

The induced role order is

$$\mathrm{orch} \prec \mathrm{ret} \prec \mathrm{syn},$$

and upstream responsibility is considered only along this order.

Symptom view and responsibility view. The refinement module keeps two diagnostic views. The *symptom view* selects the role with the strongest failure-side posterior score, i.e., the role where negative evidence is most concentrated. This role often corresponds to the visible failure locus. The *responsibility view* instead aggregates signed negative critical steps by role, using both count and negative mass. If the top two roles are tied, the responsibility view returns no candidate rather than forcing a target. This separation is used to decide whether the visible symptom role should remain the repair target or be reassigned to an upstream supported cause.

Support predicate. The implementation instantiates the support predicate $\rho_{j \to i}(\tau)$ from §3 as a constrained propagation test. Semantic divergence is detected from visible trajectory fields: a role is marked divergent when its output changes the intended plan, query, evidence set, or intermediate answer in a way inconsistent with the task evidence. The support label is derived from the frozen trajectory record. For an upstream role j to be accepted as a supported responsibility source for symptom role i, three conditions must hold. First, j must be the earliest role with semantic-divergence evidence and negative raw judge evidence that also appears in the global critical-step set. Second, the immediate downstream role must contain negative critical evidence, so the upstream signal is not isolated. Third, every role on the ordered path from j to i must carry downstream negative evidence. This prevents unsupported skips: for example, an ORCH→SYN pullback is not allowed unless RET also carries negative evidence. Equivalently, $\rho_{j \to i}(\tau) = 1$ only when j is the first supported semantic-divergence role, i is the symptom-view role, and the full ordered path from j to i is evidence-supported.

Target resolution. If no raw target is selected, $b_{\mathrm{tar}} = \varnothing$. Successful trajectories retain $b_{\mathrm{tar}} = b_{\mathrm{raw}}$ without refinement. For failed trajectories, the default is local repair with $b_{\mathrm{tar}} = b_{\mathrm{raw}}$. RISE redirects the target upstream only when the responsibility view reason is `upstream_supported`, the supported-cause set is non-empty, and a first semantic-divergence role exists. In that case, the target becomes the earliest supported upstream cause; otherwise, the selected raw target is retained. Thus, refinement does not automatically pull downstream symptoms upstream: it does so only when the trajectory contains explicit support for propagation from the upstream role to the symptom role.

Target step selection and logs. For local repair, target steps are negative critical steps of the selected role. For upstream-supported repair, target steps are critical steps of the refined upstream target role; they are not required to satisfy an additional negative threshold because they are selected as responsibility-critical evidence. For audit and reproducibility, the implementation logs the symptom-view candidate, responsibility-view candidate, structural ancestors, reachable propagation chain, supported propagation chain, supported cause roles, first negative raw-score role, first semantic-divergence role, responsibility-view reason, final refinement target, target source, and whether the target changed from the router-selected role.

B.3 Repair Pair Construction

RISE constructs repair pairs only for the selected responsibility target, rather than for every step in a failed trajectory. For local repair, the target steps are negative critical steps of b_{tar}, i.e., steps assigned to the selected role, included in the critical evidence set, and satisfying the negative-evidence threshold. When support-constrained refinement redirects the target to an upstream role, the refinement module passes explicit target step indices; these steps are treated as responsibility-critical and are used directly, without requiring an additional negative-threshold check. Each target step contributes a negative action a^- from the failed trajectory. Steps caused by parser fallback or parse failure are skipped, since such actions are not reliable model decisions and should not be used as contrastive negatives.

For each negative action, RISE constructs a matched positive action a^+ using a prioritized source order. The preferred source is a successful rollout of the same question and rollout group with the same role and step index. If this is unavailable, RISE searches a replay buffer for successful actions with the same role, compatible step type, and sufficient context overlap. For retrieval actions, the positive action must be grounded in the current candidate documents: selected titles must appear in the step's candidate document set. The implementation also skips identical positive-negative pairs, invalid positive actions, and replay matches below the similarity threshold. These checks ensure that the repair pair represents an executable role-local correction rather than an artificial target. Logs record the positive-action source and skipped-pair reasons. The resulting repair-pair set $\mathcal{P}_i$ is used in Eq. (12).

B.4 RISE Algorithm

Algorithm 1 summarizes the full target-inference and selective-update procedure.

The algorithm first converts judged step evidence into role-level posterior-strength scores, then applies margin-based abstention to avoid uncertain updates. For failed trajectories, it additionally applies support-constrained responsibility refinement before constructing the final repair target. Only the final target role receives a reinforcement or repair update; when the target is $\varnothing$, all role policies remain unchanged.

B.5 Credit Contamination Analysis

We provide the derivation for Proposition 3.1. For a trajectory τ, let $\mathbf{g}_i(\tau) = \nabla_{\theta_i}\mathcal{L}(\tau)$ denote the gradient assigned to role i. We assume a *uniform deterministic* gradient bound: there exists a constant $G > 0$, independent of the trajectory τ and of the target-selection event, such that $\|\mathbf{g}_i(\tau)\|_2 \le G$ for all roles $i \in \mathcal{R}$ and all trajectories.

