MY FAULT: SELF-DIAGNOSIS AS CREDIT ASSIGNMENT IN SELF-EVOLVING AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic reinforcement learning (RL) has emerged as a powerful approach for training large language model agents on multi-step tasks, yet reliance on terminal outcome rewards creates two credit-assignment problems, particularly in long-horizon tasks. First, same-outcome rollout groups provide no learning signal from terminal rewards. Second, terminal rewards provide only trajectory-wide feedback, making it difficult to identify which decisions caused a failure. Recent work supplements terminal rewards with finer-grained information from trajectory analysis, such as natural-language reflections on intermediate decisions and errors. However, natural-language diagnoses are difficult to use directly for credit assignment: their error claims may be unreliable, and they do not quantify how much each error should affect learning. We propose Self-Diagnosis-guided Terminal Credit Redistribution (FAULT), which turns diagnosed errors into explicit step-level credit anchored by terminal outcomes. FAULT checks diagnostic evidence and learns relative error costs from task outcomes. During training, the policy and self-diagnoser co-evolve, while error costs are updated online from recent outcomes. On ALFWorld, FAULT recovers learning signals from same-outcome groups, reaching 95% signal coverage versus 41% for GRPO and 72% for GiGPO, while better localizing credit to specific error steps. Across two model scales, FAULT delivers strong improvements on the long-horizon ALFWorld and WebShop tasks while remaining competitive on short-horizon Search-based QA.

1 INTRODUCTION

Large language model agents solve increasingly complex tasks through sequences of decisions: they reason, interact with external environments, observe feedback, and adapt their subsequent actions (Yao et al., 2023). Agentic reinforcement learning improves such behavior by optimizing these interaction trajectories, typically using rewards based on the final task outcome (Zhang et al., 2026a). Terminal outcome supervision is widely used (Jin et al., 2025; Qian et al., 2025; Feng et al., 2026; Wei et al., 2025; Feng et al., 2025) because final outcomes are often easier to verify than intermediate decisions, but it creates a fundamental credit-assignment problem: a single reward must supervise many intermediate decisions that may have contributed very differently to the final result. As trajectories grow longer, the reward becomes increasingly distant from the decisions it is meant to improve.

This limitation is especially pronounced in group-relative RL methods such as GRPO (Shao et al., 2024; Guo et al., 2025), which normalize terminal rewards across multiple rollouts of the same task, and broadcast each rollout-level advantage to all of its steps. This produces two distinct credit-assignment limitations. First, when all rollouts in a group receive the same terminal outcome, their normalized advantages vanish, so the group provides no outcome-driven learning signal, even if the trajectories differ substantially in their intermediate decisions. In our preliminary ALFWorld (Shridhar et al., 2021) runs, such same-outcome groups account for 59–64% of training groups. Second, even when a group contains both successes and failures, the resulting trajectory-level advantage does not identify which intermediate decisions were responsible for the outcome: all steps within a rollout receive the same signal. As trajectories become longer, this makes it increasingly difficult to distinguish decisive errors from otherwise reasonable actions.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

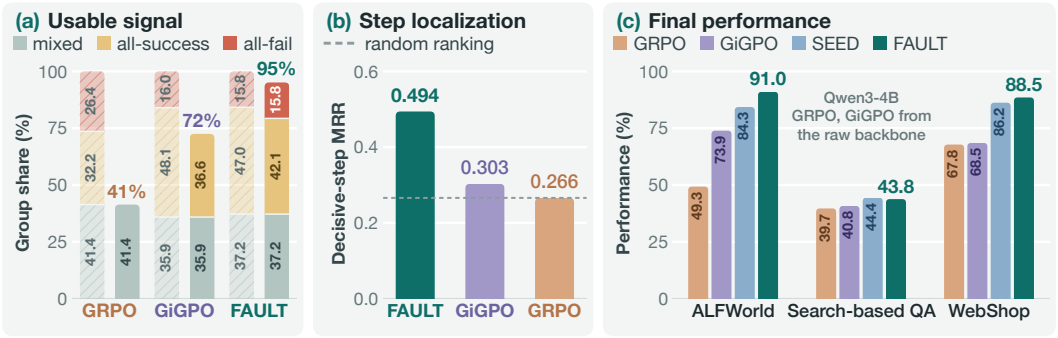


Figure 1: **Signal coverage, step localization and final performance. (a) Signal coverage.** On ALFWorld with shared diagnosis-SFT initialization, hatched bars show the group mix and solid bars the groups with usable signal. FAULT uses 95% of groups, including every all-fail group and most all-success groups. GRPO uses only mixed groups (41%); GiGPO also uses some all-success groups but no all-fail groups (72%). Thus, FAULT can learn from differences in intermediate behavior even when terminal rewards are identical, making both successful and failed rollouts useful beyond their final outcomes. **(b) Localization.** Mean reciprocal rank of the decisive error step a blind LLM judge marks in each failed trajectory, with steps ranked by each method's penalty. FAULT (0.494) nearly doubles random (0.266) while GiGPO (0.303, same-configuration rerun) stays close to it, so FAULT's step-level credit is far more accurate. The gains highlight the value of diagnostic evidence in identifying which decisions within a failed trajectory need correction and directing step-level feedback to those decisions. **(c) Outcome.** Qwen3-4B results from Table 1. FAULT leads in ALFWorld success rate and WebShop task score on these longer-horizon tasks, remains competitive with SEED on shorter-horizon Search-based QA, and outperforms GRPO and GiGPO initialized from the raw backbone on all three.

A natural way to address these limitations is to introduce process-level signals that provide finer-grained feedback than terminal rewards (Lightman et al., 2024; Wang et al., 2024; Setlur et al., 2025). One line of work constructs numerical signals for intermediate steps, using learned reward models (Xi et al., 2026), Monte Carlo estimates (Choudhury, 2025), hindsight-based scores (Ma et al., 2026; Yao et al., 2026), or comparisons across related states (Feng et al., 2025). However, many such signals are still derived from terminal outcomes and therefore lose discriminative power when outcomes are identical. A complementary line of work analyzes trajectories in natural language, using reflection (Shinn et al., 2023; Zhao et al., 2024; Zheng et al., 2026; Wu et al., 2026), hindsight (Bensal et al., 2025), or error diagnosis to identify what went wrong and where (Yuan et al., 2025; Zhang et al., 2026b). These analyses provide richer information about intermediate decisions, but remain descriptive: they are not directly expressed as quantitative step-level credit, and their error claims require heavy verification before being used for learning.

Turning process information into effective credit assignment requires solving two key challenges. First, the process signal must be trustworthy: an error diagnosis can misidentify either what went wrong or where it occurred, causing credit to be assigned to the wrong step. This is a practical concern in agent traces, where even strong models can struggle to jointly identify and localize errors. Second, identifying an error does not determine how much that error should matter for learning. Different error types may have very different effects on task success, and multiple errors within the same trajectory should not deserve equal weight. Thus, useful step-level credit requires both verifiable localization of errors and outcome-grounded calibration of their relative importance.

We propose Self-Diagnosis-guided Terminal Credit Redistribution (FAULT), which turns self-diagnosed errors into explicit step-level credit, while retaining terminal outcomes as the source of supervision. The key idea is to separate error localization from error valuation: self-diagnosis identifies what went wrong and where, while terminal outcomes determine how costly different error types are. FAULT first performs **diagnosing**, which converts trajectory analyses into structured error records and verifies each diagnosis against evidence from the cited step. It then performs **pricing**, which learns relative error costs from observed task outcomes, and uses these costs to finally **redistribute** a bounded penalty budget across the diagnosed steps. Because the learned error costs can be applied even when all rollouts receive the same terminal outcome, FAULT can distinguish

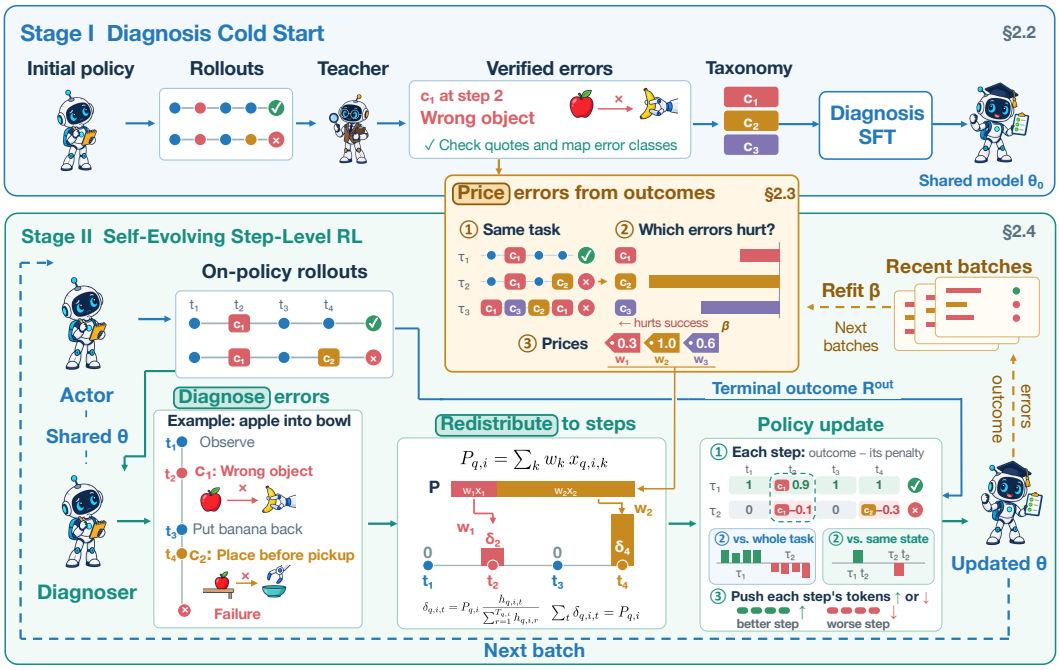


Figure 2: **Overview of FAULT.** *Stage I* initializes a shared actor and diagnoser by SFT on teacher diagnoses and learns error prices from terminal outcomes. In *Stage II*, we **diagnose** errors in on-policy rollouts, **price** them from terminal outcomes, and **redistribute** a bounded penalty budget across diagnosed steps for policy updates. The actor, diagnoser and prices co-evolve through shared model updates and online price refits.

trajectories that would otherwise receive identical credit, and can concentrate learning signal on the steps most in need of correction.

We summarize how FAULT addresses the two credit-assignment problems and how these improvements translate into downstream performance in Figure 1. First, FAULT substantially increases training signal coverage: on ALFWorld, 95% of rollout groups provide non-negligible credit contrast, compared with 41% for GRPO and 72% for GiGPO. Second, FAULT better localizes credit to the decisions responsible for failure, achieving a decisive-step MRR of 0.494, compared with 0.303 for GiGPO and 0.266 for random ranking. Finally, across two model scales, FAULT achieves the strongest improvements on the longer-horizon ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022) tasks while remaining competitive on the much shorter Search-based QA tasks. These results demonstrate the effectiveness of our finer-grained credit assignment becoming more important as trajectories grow longer.

2 Method

We present Self-Diagnosis-guided Terminal Credit Redistribution (FAULT), a framework for explicit step-level credit assignment in agentic RL. A cold start initializes a shared actor and diagnoser through diagnosis SFT and learns initial error prices; RL then locates errors through structured self-diagnosis, fits their prices from terminal outcomes, and redistributes a bounded penalty across the diagnosed steps.

2.1 Problem Formulation

An LLM policy π_θ solves task $q \sim \mathcal{D}$ through repeated interaction with an external world. At step t, it generates response $a_{q,i,t} \sim \pi_\theta(\cdot \mid o_{q,i,t})$ from context $o_{q,i,t}$ containing the task, interaction history and feedback so far; the world executes the response and returns feedback, appended to the context. Each response-feedback pair forms an *active interaction step*. Task q's i-th rollout is

$\tau_{q,i} = (o_{q,i,1}, a_{q,i,1}, \ldots, o_{q,i,T_{q,i}}, a_{q,i,T_{q,i}})$, with $T_{q,i}$ such steps. Following group-based RL, we sample G rollouts per task from frozen behaviour policy π_{old}.

Each trajectory receives terminal reward $R^{\text{out}}_{q,i}$ only at termination, giving the outcome objective

$$J_{\text{out}}(\theta) = \mathbb{E}_{q\sim\mathcal{D}}\,\mathbb{E}_{\tau_{q,i}\sim\pi_\theta}\big[R^{\text{out}}_{q,i}\big]. \tag{1}$$

This reward is the only judgement guaranteed correct; we preserve it and redistribute only its credit across steps.

2.2 Cold Start: Verifiable Diagnosis and Diagnosis SFT

Before RL, a one-time cold start fixes the error taxonomy, teaches the policy to produce verifiable diagnoses of its completed trajectories, and estimates initial error prices. Pricing errors from terminal outcomes (§2.3) and penalizing diagnosed steps (§2.4) require diagnoses that are countable across trajectories and checkable against each one. Free-form reflections lack this structure, so we map error claims to fixed classes and check evidence from their cited steps.

Teacher diagnosis. Teacher model GPT-5.6 Sol (OpenAI, 2026) reads initial-policy trajectories and outcomes, recording each error's label, step and an exact quote from that step.

Verification. We check each report's step and quote against the trajectory. Accepted labels form a fixed taxonomy, from which we select K classes with enough examples for pricing: $\mathcal{C} = \{c_1, \ldots, c_K\}$ (Appendix B.1). Teacher and later policy diagnoses of $\tau_{q,i}$ are represented as entries (k, t, e): error class, step and quote. Verified entries of priced class k give count $n_{q,i,k}$ and per-step rate $x_{q,i,k} = n_{q,i,k}/T_{q,i}$; the rates form $\mathbf{x}_{q,i}$ (Appendix B.2).

Diagnosis SFT. For each cold-start example $(q, i) \in \mathcal{D}_{\text{sft}}$ with an accepted teacher diagnosis, we fine-tune π_θ to reproduce its verified text $\mathbf{d}_{q,i}$ given trajectory $\tau_{q,i}$ and success label $y_{q,i} \in \{0, 1\}$:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\,\mathbb{E}_{(q,i)\in\mathcal{D}_{\text{sft}}} \sum_{\ell=1}^{|\mathbf{d}_{q,i}|} \log \pi_\theta\big(d_{q,i,\ell} \mid \tau_{q,i}, y_{q,i}, \mathbf{d}_{q,i,<\ell}\big), \tag{2}$$

Only diagnosis tokens are supervised; prompts include the frozen taxonomy, trajectory and outcome (Appendix B.1). The checkpoint initializes the RL actor and diagnoser; the same verified diagnoses and outcomes initialize prices via the estimator in §2.3. No teacher is used thereafter.

2.3 Terminal-Anchored Error Pricing

Locating an error does not reveal its cost, so FAULT regresses terminal outcomes on verified error rates (§2.2) to estimate an outcome association coefficient β_k for each class. A more negative β_k links higher error rates to worse outcomes, holding other features fixed. We convert these coefficients to nonnegative relative error prices w_k (§2.4).