By Eq. (10), if $b_{\text{tar}} = \varnothing$, then $\mathcal{L}_{\text{RISE}}(\tau) = 0$ and all role gradients are zero. If $b_{\text{tar}} = j \in \mathcal{R}$, the loss depends only on the selected role policy π_{θ_j}, so all non-selected roles receive zero gradient:

$$\mathbf{g}_i(\tau) = \mathbf{0}, \qquad \forall i \neq j.$$

Therefore, RISE can contaminate a non-responsible role only when it actively updates an incorrect target, i.e., when $b_{\text{tar}} \neq b^\star(\tau)$ and $b_{\text{tar}} \neq \varnothing$. Using the contamination definition in Eq. (13), this gives

$$\Gamma_{\text{RISE}}(\tau) = \mathbb{I}[b_{\text{tar}} \neq b^\star(\tau),\ b_{\text{tar}} \neq \varnothing]\, \|\mathbf{g}_{b_{\text{tar}}}(\tau)\|_2 .$$

Applying the uniform bound $\|\mathbf{g}_{b_{\text{tar}}}(\tau)\|_2 \le G$ pointwise yields

$$\Gamma_{\text{RISE}}(\tau) \le G \cdot \mathbb{I}[b_{\text{tar}} \neq b^\star(\tau),\ b_{\text{tar}} \neq \varnothing] .$$

Because G is deterministic and trajectory-independent, it factors out of the expectation, giving

$$\mathbb{E}_\tau[\Gamma_{\text{RISE}}(\tau)] \le G \cdot \mathbb{E}_\tau[\mathbb{I}[b_{\text{tar}} \neq b^\star(\tau),\ b_{\text{tar}} \neq \varnothing]] = G \cdot p_{\text{err}},$$

which is Eq. (14). If only a non-uniform stochastic bound is available, the same expression holds with G replaced by $\mathbb{E}_\tau[\|\mathbf{g}_{b_{\text{tar}}}(\tau)\|_2 \mid b_{\text{tar}} \neq b^\star(\tau),\ b_{\text{tar}} \neq \varnothing]$, recovering the deterministic case under the uniform-bound assumption.

Abstained trajectories contribute zero contamination, so the bound depends on the non-abstaining target error rate p_{err}.

Algorithm 1: RISE: Role Inference for Selective Evolution

Input : Trajectory $\tau = \{(h_t, a_t, \rho_t)\}_{t=1}^{T}$; outcome $y \in \{+1, 0, -1\}$; role set $\mathcal{R}$; role policies $\{\pi_{\theta_i}\}_{i\in\mathcal{R}}$; routing hyperparameters $\delta, \alpha, \beta, \lambda$.

Output : Updated role policies $\{\pi_{\theta_i}\}_{i\in\mathcal{R}}$.

if $y = 0$ **then**
  Keep all role policies unchanged and **return**;
end
`// Posterior-strength scoring`
Obtain step evidence scores $\{s_t\}_{t\in\mathcal{E}_\tau}$ from the LLM judge;
Normalize to $\{z_t\}$ and construct $\mathcal{C}^+, \mathcal{C}^-$ via Eq. (4);
Compute role counts $\{m_i^+, m_i^-, n_i\}_{i\in\mathcal{R}}$ via Eq. (5);
Compute posterior-strength scores $\{S_i^+, S_i^-\}_{i\in\mathcal{R}}$ via Eq. (6);
Set $S_i \leftarrow S_i^+$ if $y = +1$, else $S_i \leftarrow S_i^-$, for all $i \in \mathcal{R}$;
`// Margin-based abstention`
$i^* \leftarrow \arg\max_{i\in\mathcal{R}} S_i$;
$\Delta \leftarrow S_{i^*} - \max_{i\neq i^*} S_i$;
if $\Delta < \delta$ **then**
  $b_{\mathrm{tar}} \leftarrow \varnothing$;
else
  $b_{\mathrm{raw}} \leftarrow i^*$;
  $b_{\mathrm{tar}} \leftarrow b_{\mathrm{raw}}$;
  `// Support-constrained responsibility refinement`
  if $y = -1$ **then**
    Compute $\mathcal{U}_{b_{\mathrm{raw}}}(\tau)$ via Eq. (8);
    if $\mathcal{U}_{b_{\mathrm{raw}}}(\tau) \neq \emptyset$ **then**
      $b_{\mathrm{tar}} \leftarrow \min_{\prec} \mathcal{U}_{b_{\mathrm{raw}}}(\tau)$;
    end
  end
end
`// Selective role evolution`
if $b_{\mathrm{tar}} = \varnothing$ **then**
  Keep all role policies unchanged;
else
  if $y = +1$ **then**
    if *group reward variance* > 0 **then**
      Update only $\theta_{b_{\mathrm{tar}}}$ with $\mathcal{L}_{\mathrm{rein}}(b_{\mathrm{tar}})$ via Eq. (11);
    else
      Keep all role policies unchanged;
    end
  else
    Build repair pairs $\mathcal{P}_{b_{\mathrm{tar}}}$ and update only $\theta_{b_{\mathrm{tar}}}$ with $\mathcal{L}_{\mathrm{rep}}(b_{\mathrm{tar}})$ via Eq. (12);
  end
end

Attribution condition. Under the condition in §3.3,

$$S_{b^\star}^- - \max_{i\neq b^\star} S_i^- \geq \gamma - 2\xi > \delta.$$

Thus, $b_{\mathrm{raw}} = b^\star$ and the abstention gate is passed.