Tasks differ in difficulty, so we compare successful and failed rollouts within each task. We take the observed number of successes as given and use error rates to explain which rollouts succeed, without separately estimating each task's baseline success rate. For group q, let $\mathcal{Y}_q$ be the successes retained by the diagnosis gate (Appendix B.2), and $\mathcal{A}_q$ all candidate success sets Γ drawn from all retained trajectories with $|\Gamma| = |\mathcal{Y}_q|$. This gives the observed success-set probability and likelihood objective over informative groups $\mathcal{Q}_{\text{info}}$, containing both successes and failures:

$$\Pr_q(\boldsymbol{\beta}) = \frac{\exp\big(\sum_{i\in\mathcal{Y}_q} \boldsymbol{\beta}^\top \mathbf{x}_{q,i}\big)}{\sum_{\Gamma\in\mathcal{A}_q} \exp\big(\sum_{i\in\Gamma} \boldsymbol{\beta}^\top \mathbf{x}_{q,i}\big)}, \qquad \boldsymbol{\beta}^\star = \arg\min_{\boldsymbol{\beta}} \Big[-\sum_{q\in\mathcal{Q}_{\text{info}}} \log \Pr_q(\boldsymbol{\beta}) \Big]. \tag{3}$$

For binary outcomes, these coefficients are fitted by bounded quasi-Newton optimization with a small-sample correction and shrinkage toward a fixed anchor: initially on teacher-diagnosed trajectories (§2.2), then periodically on a sliding window of recent RL rollouts (derivation, estimator, acceptance test and update rule in Appendix B.3).

2.4 Self-Evolving RL with Step-Level Credit Redistribution

Verified errors can differ even when all rollouts in a group succeed or fail, and can provide step-specific process signals when priced. In each batch, a frozen policy generates and diagnoses a rollout

group per task. Using fixed error prices, we set each trajectory's bounded total penalty, redistribute it to diagnosed steps, and use each step's share to form group-relative advantages for policy learning.

Self-diagnosis of on-policy rollouts. After each rollout, the actor diagnoses its completed trajectory given the observed outcome, using the format of §2.2. Verified errors are aggregated into error-rate vector $\mathbf{x}_{q,i}$.

Bounded budget. We cap the total penalty below the success–failure reward gap for binary outcomes, so steps from successful trajectories retain higher episode scores than those from failures. Before this cap, fixed prices turn each trajectory's verified diagnosis into a score $S_{q,i} \in [0,1]$ and budget $P_{q,i}$:

$$P_{q,i} = \lambda_u S_{q,i} = \lambda_u \ \mathrm{clip}\Big(\sum_k w_k \, x_{q,i,k},\, 0,\, 1\Big), \quad w_k = \begin{cases} |\beta_k| / \max_{j:\,\beta_j<0} |\beta_j|, & \beta_k < 0, \\ 0, & \beta_k \geq 0. \end{cases} \tag{4}$$

Here $x_{q,i,k} = n_{q,i,k}/T_{q,i}$ is the verified error rate, and w_k is the nonnegative error price derived from β_k (§2.3). Using rates rather than counts avoids rewarding the RL actor for shorter trajectories solely because longer ones accumulate more diagnosed errors. λ_u sets batch u's penalty strength; Appendices B.3 and B.4 detail the price mapping, schedule and cap.

Conserved redistribution. The budget is split in proportion to step localization weights $h_{q,i,t}$,

$$\delta_{q,i,t} = P_{q,i} \, \frac{h_{q,i,t}}{\sum_{r=1}^{T_{q,i}} h_{q,i,r}}, \qquad h_{q,i,t} = \sum_{e \in \mathcal{V}_{q,i,t}} w_{k(e)}, \tag{5}$$

Here $\mathcal{V}_{q,i,t}$ contains verified errors cited at step t, and $k(e)$ is entry e's class. Their summed prices give the raw step penalty weight $h_{q,i,t}$; normalization to $P_{q,i}$ gives $\delta_{q,i,t}$, the penalty allocated to step t for the policy update (exact form in Appendix B.5).

Dual-channel advantage. Task-wide comparisons provide overall outcome guidance but use a baseline shared across different contexts. We therefore add comparisons within similar contexts to give each decision a more relevant reference. Following GiGPO (Feng et al., 2025), the episode channel compares each active step with Ω_q, all task-q step samples (j, r); the step channel uses the anchor-state group $\mathcal{H}(o_{q,i,t}) \subseteq \Omega_q$ of samples with the same or similar pre-response context (Appendix B.6). Both channels use the same step penalty; group centering cancels any penalty shared by all its samples. The episode channel centers terminal reward minus the step's share within the task group,

$$A^{\mathrm{ep}}_{q,i,t} = E_{q,i,t} - \frac{1}{|\Omega_q|} \sum_{(j,r)\in\Omega_q} E_{q,j,r}, \qquad E_{q,i,t} = R^{\mathrm{out}}_{q,i} - \delta_{q,i,t}, \tag{6}$$

and the step channel centers discounted return minus the step penalty within the anchor-state group,

$$A^{\mathrm{step}}_{q,i,t} = \widetilde{Z}_{q,i,t} - \frac{1}{|\mathcal{H}(o_{q,i,t})|} \sum_{(j,r)\in\mathcal{H}(o_{q,i,t})} \widetilde{Z}_{q,j,r}, \qquad \widetilde{Z}_{q,i,t} = \gamma^{\,T_{q,i}-t} R^{\mathrm{out}}_{q,i} - \delta_{q,i,t}, \tag{7}$$

Policy update. The joint step advantage $A_{q,i,t} = A^{\mathrm{ep}}_{q,i,t} + \omega A^{\mathrm{step}}_{q,i,t}$, with step-channel weight ω, is broadcast uniformly to step t's response tokens $\mathcal{U}_{q,i,t}$. The actor minimizes the clipped policy-gradient objective over all N response tokens in the batch:

$$\mathcal{L}_{\mathrm{actor}}(\theta) = -\frac{1}{N} \sum_{(q,i,t)} \sum_{\ell \in \mathcal{U}_{q,i,t}} \Big[\min\big(\varrho_\ell A_{q,i,t},\ \mathrm{clip}(\varrho_\ell, 1-\epsilon, 1+\epsilon)\, A_{q,i,t}\big) - \lambda_{\mathrm{KL}} \, D^{\mathrm{LV}}_\ell \Big], \tag{8}$$

where $\varrho_\ell = \pi_\theta(a_\ell \mid c_\ell)/\pi_{\mathrm{old}}(a_\ell \mid c_\ell)$ is token ℓ's importance ratio against the rollout policy given context c_ℓ, and D^{LV}_ℓ is a low-variance KL estimate to frozen reference policy π_{ref} at that token (Appendix B.6).

Co-evolution of diagnoser, prices and policy. The diagnoser shares the actor's weights, but its diagnosis tokens receive no policy gradient, preventing a direct RL shortcut of reporting fewer errors to avoid penalties. After each actor update, the batch's verified records $(q, \mathbf{x}_{q,i}, y_{q,i})$ enter a sliding window; with enough informative groups, prices are periodically refitted online via Eq. (3) with the regularization and smoothing of Appendix B.3.5. The updated model generates and diagnoses new rollouts, and the latest prices turn these diagnoses into step penalties for subsequent policy updates, closing the loop (Algorithm 1 in Appendix B.7).

3 EXPERIMENTS

3.1 EXPERIMENTAL SETTING

We evaluate FAULT on three benchmarks. Further details are in Appendix C.

Benchmarks. ALFWorld (Shridhar et al., 2021) spans six text-based household task families; we train on standard games and test on unseen tasks within 30 steps. In WebShop (Yao et al., 2022), the agent searches 1,000 products and buys one matching a natural-language request within 15 steps. Search-based QA follows Search-R1 (Jin et al., 2025): the agent searches for evidence to answer questions from NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022) and Bamboogle (Press et al., 2023).

Baselines. Vanilla is the backbone without task-specific training. GRPO (Shao et al., 2024) uses group-normalized outcome rewards and GiGPO (Feng et al., 2025) adds step credit from anchor-state comparisons; both start from the raw backbone. SEED (Wu et al., 2026) uses its own teacher-written hindsight-skill SFT data and distils skill-conditioned behaviour during RL. At 4B, GRPO (diag-SFT) and GiGPO (diag-SFT) start from our diagnosis-SFT checkpoint without diagnostic shaping.

Evaluation metrics. ALFWorld reports success by task family and over all 134 unseen tasks, averaging three greedy evaluations per task. WebShop reports mean normalized task score and strict purchase success over 256 episodes. Search-based QA reports exact match for each of seven datasets and the unweighted mean of their accuracies, not a pooled question-level average.

Implementation details. We use Qwen3-1.7B (thinking-capable) and Qwen3-4B-Instruct-2507 (non-thinking) (Yang et al., 2025; Qwen Team, 2025), with the same no-thinking evaluation template. For ALFWorld and WebShop, cold start samples eight rollouts for each of 180 tasks per checkpoint. Verified diagnoses yield the frozen taxonomy, diagnosis-SFT checkpoint and initial prices; the checkpoint initializes actor and diagnoser. RL runs 160 updates with 16 tasks (128 for Search-based QA), $G = 8$ rollouts and three self-diagnoses per trajectory; prices are refit every 10 updates from update 30.

3.2 MAIN RESULTS

Table 1 reports results for two backbones on three benchmarks.

Strong performance across benchmarks and scales. At both scales, FAULT has the highest ALFWorld success rate and WebShop task score. Across scales, its gains over raw-initialized GRPO/GiGPO span 17.1–41.7 points on ALFWorld, 8.7–21.0 in WebShop task score and 3.0–5.1 on the Search-based QA average. At 4B, it also beats their (diag-SFT) variants on ALFWorld and WebShop, showing gains beyond shared initialization. Margins over SEED, the strongest ALFWorld/WebShop baseline, are smaller: 6.7 and 3.0 points on ALFWorld and 2.3 and 0.7 in WebShop task score at 4B and 1.7B. On Search-based QA, FAULT ties SEED at 38.9 at 1.7B; at 4B, it scores 43.8, 0.6 points below SEED. SEED leads WebShop strict success by 2.7 and 3.9 points at 4B and 1.7B, likely because FAULT, unlike the baselines, trains on the graded score that also credits partial matches, and SEED's distilled skills suit this all-or-nothing metric.

Initialization and RL both contribute. The 4B (diag-SFT) rows distinguish cold-start and RL gains, though they are imperfect single-factor controls (Appendix C). Our diagnosis-SFT checkpoint raises GRPO/GiGPO from 49.3/73.9 to 71.6/80.6 on ALFWorld, 67.8/68.5 to 82.2/83.5 in WebShop score, and 39.7/40.8 to 45.7/46.5 on Search-based QA. GiGPO (diag-SFT) still trails SEED by 3.7/2.7 points on ALFWorld/WebShop. From the same checkpoint, FAULT beats GiGPO (diag-SFT) by 10.4 points and SEED by 6.7 on ALFWorld; its 88.5 WebShop score also exceeds SEED's, though with a different reward. On Search-based QA, however, FAULT trails these two (diag-SFT) controls by 1.9/2.7 points, so diagnosis-guided RL's additional gains concentrate in ALFWorld and WebShop.

Margins and episode length. The above GRPO/GiGPO margin ranges shrink from ALFWorld to WebShop to Search-based QA, matching their step-limit order (30, 15 and 4). The same order holds against GiGPO (diag-SFT), with 4B margins of 10.4, 5.0 and −2.7 points, and against SEED at both scales. This pattern is consistent with greater benefits from step-level credit on longer tasks: longer

Table 1: Performance on ALFWorld, Search-based QA, and WebShop. ALFWorld reports success rate (%) by task family and over all 134 unseen tasks (All); Search-based QA reports exact-match accuracy (%) by dataset and its unweighted average (Avg); WebShop reports mean task score and strict purchase success (%) over 256 episodes. Rows marked (diag-SFT) start RL from our Stage 1 cold-start model. Within each backbone, **bold** and underlined mark the highest and second-highest reported values per column (ties share formatting).

	ALFWorld							Search-based QA								WebShop	
Method	Pick	Look	Clean	Heat	Cool	Pick2	All	NQ	Triv	Pop	Hotp	2Wk	MuS	Bam	Avg	Score	Succ.
Qwen3-1.7B																	
Vanilla	8.3	22.2	6.5	0.0	0.0	0.0	6.0	29.7	48.2	35.0	23.7	19.6	6.4	17.6	25.7	51.0	2.3
GRPO	66.7	38.9	35.5	43.5	52.4	17.6	43.3	39.3	57.1	43.3	35.7	33.9	11.7	28.0	35.6	66.5	39.1
GiGPO	58.3	27.8	29.0	65.2	38.1	29.4	41.8	39.1	56.6	44.4	33.2	29.2	11.2	23.2	33.8	78.8	49.2
SEED	**79.2**	**61.1**	71.0	65.2	61.9	**47.1**	65.7	42.3	58.5	44.8	38.4	**36.9**	**15.4**	**36.0**	**38.9**	86.8	**73.4**
FAULT (Ours)	70.8	44.4	**74.2**	**78.3**	**85.7**	**47.1**	**68.7**	**44.6**	**59.7**	**46.5**	**39.8**	35.4	13.7	32.8	**38.9**	**87.5**	69.5
Qwen3-4B-Instruct-2507																	
Vanilla	50.0	11.1	16.1	13.0	19.0	0.0	19.4	4.0	11.0	4.3	5.9	6.9	1.9	8.8	6.1	20.5	1.6
GRPO	75.0	16.7	32.3	69.6	57.1	41.2	49.3	43.7	60.6	44.8	40.7	35.0	18.7	34.4	39.7	67.8	55.1
GiGPO	75.0	66.7	83.9	78.3	81.0	47.1	73.9	42.6	62.3	44.6	39.6	42.0	16.9	37.6	40.8	68.5	50.4
GRPO (diag-SFT)	79.2	61.1	77.4	82.6	76.2	41.2	71.6	**50.4**	65.5	**49.4**	**48.1**	42.3	18.5	45.6	45.7	82.2	66.4
GiGPO (diag-SFT)	79.2	**100.0**	80.6	78.3	76.2	70.6	80.6	47.6	**65.6**	48.8	45.7	**47.0**	**21.7**	**48.8**	**46.5**	83.5	68.0
SEED	83.3	**100.0**	83.9	73.9	**95.2**	70.6	84.3	48.7	63.5	47.3	43.7	43.6	20.1	44.0	44.4	86.2	**74.2**
FAULT (Ours)	**91.7**	**100.0**	**93.5**	**87.0**	85.7	**88.2**	**91.0**	47.9	64.1	46.8	44.3	41.8	20.4	41.6	43.8	**88.5**	71.5

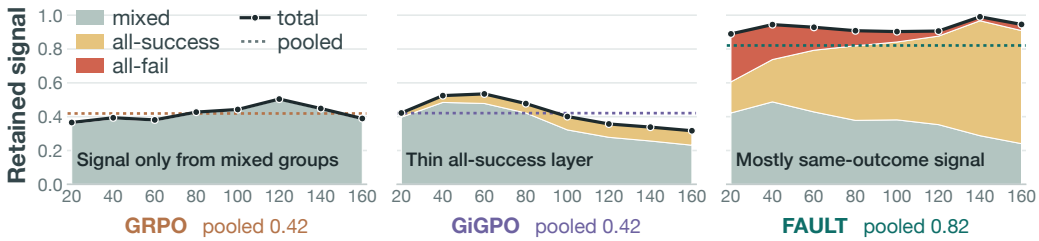


Figure 3: **Relative signal strength during training.** Stacked areas show the contributions of mixed, all-success, and all-fail groups to the signal-retention Index. Black lines show totals for 20-update windows; dotted lines show run-pooled values. Both baselines start from diagnosis-SFT.

failed episodes have more possible error steps, whereas in episodes of at most four steps, outcome rewards already fall close to the wrong step.

3.3 Analysis

We analyze signal coverage and strength, step-level credit, and training trajectory length.