Finite-sample scoring. With $\alpha = \beta = \lambda = 1$, two negative critical steps out of two effective steps give $S(2, 2) \approx 0.556$, while three out of three give $S(3, 3) \approx 0.637$. Both have an evidence ratio of one, but their score difference exceeds $\delta = 0.03$, illustrating the effect of evidence volume.

C Implementation and Hyperparameters

C.1 Pipeline, Prompts, and No-op Steps

This subsection specifies the common rollout interface and logging contract used by RISE and role-based baselines. Given a question and its candidate retrieval pools, the executor emits a fixed eight-slot trajectory with step indices, role ids, visible contexts, structured actions, next states, and parse/no-op flags. The eight slots are

Step	Role-stage	Step	Role-stage
1	ORCH.plan	5	RET.retrieve_follow_up
2	RET.retrieve_a	6	SYN.sub_answer_a
3	RET.retrieve_b	7	SYN.sub_answer_b
4	ORCH.follow_up	8	SYN.final_answer

Thus the role-step mapping is ORCH : $\{1, 4\}$, RET : $\{2, 3, 5\}$, and SYN : $\{6, 7, 8\}$. The interface is a maximum-slot execution schema rather than a claim that every question requires eight active reasoning operations. It exposes planning, evidence selection, follow-up control, intermediate synthesis, and final synthesis as observable decisions for role-level credit assignment. Dataset-specific case construction and retrieval-pool construction are described separately in Appendix D.1.

All role-based methods use the same trajectory collector, role prompts, structured-output parser, candidate-document interface, and evaluation code. This includes the frozen no-update executor and training variants such as joint-role update, random-role update, attribution-weighted update, AUDIT REFERENCE update, and RISE variants. Their difference lies only in the update controller: which role receives gradients, which objective is applied, and whether uncertain trajectories are frozen. MAIN-RAG is treated as an external strong RAG baseline: it is evaluated on the same QA examples and retrieval resources where applicable, but it does not instantiate the tri-role trajectory interface and is not used for role-attribution diagnostics.

Role prompts share a common structured-output contract across training methods. Each prompt specifies the role identity, stage description, output rules, example context/action format, and task context JSON. The required action schemas are:

Stage	Required action schema
`orch.plan`	`{sub_questions: [str], retrieval_queries: [str], execution_order: [int]}`
`orch.follow_up`	`{need_more_retrieval: bool, additional_query: str/null, reason: str}`
`ret.retrieve`	`{selected_titles: [str]}`
`syn.sub_answer`	`{sub_answer: str}`
`syn.final_answer`	`{final_answer: str}`

The same schema constraints and parsing rules are used before and after training and across all role-based baselines. Therefore, answer metrics and parse-failure diagnostics are measured under the same output contract.

No-op steps preserve alignment across trajectories while allowing the number of effective decisions to vary by example. In particular, if ORCH.follow_up returns `need_more_retrieval=false`, step 5 is still logged as RET.retrieve_follow_up, but with `selected_titles=[]`, `skipped=true`, empty retrieved documents, `is_noop=true`, and `parse_failure=false`. For attribution and update construction, no-op steps are treated as structural placeholders rather than model decisions. They receive zero judge evidence, are excluded from trajectory-local normalization, are not eligible for critical-step selection, and are not used in role-level evidence counts, replay positives, repair-pair construction, or update target selection.

Parse failure (PF) tracks malformed role-stage outputs that cannot be parsed or coerced into the required JSON/action schema. When this occurs, the executor uses a deterministic fallback action for logging and rollout continuation. Fallback events are counted in PF diagnostics, but fallback actions are not treated as faithful model decisions for repair. Any target step marked as a parse failure is

skipped during repair-pair construction, preventing contrastive repair from learning against heuristic fallback actions rather than actual model outputs.

C.2 Hyperparameters

Training and execution.

Parameter	Value	Notes
Role-step interface	3R-8S	ORCH:{1,4}, RET:{2,3,5}, SYN:{6,7,8}
Training rounds	3	Main conservative training setting
Group size	4	Grouped rollouts for trajectory-level advantage estimation
Optimizer	AdamW	Applied to the selected role parameters
Learning rate	1×10^{-5}	Matched across learning-based methods unless otherwise stated
Weight decay	0.0	Default selective-update setting
Gradient clipping	1.0	Applied before each role update
Max context tokens	2048	Matched across formal runs
LoRA rank / alpha / dropout	8 / 16 / 0.0	Role policy adapters
Generation temperature / top-p	0.8 / 0.95	Default role-policy decoding
Outcome buckets	0.7 / 0.3	Success if F1 $\geq$ 0.7, failure if F1 $\leq$ 0.3; intermediate outcomes are frozen

We report the resolved configuration used in the main experiments. When code defaults differ from the formal experimental setting, the values below correspond to the paper experiments.