3.3.1 Learning Signal in Same-Outcome Groups

Signal coverage. On ALFWorld, each prompt's eight rollouts form a *same-outcome* group (all-fail or all-success) or a *mixed* group. Same-outcome groups make up 59–64% of training groups, rising to about 75% by the end of GiGPO and FAULT training. To compare scales, we measure each group's credit gap, its highest minus lowest credit value. Mixed groups provide outcome contrast, so a group is usable if its gap exceeds 5% of that method's mean mixed-group gap, allowing weak but non-negligible signals (Appendix D.1). From FAULT's diagnosis-SFT checkpoint, GRPO, GiGPO and FAULT find usable signal in 41.4%, 72.5% and 95.1% of all groups (solid bars, Figure 1(a)).

Signal strength. In each 20-update window, we average the credit gaps of all-success, all-fail and mixed groups separately. Dividing each mean by the method's mixed-group mean in that window expresses signal strength relative to mixed groups. We then weight each ratio by that group type's share of the window's groups. The three contributions sum to the signal-retention Index; mixed groups contribute exactly their share, as their ratio is one (Appendix D.1). From the first to last window, FAULT's all-success share rises from 11.6% to 70.3%, while its mixed and all-fail shares fall from

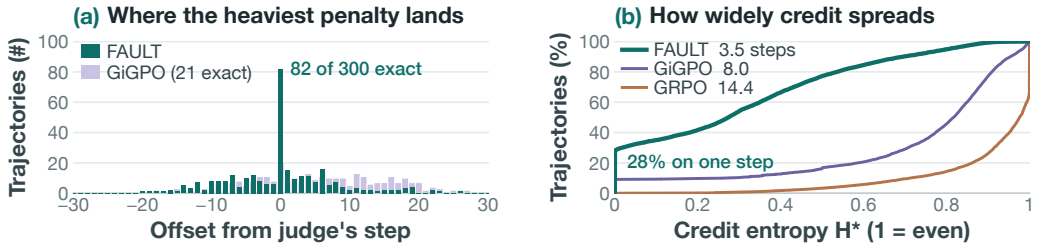


Figure 4: **Placement and concentration of step-level credit.** (a) Offset of the most penalized step from the judge's decisive step over 300 failed trajectories: FAULT in dark bars, GiGPO (same-configuration rerun) in light bars. (b) Cumulative distribution of the normalized credit entropy on same-outcome groups; curves further left are more concentrated, and the legend gives the mean effective number of credited steps. Baselines start from the diagnosis-SFT checkpoint.

42.2% to 24.1% and 46.3% to 5.6%. Its all-success contribution rises from 0.184 to 0.668, keeping the total near 0.9 (Figure 3). At 72.5% coverage, GiGPO's pooled Index is 0.421 (same-outcome contribution 0.063), near GRPO's 0.419, versus FAULT's 0.821 (same-outcome contribution 0.449).

Why outcome credit falls short. All mixed groups qualify, but same-outcome groups expose the limits of outcome credit. GRPO uses none because subtracting the group mean cancels equal rewards. GiGPO uses about 75% of all-success but no all-fail groups: discounting separates rollouts by steps left to success from shared anchor states, while failures return zero. FAULT uses 90% and 100%, respectively, as its diagnostic penalties distinguish trajectories despite equal outcomes (derivations in Appendix D.1). Successes can still contain correctable process errors: the audit finds 61.85% of FAULT's successes (8,719/14,097) penalized while keeping their success labels (Appendix D.2). The strength comparison uses budget gaps for FAULT and advantage gaps for baselines, so the pooled 0.821 is a contrast proxy, not a common-scale advantage estimate (Appendix D.1).

3.3.2 Step-Level Localization of Credit

Section 3.3.1 asked whether each group keeps a training signal. Here we ask how this signal is split across a trajectory's steps and whether it falls on the right steps.

Credit placement. An independent LLM judge reads each sampled failed training trajectory and marks its decisive error step, the earliest step whose correction would most likely turn the failure into a success. Each method ranks a trajectory's steps by the penalty it gave them. Figure 1(b) reports the mean reciprocal rank of the judge's step, $\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N} 1/\mathrm{rank}_i$ (allowing one step of error), and Figure 4(a) shows the offset of the most penalized step from the judge's step (Appendix D.3). FAULT reaches an MRR of 0.494 versus 0.266 for random ranking, and its most penalized step is exactly the judge's step in 82 of 300 trajectories. GiGPO reaches 0.303 with 21 exact matches, and its most penalized step usually comes several steps after the judge's; GRPO stays close to random.

Credit concentration. The normalized entropy measures how evenly a trajectory's credit is spread over its steps. We normalize each step's absolute credit (penalty shares for FAULT, advantages for baselines) into a distribution and divide its entropy by the log of the step count, so it is 1 for an even spread and 0 when all credit is on one step (Appendix D.3). On same-outcome groups, the mean normalized entropy is 0.29 for FAULT versus 0.72 for GiGPO and 0.91 for GRPO, and 28% of FAULT's penalized trajectories put the whole penalty on one step (Figure 4(b)).

Why step credit differs. GRPO gives every step of a trajectory the same credit. GiGPO's credit varies across steps but comes from comparing returns at states shared by several rollouts: a failed rollout is penalized at every state that a successful rollout of the same task also reached, and in all-fail groups this difference disappears (Proposition 2). FAULT penalizes only the steps cited by verified diagnoses. This explains both results: equal credit keeps GRPO near random and the most spread out, GiGPO's state-based credit is more concentrated but rarely hits the decisive error, and FAULT's credit is both the most concentrated and the best placed. Appendices D.3, D.4 and F.1 give more results.

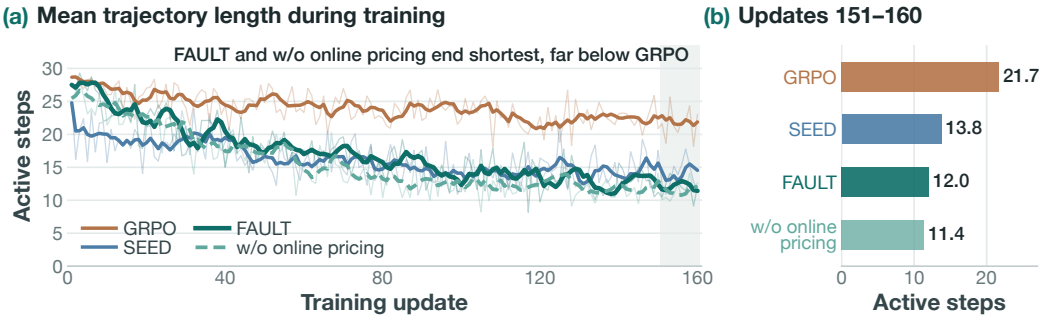


Figure 5: **ALFWorld training trajectory length.** (a) Mean active steps per training update (thin lines) and five-update moving averages (thick lines); shading marks updates 151–160. (b) Means over updates 151–160. All runs use a 30-step limit but differ in initialization and training recipe.

3.3.3 Training Trajectory Length

Figure 5 tracks the mean number of active steps per ALFWorld training episode, counting successes and failures. GRPO ends near 22 steps, while SEED, FAULT and the variant without online pricing (Appendix D.6) become much shorter, averaging 13.8, 12.0 and 11.4 steps over updates 151–160; FAULT is shorter than GRPO and SEED in every window of the final 5 to 40 updates (Appendix E.1). Two effects explain this: failed episodes run to the 30-step limit, so higher training success lowers the mean, and FAULT penalizes steps that waste this budget, such as revisits to searched locations, which recede faster under FAULT than under GRPO (diag-SFT; Appendices E.2–E.3).

4 Related Work

Outcome-based agentic RL. Agentic RL typically trains LLM policies from terminal rewards (Zhang et al., 2026a), often with GRPO (Shao et al., 2024), which gives each rollout's group-normalized outcome advantage to all its steps. Same-outcome groups therefore yield no outcome-driven gradient (Yu et al., 2025; Yang et al., 2026a). Agentic variants change how rollouts are sampled or compared (Dong et al., 2026; Feng et al., 2025), but still derive credit from outcomes.