Attribution, routing, and repair.

Parameter	Value	Notes
Judge prompt version	v1c	Formal LLM judge prompt
Judge score range	$[-10, 10]$	Signed integer JSON score
Score normalization	Trajectory-local z-score	No-op steps excluded
Critical-step budget	$\min(\lvert\mathcal{E}_\tau\rvert, \max(2, \mathrm{round}(0.4\lvert\mathcal{E}_\tau\rvert)))$	Computed over effective non-noop steps
Posterior-strength mode	Beta-LCB	Used for role-level attribution
Beta prior	$\alpha = 1.0,\ \beta = 1.0$	Role posterior parameters
LCB coefficient	$\lambda = 1.0$	Used in Eq. (6)
Abstention rule	Top-2 margin	Difference between the top two role scores
Margin threshold	$\delta = 0.03$	Main experimental setting
Neutral outcomes	Freeze all	No role update is applied
Zero reward variance	Skip reinforcement	Does not disable the repair branch
Repair target mode	Refined target	Route-selected-only used as ablation
Repair objective	Contrastive repair	Used in Eq. (12)
Repair temperature η	0.1	Temperature in the repair loss
Local repair threshold	$\delta_{\mathrm{neg}} = 0.5$	Additional filter on negative critical steps for local repair
Replay buffer size	1000 per role	Stores successful non-noop steps for matched positives
Minimum context similarity	0.15	Replay positive filter

D Experimental Setup

D.1 Benchmark Construction and Retrieval Pools

We construct all QA benchmarks from the original HotpotQA [33], 2WikiMultihopQA [10], and MuSiQue [25] examples by converting each raw instance into a shared canonical record. Each record stores the question, gold answer, dataset label, question type, normalized context documents, supporting-title metadata, primary supporting titles, aligned oracle document ids/titles, and auxiliary metadata for diagnostics. This conversion standardizes the input format across datasets but does not change the underlying QA task: the retained examples still come from the original benchmark

distributions, and filtering is limited to cases where the required supporting documents cannot be aligned to the canonical corpus. The oracle fields are retained for scoring, audit construction, and controlled diagnostic protocols, but the rollout executor receives only the fields allowed by the selected evaluation profile.

For each dataset, we build a canonical document corpus from the normalized context documents of the retained examples. Document titles are normalized using Unicode canonicalization, underscore replacement, whitespace normalization, and case folding. When multiple documents share the same normalized title, we keep the longest text version and assign a unique document id. We then align each example's first two primary supporting titles to this corpus to obtain oracle document ids and titles. Examples with fewer than two aligned primary supporting documents are removed, which ensures that controlled diagnostics and oracle-reference analyses are well-defined while keeping the constructed benchmark close to the original data source. Evaluation subset sizes and corpus statistics are reported in Appendix D.2.

Splits are generated with a fixed random seed and stratified by the dataset-provided question type when available. This preserves the relative mixture of bridge, comparison, and dataset-specific reasoning types across training, calibration, and evaluation subsets. The preprocessing code supports fixed development and formal splits, including training, calibration, sanity-check, development, and test partitions. All methods within a given experiment use the same split files, canonical records, and retrieval corpora, so performance differences are not caused by different example selection or evidence resources.

Retrieval pools are constructed from a BM25 index over the canonical corpus. For each retrieval query generated during a rollout, the retriever receives a candidate pool of 50 documents. The two initial retrieval steps use the two planner-generated queries, while the optional follow-up retrieval step uses the follow-up query only when ORCH requests additional retrieval. Under the main natural visible-only protocol, oracle documents are not forcibly inserted into the candidate pool; they appear only if retrieved by BM25 for the generated query. In oracle-injected diagnostic protocols, missing oracle documents may be inserted by replacing tail BM25 candidates, yielding a pool with up to two oracle documents and BM25 hard negatives. These oracle-injected settings are used only for diagnostics or upper-bound analyses and are reported separately from the natural visible-only main evaluation.

D.2 Dataset Details

We evaluate on controlled aligned subsets of HotpotQA, 2WikiMultihopQA, and MuSiQue. All datasets are converted into the canonical record format described in Appendix D.1, but they differ in raw annotation structure. HotpotQA provides titled context paragraphs, bridge/comparison type labels, and supporting-fact annotations; we extract supporting titles from the supporting facts, while leaving the decomposition field empty because HotpotQA does not provide explicit decomposition labels. 2WikiMultihopQA provides titled context documents, supporting facts, answer fields, and type labels; when evidence triples are available, we serialize them into lightweight decomposition metadata. MuSiQue provides paragraph-level records with supporting flags and question decompositions; we group paragraphs by title to form context documents and infer the hop/type label from the example id or decomposition length.

Across datasets, each retained example must align at least two primary supporting titles to the canonical corpus. This requirement ensures that every retained case can instantiate the same two-primary-evidence interface used by the tri-role executor and controlled diagnostics. For MuSiQue, which may contain more than two annotated reasoning steps, the first two unique supporting titles define the primary evidence slots, while the full decomposition and supporting metadata are retained for analysis. Examples that fail the support-document alignment check are excluded before subset construction. The aligned subsets are used for controlled method comparison and are not intended as public leaderboard submissions.