Process-based agentic RL. Step rewards improve on outcome rewards in mathematical reasoning (Lightman et al., 2024; Wang et al., 2024), and agent RL now scores steps with learned models, rollouts or hindsight (Choudhury, 2025; Xi et al., 2026; Kim et al., 2026; Ma et al., 2026; Yao et al., 2026). Learned step rewards invite reward hacking (Guo et al., 2025), and GiGPO's outcome credit vanishes in all-fail groups (Feng et al., 2025). Language analyses of finished trajectories can instead guide later attempts (Shinn et al., 2023; Zhao et al., 2024), correct training data (Yuan et al., 2025; Zhang et al., 2026b) or be distilled into the policy (Wu et al., 2026). Yet the analysis stays text, its error claims are unchecked, and its weight is not learned from outcomes. FAULT keeps the terminal reward as the trusted signal and closes these gaps with verified diagnoses, outcome-fitted prices and bounded step credit (Appendix G).

5 Conclusion

Agentic RL has advanced rapidly in training agents for multi-step tasks, yet terminal rewards offer limited signals in same-outcome groups and little guidance on individual errors, particularly in long-horizon tasks. Natural-language diagnoses provide process information but need verification and quantification. FAULT addresses this gap by checking diagnostic evidence and learning relative error costs from terminal outcomes to assign explicit step-level credit. The policy and diagnoser co-evolve during RL. On ALFWorld, FAULT raises usable signal coverage to 95%, versus 41% for GRPO and 72% for GiGPO, and achieves a decisive-step MRR of 0.494, versus GiGPO's 0.303. At both model scales, it leads in ALFWorld success rate and WebShop task score while remaining competitive on shorter-horizon Search-based QA. Its ALFWorld training trajectories are also shorter than those of GRPO and SEED.

AI USE STATEMENT

In this work, we used generative AI tools to design and give feedback on the method and experiments, implement the method, assist in writing proofs, translate text and interpret results. Generative AI is also part of the method: GPT-5.6 Sol writes the teacher diagnoses for cold-start SFT and organizes free-text error labels into the error taxonomy (Section 2.2 and Appendix B.1), Qwen3.8-27B generates the Search-based QA solving trajectories used in cold-start SFT, and Qwen3.8-Max is the blind judge that labels decisive error steps for the localization analysis (Section 3.3.2). All RL training and evaluation tasks come from the public ALFWorld, WebShop and Search-based QA benchmarks. We have not used generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proofs, propose or refine hypotheses, or clean and reformat datasets. Additionally, we used generative AI tools to create and modify figures, suggest experimental parameters, write and edit code, draft parts of the paper, summarize and identify relevant literature, edit the paper for readability, and suggest the title and keywords. We have reviewed all AI-assisted work: the authors checked all AI-assisted content and tested all AI-assisted code, and for proofs, the authors proposed the proof ideas, AI tools helped with the derivations, and the authors verified every step. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies credit assignment for language-model agents using established research benchmarks. We did not recruit human participants or collect private user data. Our goal is to make credit assignment more transparent by linking step-level learning signals to diagnosed errors and observed task outcomes.

REPRODUCIBILITY STATEMENT

Section 2 and Appendix B specify the full method, including diagnosis verification, the pricing estimator and the penalty schedule, and Algorithm 1 in Appendix B.7 lists the complete training loop. Propositions 1 and 2 state their assumptions and are proved in Appendices B.4 and D.1. Appendix C describes the benchmarks, baselines, evaluation protocols, cold-start data and initialization, and Table 3 lists the training hyper-parameters. The diagnosis prompt is given in Appendix C.5, and the remaining prompts are released with our code. All benchmarks, base models and Qwen3.8-27B are public, and GPT-5.6 Sol and Qwen3.8-Max are available through commercial APIs.

REFERENCES

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

Jinhe Bi, Chennan Zhou, Zengjie Jin, Aniri, Shuo Lu, Wenke Huang, Hu Cao, Xun Xiao, Zhihong Zhu, Volker Tresp, Fei Shen, Yunpu Ma, and Tat-Seng Chua. ReflectRL: Learning from golden negative trajectories via reflective-to-direct reasoning. *arXiv preprint arXiv:2608.03972*, 2026.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2503.13657.

Gary Chamberlain. Analysis of covariance with qualitative data. *The Review of Economic Studies*, 47 (1):225–238, 1980.

Sanjiban Choudhury. Process reward models for LLM agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. TRAIL: Trace reasoning and agentic issue localization. *arXiv preprint arXiv:2505.08638*, 2025.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2507.19849.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. ReTool: Reinforcement learning for strategic tool use in LLMs. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2504.11536.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.10978.

David Firth. Bias reduction of maximum likelihood estimates. *Biometrika*, 80(1):27–38, 1993.

Yubin Ge, Salvatore Romeo, Jason Cai, Monica Sunkara, and Yi Zhang. SAMULE: Self-learning agents enhanced by multi-level reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 16591–16610, 2025. arXiv:2509.20562.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Georg Heinze and Michael Schemper. A solution to the problem of separation in logistic regression. *Statistics in Medicine*, 21(16):2409–2419, 2002.

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 (COLING)*, 2020.

Michael Y. Hu, Benjamin Van Durme, Jacob Andreas, and Harsh Jhamtani. Sample-efficient online learning in LM agents via hindsight trajectory rewriting. *arXiv preprint arXiv:2510.10304*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Ö. Arık, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling (COLM)*, 2025. arXiv:2503.09516.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Wonjoong Kim, Yeonjun In, Sangwu Park, Dongha Lee, and Chanyoung Park. PAIR: Prefix-aware internal reward model for multi-turn agent optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026. arXiv:2605.17877.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

Zhicong Li, Lingjie Jiang, Yulan Hu, Xingchen Zeng, Yixia Li, Xiangwen Zhang, Guanhua Chen, Zheng Pan, Xin Li, and Yong Liu. No more stale feedback: Co-evolving critics for open-world agent learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 12643–12660, 2026. arXiv:2601.06794.

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 *International Conference on Learning Representations (ICLR)*, 2024.

Xinbei Ma, Congmin Zheng, Jiyang Qiu, Jiale Hong, Yao Yao, Xiangmou Qu, Jiaxin Yin, Xingyu Lou, Jun Wang, Weiwen Liu, Weinan Zhang, Zhuosheng Zhang, and Hai Zhao. Retrospective progress-aware self-refinement for LLM agent training. *arXiv preprint arXiv:2606.14302*, 2026.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. `https://openai.com/index/gpt-5-6/`, 2026.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2504.13958.

Qwen Team. Qwen3-4B-Instruct-2507. `https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507`, 2025.

Qwen Team. Qwen3.8-27B. `https://huggingface.co/Qwen/Qwen3.8-27B`, 2026a.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork. `https://qwen.ai/blog?id=qwen3.8`, 2026b.

John Schulman. Approximating KL divergence. Blog post, `http://joschu.net/blog/kl-approx.html`, 2020.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.08146.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

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.

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 (ACL)*, 2024.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. WebAgent-R1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7909–7928, 2025. arXiv:2505.16421.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SEED: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentPRM: Process reward models for LLM agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026 (WWW)*, pp. 4184–4195, 2026. arXiv:2511.08325.

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

Kaibing Yang, Guangfeng Cai, Shengtian Yang, Shuo He, Yu Li, Mengyi Liu, Pengwei Chen, Jun Xu, and Lei Feng. Progress-conditioned group policy optimization for long-horizon agentic tasks. *arXiv preprint arXiv:2607.22724*, 2026a.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, and Jianhua Tao. OPID: On-policy skill distillation for agentic reinforcement learning. *arXiv preprint arXiv:2606.26790*, 2026b.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. 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 (EMNLP)*, 2018.