Main QA results use the 1000-example aligned evaluation subset for each dataset. Smaller subsets are used only for calibration, mechanism audits, and qualitative analysis. All subsets are selected deterministically after alignment: we sort aligned examples using a SHA1 key computed from the dataset name, record id, and seed, then take the first N examples. We use seeds 42 and 43 and report the average across the two deterministic subsets. No model prediction, retrieval success, judge score, or RISE routing signal is used for subset selection. The main natural-track evaluation does

not inject oracle documents into retrieval pools; oracle documents appear only if retrieved by BM25 for the generated query. Thus, subset construction fixes the experimental interface without selecting examples based on model behavior.

Dataset subset	Main eval N	Corpus docs	Annotation structure
HotpotQA aligned dev	1000	66568	bridge/comparison questions with supporting-fact titles
2Wiki aligned dev	1000	54943	compositional, comparison, inference, and bridge-comparison questions
MuSiQue answerable aligned dev	1000	17629	2-hop and higher-hop questions with paragraph-level supporting flags

D.3 Baseline Adaptation Details

We separate baselines into role-based training baselines and external QA/RAG baselines. Role-based baselines use the same tri-role/eight-step executor as RISE, including the same trajectory collector, role policies, structured-output parser, candidate-document interface, reward computation, and post-training evaluation. They differ only in how rollout trajectories are converted into parameter updates. External QA/RAG baselines are evaluated on the same examples, corpus, validation profile, and answer metrics, but they are not used for role-attribution diagnostics.

Baseline	Executor	Update rule	RISE attribution
No-Update	Shared tri-role	None	No
GRPO	Shared tri-role	All-role outcome update	No
RLOO	Shared tri-role	All-role leave-one-out update	No
GiGPO	Shared tri-role	All-role macro/step update	No
iStar	Shared tri-role	Joint implicit-step-reward update	No
Joint-role	Shared tri-role	All eligible roles	Partial
Random-role	Shared tri-role	Deterministic random role	Partial
Symptom-role	Shared tri-role	Visible symptom role	Partial
No-refinement	Shared tri-role	Raw routed role	Partial
Audit Reference	Shared tri-role	Offline reference role	Reference diagnostic
MAIN-RAG	Evaluation wrapper	None	No

Role-based training baselines. NO-UPDATE evaluates the initial role policies without any training. Outcome-only baselines train from final-answer rewards without role bottleneck inference. GRPO uses group-normalized final rewards, RLOO uses a leave-one-out reward baseline, and the GiGPO adapter adds structural step-level proxy advantages. These baselines update all eligible role policies with non-noop actions and do not use RISE's LLM-judge attribution, posterior-strength routing, responsibility refinement, or contrastive repair. The iStar baseline is adapted as a joint-update step-reward baseline: it uses the same QA executor and grouped rollout protocol, but replaces RISE's explicit bottleneck routing with an implicit PRM step reward.

Mechanism ablations. Mechanism ablations use the same selective-training runner, rollout interface, judge evidence, optimizer, and decoding configuration. Each variant changes the component under study. Joint-role update applies updates to all eligible roles rather than selecting a single target. Random-role update replaces the inferred target with a deterministic random role. Symptom-role update repairs the visible symptom role rather than the responsibility-refined role. The naive-attribution variant replaces the Beta-LCB posterior-strength score with an uncalibrated critical-step ratio. The no-abstention variant uses the top-1 role without the top-2 margin gate, and the no-refinement variant repairs the raw routed role directly. AUDIT REFERENCE replaces the inferred update target with an offline responsibility reference. This reference is available only to the diagnostic baseline, not to RISE during training, and does not constitute uniquely identifiable causal ground truth.

External QA/RAG baseline. MAIN-RAG is treated as an external strong RAG baseline rather than a role-evolution baseline. We wrap it into the common evaluation harness so that it uses the same QA examples, corpus, validation profile, and answer metrics. Internally, MAIN-RAG follows a predictor/judge/final-predictor procedure: candidate documents are scored with a document-level

predictor and support judge, the highest-scoring evidence is retained, and a final predictor generates the answer from the retained documents. No parameter update is applied. Since MAIN-RAG does not perform role-level bottleneck inference or selective updates, it is excluded from routing and responsibility-attribution diagnostics.

In the table, "Partial" means that the variant uses the same evidence/logging machinery as RISE but replaces one of the target-selection components. For role-based baselines, fairness is enforced at the executor level: they share the same trajectory schema, candidate-document pools, prompts, parser, reward computation, and evaluation code. For external baselines such as MAIN-RAG, fairness is enforced at the benchmark level: they use the same examples, corpus, retrieval resources where applicable, validation profile, and answer metrics, but are not included in role-attribution comparisons.

D.4 Training and Compute Details

Method family	Train rollouts	RISE judge/router	Roles/update	Training updates
No-Update	0	No	0	None
RISE	$N \times R \times G$	Yes	≤ 1	Abstention-dependent
GRPO/RLOO/GiGPO	$N \times R \times G$	No	All eligible	Outcome-dependent
iStar-style	$N \times R \times G$	No	All eligible	Step-reward-dependent
MAIN-RAG	0	No	–	None

This subsection reports the training protocol, compute accounting, and evaluation cadence used in the main experiments. Hyperparameter values are listed in Appendix C.2; here we focus on how training budgets are matched across methods and how checkpoints are evaluated.

For the main natural-track experiments, all role-based training methods use the same policy rollout backend, validation profile, training cases, and grouped rollout budget. Given a training subset of size N, each method samples $N \times R \times G$ training trajectories, where $R = 3$ training rounds per case and $G = 4$ rollouts per group in the main configuration. The post-training evaluation uses one rollout per evaluation example unless otherwise stated. Main tables report the final post-training checkpoint under the same natural visible-only evaluation protocol.

We match role-based training methods by rollout budget rather than by the number of active parameter updates. This is conservative for selective evolution: RISE may abstain on uncertain trajectories and updates at most one selected role for a routed trajectory, whereas joint-update baselines may apply gradients to all eligible roles from the same rollout batch. For Table 2, baseline diagnostic counts from smaller runs are linearly normalized to the common 720-trajectory scale. RISE also adds attribution-time judge calls during training; these calls are part of the method cost and are logged separately. Evaluation does not use the RISE judge.

For role-based methods, the orchestrator, retriever, and synthesizer are initialized from the same pretrained backbone but maintain separate trainable role parameters/adapters and optimizer states. During a selective update, gradients are enabled only for the selected role policy; all other role parameters are masked. Repair losses use frozen reference policies initialized from the starting policies, except in the no-anchor ablation. Outcome-only baselines use the same grouped rollout budget and executor but do not call the LLM judge, bottleneck router, or repair-pair constructor.

MAIN-RAG performs no training in our comparison and is reported as an evaluation-only external RAG baseline. It is evaluated on the same examples and retrieval resources, but its inference-time predictor/judge/final-predictor calls are not counted as role-evolution training compute. Thus, role-based comparisons are compute-accounted at the rollout/update level, while MAIN-RAG is interpreted as a training-free external QA reference.

Compute resources. All experiments were run on a local GPU server equipped with two NVIDIA GeForce RTX 4090 GPUs and five NVIDIA A100-SXM4-80GB GPUs. The main role-based training runs used A100-SXM4-80GB GPUs unless otherwise stated; RTX 4090 GPUs were used only for preprocessing, debugging, and validation runs.

D.5 Evaluation Metrics and Parsing

We compute answer quality from the final answer only. F1 and exact match (EM) use HotpotQA-style normalization: lowercasing, article removal, punctuation removal, and whitespace normalization.

EM is the normalized string match, and F1 is token-overlap F1 between the predicted final answer and the gold answer. The trajectory reward is final-answer F1. For training buckets, a trajectory is labeled success if F1 ≥ 0.7, failure if F1 ≤ 0.3, and neutral otherwise. Neutral trajectories are frozen and do not receive role updates.

Parse failure (PF) measures structured-output stability. A step is marked as a parse failure when the generated completion cannot be parsed or coerced into the required role-stage JSON schema, causing the executor to use a deterministic fallback action. Unless otherwise stated, PF is reported as the mean number of parse-fallback steps per rollout. Parse-fallback actions are counted in PF diagnostics but are not used as valid negative actions for repair-pair construction.

We report update diagnostics separately from answer quality. Upd is the fraction of training trajectories with a non-empty update target. Active counts gradient-producing role-level loss events. Event-Int. is $\mathrm{Active}/(\mathrm{Traj.} \times |\mathcal{R}|)$, and Rel. Upd. is the event count relative to RISE on the same scale. A routed RISE trajectory may contribute multiple events, all assigned to its selected role. For responsibility audits, active precision measures correctness among executed updates, update coverage measures how often a method updates, off-target rate measures wrong-role updates, and target accuracy counts abstentions as missed targets.

Additional logs record selected roles, routing margins, freeze/reinforce/repair counts, refinement-target changes, repair-pair counts, and parse failures by role and stage. Role diagnostic metrics use offline responsibility references, which are unavailable to the executor and RISE's routing procedure.

D.6 Audit Protocols

Audit	Input	Representative metrics	Supported claim
Judge validity	Step evidence packets	Severity alignment, critical-step agreement, role agreement	Judge evidence is usable but noisy
Responsibility target	Failed trajectories	Active precision, off-target rate, pullback agreement	Visible symptom and bottleneck can differ
Margin gate	Logged role margins	Freeze rate, active-update rate, selected-role distribution	Abstention controls uncertainty
Repair pairs	Repair jobs	Pair source, skip counts, positive validity	Repair supervision is grounded
Update selectivity	Training logs	Active roles, freeze/repair counts, non-target proxy	Selective updates reduce contamination risk

We use mechanism audits to validate components that are not fully captured by final-answer F1. These audits are separate from end-task QA evaluation: QA metrics measure answer quality, while audits measure whether judge evidence, margin abstention, responsibility refinement, repair-pair construction, and selective updates behave as intended. The audits support mechanism claims rather than leaderboard claims.