Jiashu Yao, Heyan Huang, Zeming Liu, and Yuhang Guo. Utilizing and calibrating hindsight process rewards via reinforcement with mutual information self-evaluation. *arXiv preprint arXiv:2604.11611*, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. Retroformer: Retrospective large language agents with policy gradient optimization. In *International Conference on Learning Representations (ICLR)*, 2024.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, Qiaobo Chen, Yinyuting Yin, Hao Zhang, Tengfei Shi, Liang Wang, Qiang Fu, Wei Yang, and Lanxiao Huang. Mastering complex control in MOBA games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6672–6679, 2020.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2503.14476.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-R: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita-Velez, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research*, 2026a. arXiv:2509.02547.

Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. Advancing LLM reasoning with natural language and numerical feedback. In *International Conference on Machine Learning (ICML)*, 2026b. arXiv:2506.03106.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Binbin Zheng, Zijun Xie, Guanqun Zhao, Enlei Gong, Xing Ma, Xiaoliang Fu, and Zeyu Chen. Group-reflective self-distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.28076*, 2026.

Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, Xiaoteng Ma, Xiaodong Yu, Gowtham Ramesh, Jialian Wu, Zicheng Liu, Pan Lu, James Zou, and Jiaxuan You. Where LLM agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

APPENDIX

Contents

A Notation **16**

B Method Details **17**

B.1 Cold Start: Taxonomy, Diagnosis SFT and Initial Prices 17

B.2 Online Self-Diagnosis, Verification and Aggregation 17

B.3 Terminal-Anchored Error Pricing 18

B.4 Score, Penalty Schedule and Order-Preserving Budget 20

B.5 Conserved Step Redistribution 22

B.6 Dual-Channel Advantage and Policy Update 22

B.7 Algorithm 23

C Additional Experimental Setup **24**

C.1 Benchmarks and Baselines 24

C.2 Evaluation Protocols 25

C.3 Cold-Start Data and Initialization 25

C.4 Training Configuration and Hyper-parameters 26

C.5 Diagnosis Prompt 28

D Additional Experimental Results **28**

D.1 Learning Signal in Same-Outcome Groups 28

D.2 Penalties on Successful Trajectories 30

D.3 Step-Level Localization: Measurement and Additional Views 30

D.4 Penalties on Wasted Steps 33

D.5 Diagnostic Quality Before and During RL 33

D.6 Ablation Studies and Cold-Start Data Scale 34

E Training Dynamics. **36**

E.1 Training Trajectory Length 36

E.2 Penalized Behaviors over Training 36

E.3 Initial Prices and Online Price Dynamics 37

F Case Studies **40**

F.1 Credit in an All-Fail Group 40

F.2 Recovery in a Successful Trajectory 40

G Extended Related Work **45**

A NOTATION

Table 2 lists the symbols of §2 and the main ones of Appendix B; others are defined where used, and Table 3 gives the constants. Three letters have two meanings: j also indexes classes (as in $\max_j$), c is a class c_k or a token's context c_ℓ, and e is the quote of an entry (k, t, e) or, in $k(e)$ and $e \in \mathcal{V}_{q,i,t}$, the entry itself. Superscripts such as (u), (v), (0), out, ep and step mark versions, roles or stages, never powers. Bold lower-case letters are vectors; calligraphic letters are sets, except the losses $\mathcal{L}$, the task distribution $\mathcal{D}$ and the information matrix $\mathcal{I}$; $\mathbb{1}[\cdot]$ is the indicator function, $\mathrm{clip}(\cdot, \mathrm{lo}, \mathrm{hi}) = \min(\max(\cdot, \mathrm{lo}), \mathrm{hi})$, and $\mathrm{sd}(\cdot)$ is the population standard deviation.

Table 2: Notation. "Where" gives the place where the symbols are first introduced.

Symbol	Meaning	Where
Indices, sets and constants		
$q \sim \mathcal{D}$	task and its rollout group; $\mathcal{D}$: training task distribution	§2.1
$i, j \in \{1, \dots, G\}$	rollout indices in a group; G rollouts per task	§2.1
$t, r \in \{1, \dots, T_{q,i}\}$	active steps; $T_{q,i}$: steps executed, not the step limit	§2.1
$k \in \{1, \dots, K\}$	priced error class; $\mathcal{C} = \{c_1, \dots, c_K\}$: frozen priced set	§2.2
$u \in \{1, \dots, U\}$; v	actor update (batch), U in total; accepted price refit	§2.4
$u_0,\ \Delta u,\ W$	first refit update, refit interval, window length	App. B.3.5
$\mathcal{Q}_{\mathrm{info}}$	groups with at least one success and one failure	§2.3
Trajectories, terminal signal and policies		
$\tau_{q,i}$; $o_{q,i,t}$, $a_{q,i,t}$	trajectory i of task q; context before response t; response t	§2.1
$y_{q,i} \in \{0, 1\}$	success indicator of $\tau_{q,i}$	§2.2
$R_{\min}, R_{\max}, \Delta R$; $R^{\mathrm{out}}_{q,i}$	failure/success rewards, gap; $R^{\mathrm{out}}_{q,i} = R_{\min} + \Delta R\, y_{q,i}$ if binary	§2.1
$J_{\mathrm{out}}(\theta)$	outcome objective (expected terminal reward)	Eq. (1)
$\pi_\theta,\ \pi_{\mathrm{old}},\ \pi_{\mathrm{ref}}$	trainable, per-batch behaviour and KL reference policies	§2.1
Structured diagnosis		
(k, t, e); $k(e)$	entry: frozen-table class, step, verbatim quote; priced class of entry e	§2.2
M; $\mathcal{M}_{q,i},\ M_{q,i}$	diagnoses sampled per trajectory; the parsable ones and their number	App. B.2
$\mathcal{V}_{q,i,t}$	verified entries citing step t	Eq. (5)
$n_{q,i,k}$; $x_{q,i,k}$, $\mathbf{x}_{q,i}$	mean verified count of class k; rate $n_{q,i,k}/T_{q,i}$; rate vector	§2.2
$k^\star_{q,i}$	class of the earliest verified error	App. B.2
$\mathcal{D}_{\mathrm{sft}}$; $\mathbf{d}_{q,i}$, $d_{q,i,\ell}$; $\mathcal{L}_{\mathrm{SFT}}$	SFT trajectories; verified diagnosis text, its ℓ-th token; SFT loss	Eq. (2)
Terminal-anchored pricing		
α_q	task intercept, removed by conditioning	§2.3
$\boldsymbol{\beta} = (\beta_1, \dots, \beta_K)$	shared error-outcome coefficients; negative ones become prices	§2.3
$\mathcal{Y}_q$; $\mathcal{A}_q$, Γ	observed success set; all same-size candidate sets, one member	Eq. (3)
$\mathrm{Pr}_q(\boldsymbol{\beta})$; $\boldsymbol{\beta}^\star$	conditional probability of $\mathcal{Y}_q$; its unpenalized maximizer	Eq. (3)
$\widehat{\boldsymbol{\beta}}(\boldsymbol{\beta}_{\mathrm{ref}})$; $\boldsymbol{\beta}^{(0)}$; $\boldsymbol{\beta}^{(v)}$	fit with reference $\boldsymbol{\beta}_{\mathrm{ref}}$; anchor $\min(\widehat{\boldsymbol{\beta}}^{(0)}, 0)$; refit-$v$ coefficients	App. B.3.2
$\rho,\ \rho_u$	cold-start shrinkage; its decaying online schedule	Eq. (14)
ν	smoothing weight on the previous coefficients	Eq. (20)
w_k; $\mathbf{w}^{(u)}$	max-normalized price in $[0, 1]$, 0 unless $\beta_k < 0$; prices of batch u	Eq. (4)
Score, budget and redistribution		
$S_{q,i}$	rate-weighted, clipped error score in $[0, 1]$	Eq. (4)
λ_u; $\overline{T}^{(0)}$	penalty strength (warm-up, cosine decay); first-batch mean length	Eq. (4)
$P_{q,i}$	budget $\min(\lambda_u S_{q,i}, \eta\, \Delta R)$	Eq. (4)
$h_{q,i,t}$	total price located at step t	Eq. (5)
$\delta_{q,i,t}$; $\bar{\delta}$	step share of the budget, $\sum_t \delta_{q,i,t} = P_{q,i}$; group-mean share	Eq. (5)
Dual-channel advantage and policy update		
$E_{q,i,t}$; Ω_q	episode score $R^{\mathrm{out}}_{q,i} - \delta_{q,i,t}$; all step samples of task q	Eq. (6)
$\widetilde{Z}_{q,i,t}$; γ	discounted return from step t minus the share; discount factor	Eq. (7)
$\mathcal{H}(o)$	anchor-state group: same-task samples with similar contexts	Eq. (7)
$A^{\mathrm{ep}},\ A^{\mathrm{step}},\ A_{q,i,t}$; ω	episode, step and joint advantage $A^{\mathrm{ep}} + \omega A^{\mathrm{step}}$; step weight	Eq. (6)
$\mathcal{U}_{q,i,t}$; ℓ; N	tokens of response t; token index; batch token count	Eq. (8)
ϱ_ℓ; ϵ	ratio $\pi_\theta(a_\ell \mid c_\ell)/\pi_{\mathrm{old}}(a_\ell \mid c_\ell)$; two-sided clip width	Eq. (8)
D^{LV}_ℓ, λ_{KL}; $\mathcal{L}_{\mathrm{actor}}$	token KL estimate to π_{ref}, its weight; actor loss	Eq. (8)