For Figures 3 and 4, two annotators independently labeled step severity and responsibility targets on 100 frozen HotpotQA failure trajectories covering 700 non-noop steps, without access to judge scores or routing decisions. These labels were used only for offline audits. We report whether judge scores are aligned with failure severity, whether critical steps are ranked near the top, and whether negative evidence concentrates on the expected role. These audits validate the judge as a useful noisy evidence source for routing, not as a calibrated causal oracle. Second, responsibility-target audits focus on failed trajectories where the visible symptom role may differ from the update-worthy bottleneck. Each audited case records the symptom-view role, responsibility-view role, supported cause roles, first semantic-divergence role, propagation-chain evidence, and final refinement target. Figure 4 compares RISE with symptom updating, mean attribution, RISE without abstention, and the AUDIT REFERENCE. This audit measures whether support-constrained refinement pulls repair targets upstream only when visible trajectory evidence supports that responsibility relation. Third, margin-gate audits replay the top-2 margin decision from logged role scores under candidate thresholds. Logs record role margins, freeze decisions, update branches, and selected roles. The main setting uses $\delta = 0.03$. Offline sweeps are used to verify that the threshold controls routing pressure rather than collapsing into always-update or always-freeze behavior. Fourth, repair-pair audits inspect the construction of contrastive repair pairs. Repair logs record target steps, constructed pairs, positive sources, and

skip reasons. This audit checks that repair training uses grounded, non-identical positive/negative pairs rather than parser fallback actions or unsupported positives. Finally, update-selectivity audits summarize active update counts, selected-role distributions, freeze/reinforce/repair counts, symptom-responsibility disagreement, and non-target update proxies. These logs support the claim that RISE reduces credit-contamination risk relative to joint-update baselines. We treat these audits as mechanism evidence; stronger causal claims about collateral damage require paired counterfactual or compute-matched runs and are reported separately when available. Audit labels are used only for offline analysis and the explicitly labeled AUDIT REFERENCE baseline, not for RISE training.

Case A: Visible symptom is misleading; RISE redirects the update upstream

Question. Who opened Gunnersbury Park and signed the Munich Agreement?
Gold answer. Neville Chamberlain.

Observed failure	The original trajectory returns *"No information available"* (F1 = 0.0), although the gold answer is available from the evidence.
Visible symptom	The final incorrect answer is produced by the synthesizer, so the local symptom appears at SYN.
RISE evidence	The responsibility view identifies an upstream supported chain ORCH → RET → SYN, with `supported_cause_roles = [Orch]` and `responsibility_view_reason = upstream_supported`.
RISE target	Although the router first selects SYN, the refinement module redirects the update target to ORCH.
Audited target	ORCH.
Post-update result	The updated trajectory answers *"Arthur Neville Chamberlain"* (F1 = 0.8).

Interpretation. This case illustrates the central mechanism of RISE: the role that visibly emits the wrong answer is not necessarily the role that should receive the update. RISE uses supported propagation evidence to avoid a local synthesizer-only repair and redirects credit to the upstream bottleneck.

Figure 5: Case study of symptom–responsibility separation. The visible error is emitted by SYN, but the supported responsibility target is ORCH.

E Additional Experimental Results

E.1 Parameter Sensitivity Analysis

Figure 6 shows how the margin threshold δ controls the trade- off between answer quality and update selectivity. When δ is too small, RISE updates aggressively, leading to high update rates and lower F1 because ambiguous trajectories are forced into training. As δ increases, update rate drops sharply and F1 improves, reaching its best value around $\delta = 0.03$. Further increasing δ makes the router overly conservative: fewer uncertain updates are applied, but useful repair opportunities are also filtered out, causing F1 to decline. The near-optimal region around $\delta \in [0.02, 0.04]$ indicates that RISE does not rely on a brittle threshold choice; it benefits from a moderate abstention margin that removes noisy updates while preserving enough supported training signals.

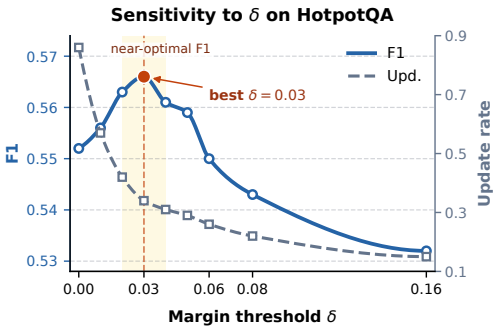


Figure 6: Sensitivity to the margin threshold δ on HotpotQA using Qwen2.5-7B-Instruct.

E.2 WebShop Compatibility

We include WebShop [34] as a supplementary compatibility check rather than a primary benchmark. Unlike multi-hop QA, WebShop emphasizes interactive navigation, valid-action selection, and sparse rewards, while RISE targets role-level update selection under noisy trajectory evidence. We therefore use WebShop only to test whether the role-partitioned execution and selective-update interface can be instantiated beyond QA.

Case B: Evidence is insufficiently separated; RISE abstains

Question. What was the origin of the player that the Bulls traded with the 76ers during the 1999–2000 midseason?

Gold answer. Croatian.

Observed failure	The trajectory outputs an incorrect answer such as *"Philippines"* or an uninformative answer, yielding F1 = 0.0.
Visible symptom	The evidence points to RET as the strongest failure-side role, while SYN remains a close competing explanation.
RISE evidence	The top two posterior-strength scores are close, and their margin falls below the main abstention threshold $\delta = 0.03$. The trajectory therefore lacks a sufficiently separated responsibility target under visible evidence.
RISE target	RISE predicts $\varnothing$ and applies FREEZE ALL ROLES.
Audited target	Ambiguous under the available trajectory evidence.
Post-update result	No update is applied; the case remains unresolved.

Interpretation. This case shows that RISE does not force every failed trajectory into a role update. When the posterior margin falls below the main threshold, the method abstains, treating "no update" as a first-class decision for contamination control.

Figure 7: Case study of uncertainty-aware abstention. When competing role explanations are insufficiently separated, RISE freezes all roles rather than applying an ambiguous update.

Case C: Upstream evidence is identified, but answer recovery remains incomplete

Question. In the 2012–13 FC Sion they brought in Gennaro Gattuso and a player who now plays for what Scottish team?

Gold answer. Scottish Premiership club Hearts.

Observed failure	The standalone RISE trajectory outputs an uninformative answer, yielding F1 = 0.0 and EM = 0.0.
Visible symptom	The failure is associated with the retrieval stage, so the visible symptom is at RET.
RISE evidence	RISE identifies an upstream-supported responsibility signal with `responsibility_view_reason = upstream_supported` and a supported chain ORCH $\rightarrow$ RET.
RISE target	RISE assigns the responsibility target to ORCH.
Audited target	The trace supports an upstream planning/query bottleneck, but the available evidence is not sufficient to guarantee answer recovery.
Post-update result	The standalone recovery path remains incorrect.

Interpretation. This limitation case separates responsibility-target inference from end-task answer recovery. RISE can expose an auditable upstream responsibility signal, but correcting the final answer also depends on the executor's ability to use the repaired trajectory and produce the complete answer form.

Figure 8: Appendix limitation case. Unlike the main case study, this example is not meant to show successful recovery. It clarifies the boundary of the qualitative evidence: RISE provides an auditable role-target inference mechanism, while per-case QA recovery can still fail when the downstream executor or answer surface form remains insufficient.

Under a local small-preview protocol with 1,000 products and 200 deterministic held-out goals, the RISE-style interface completes all episodes without structured-output parse failures. This result shows that the execution, logging, and update interface can transfer to an interactive setting, but it is not a controlled performance comparison against WebShop-specialized RL methods.

E.3 Case Study

We include three qualitative cases to expose the decision chain from trajectory evidence to role-level update targets. Cases A and B illustrate the two intended behaviors of RISE. Case A demonstrates responsibility redirection: the visible error appears at the synthesizer, but the evidence supports an upstream propagation chain, so RISE redirects the update target to ORCH. Case B demonstrates

uncertainty-aware abstention: when competing role explanations are not sufficiently separated, RISE freezes all roles rather than forcing a low-confidence update. Case C illustrates a boundary case: even when RISE identifies a plausible upstream responsibility target, the final answer may still fail if the executor cannot retrieve or verbalize the complete answer. These cases should therefore be read as mechanism audits, not performance evidence: they show that RISE decisions are inspectable, while per-case QA recovery still depends on downstream execution and answer generation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The summarized contributions are provided in the abstract and introduction.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in Appendix A, including the pipeline and dependency assumptions, reliance on the LLM judge and support predicates, the operational nature of the Audit Reference, and training-time overhead. Appendix E.3 further includes a case where RISE identifies a plausible upstream target but the final answer remains unrecovered.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper includes a contamination-accounting proposition in §3.3, with the bounded-gradient assumption and non-abstaining target error rate stated in the proposition. A proof sketch is provided in the main text, and the fuller derivation is referenced in Appendix B.5.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the method in §3, the experimental setup in §4.1, and hyperparameters in Appendix C.2. The paper also reports dataset construction, baseline adaptation, evaluation metrics, and mechanism-specific configurations in Appendix D.1, Appendix D.3, and Appendix D.5; code is publicly available at the URL provided in the abstract.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide a public code repository at the URL in the abstract, with instructions for reproducing the main experiments. The datasets used in the paper are public benchmarks, and data construction and retrieval-pool details are described in Appendix D.1.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We describe the experimental setup, metrics, models, and baselines in §4.1. Full hyperparameters and implementation details are provided in Appendix C.2.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: The results are accompanied by error bars, confidence intervals, or statistical significance tests. Main experimental results are averaged over 2 repeated runs, and the reported tables use these averaged values.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Main results are averaged over two deterministic evaluation subsets (Appendix D.2), and RISE achieves consistent improvements across all model–dataset conditions.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cited the original paper that produced the code package or dataset.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are central to the proposed method and evaluation: role policies are LLM agents, and an LLM judge provides step-level evidence for attribution.