B METHOD DETAILS

This appendix details each module of §2 in the order of the training pipeline, from the cold start (B.1) to the full algorithm and its implementation notes (B.7). Appendix C gives the benchmark-specific instantiation and Appendix C.4 the hyper-parameters.

B.1 COLD START: TAXONOMY, DIAGNOSIS SFT AND INITIAL PRICES

Before RL, a one-time cold start produces a frozen error taxonomy, a diagnosis-SFT checkpoint and initial error prices. Appendix C.5 gives the diagnosis prompt; Appendix C provides the data and training settings.

Teacher diagnosis. We collect training trajectories from the base policy and, optionally, early checkpoints trained with outcome-only RL. A teacher diagnoses each trajectory once using greedy decoding, given its task, outcome, reward and step count. Each error report identifies a mechanism through a free-form label, a brief explanation, cited steps, a verbatim quote and a severity hint. The teacher is used only during cold start.

Taxonomy construction. An LLM organizes the accepted labels into L1 error families and L2 mechanisms. Starting from an empty table, a growth pass processes labels in descending frequency, matching existing mechanisms or adding new classes when uncertain. A second pass assigns the remaining labels to existing classes or marks them as off-table. Rare L2 classes are merged back into their L1 families, and the taxonomy is frozen. We select K pricing classes $\mathcal{C}$ from the retained L2 classes and L1 fallback categories using minimum verified counts and frequency-rank thresholds. Selection ensures data support; the fitted price may still be zero.

Diagnosis SFT. Teacher diagnoses mapped to the frozen taxonomy become the structured targets $\mathbf{d}_{q,i}$. We reserve a held-out set stratified by checkpoint, outcome and task type; the remaining diagnoses form $\mathcal{D}_{\text{sft}}$, supplemented during training by a few action-replay examples from successful trajectories. Each diagnosis input contains the rules, taxonomy, task, outcome and trajectory, with the teacher's JSON as its target. We fine-tune all model parameters using Eq. (2); the diagnosis targets contain no prices, scores or advantages.

Initial prices. The verified teacher diagnoses also provide the error rates used with terminal outcomes to fit $\widehat{\boldsymbol{\beta}}^{(0)}$ and obtain $\mathbf{w}^{(1)}$. Appendix B.3.4 details this fit, which learns prices separately from diagnosis SFT.

B.2 ONLINE SELF-DIAGNOSIS, VERIFICATION AND AGGREGATION

Online diagnosis provides error rates for pricing and step evidence for penalty allocation.

Diagnosis generation. Before each actor update, the unchanged rollout policy samples M diagnoses per completed trajectory at a fixed temperature, given the frozen taxonomy, task, outcome and trajectory. Each diagnosis is JSON containing the reported outcome and a bounded list of entries (k, t, e): class, step and quote. Free-form labels outside the taxonomy are logged separately and excluded from scoring.

Entry verification. An entry is verified when its class belongs to the frozen taxonomy (L2, or an L1 fallback when L2 is absent), its step $t \in \{1, \ldots, T_{q,i}\}$ was executed, and its quote meets the minimum length and matches a verbatim, case-sensitive substring of that step's text (observation, response and environment feedback). Sampling settings and verification thresholds are listed in Table 3.

Trajectory selection. A trajectory passes the gate when at least two diagnoses are parsable JSON and a majority of the parsable diagnoses correctly report its outcome. If it fails, it is excluded from price fitting; its score $S_{q,i}$ is filled with the mean score of same-outcome trajectories in the batch and marked as imputed.

Error aggregation. We map verified entries to pricing classes. Let $\mathcal{M}_{q,i}$ be the parsable diagnoses of $\tau_{q,i}$, $M_{q,i} = |\mathcal{M}_{q,i}|$, and $n_{q,i,m,k}$ the number of verified class-k entries in diagnosis m, counting repeated entries separately. We average these counts across all parsable diagnoses and normalize by

the number of executed steps:

$$n_{q,i,k} = \frac{1}{M_{q,i}} \sum_{m \in \mathcal{M}_{q,i}} n_{q,i,m,k}, \qquad x_{q,i,k} = \frac{n_{q,i,k}}{T_{q,i}}. \tag{9}$$

Averaging retains a verified error reported by only one diagnosis, but reduces its contribution. We also record $k^*_{q,i}$, the class of the earliest verified error across all parsable diagnoses. The rates form the pricing record $(q, \mathbf{x}_{q,i}, y_{q,i})$; for graded outcomes, the normalized reward replaces $y_{q,i}$ as defined in Appendix B.3.2.

B.3 Terminal-Anchored Error Pricing

We estimate how verified error rates relate to terminal outcomes while accounting for differences in task difficulty. We first remove the task-specific intercept, derive the regularized estimators, and then describe how their coefficients produce initial prices and online updates.

B.3.1 Conditional Likelihood

Let $\mathcal{G}_q$ contain the trajectories retained for fitting in group q, with error-rate vectors $\mathbf{x}_{q,i}$ and binary outcomes $y_{q,i}$. The observed success set is $\mathcal{Y}_q = \{i \in \mathcal{G}_q : y_{q,i} = 1\}$, with size $s_q = |\mathcal{Y}_q|$. We use a logistic model with shared error coefficients $\boldsymbol{\beta}$ and a task-specific intercept α_q:

$$p_{q,i} = \Pr(y_{q,i} = 1 \mid \mathbf{x}_{q,i}, q) = \mathrm{sigmoid}(\alpha_q + \boldsymbol{\beta}^\top \mathbf{x}_{q,i}). \tag{10}$$

Conditioning on s_q removes α_q without estimating a separate difficulty parameter for every group (Chamberlain, 1980). To show this, define the candidate success sets and their feature sums as

$$\mathcal{A}_q = \{\Gamma \subseteq \mathcal{G}_q : |\Gamma| = s_q\}, \qquad \mathbf{f}_{q,\Gamma} = \sum_{i \in \Gamma} \mathbf{x}_{q,i}.$$

Assuming conditionally independent outcomes under the model, the probability that exactly the trajectories in Γ succeed is

$$\begin{aligned} \Pr(\Gamma \mid \alpha_q, \boldsymbol{\beta}) &= \prod_{i \in \Gamma} p_{q,i} \prod_{i \in \mathcal{G}_q \setminus \Gamma} (1 - p_{q,i}) \\ &= \frac{\exp\big(|\Gamma|\alpha_q + \boldsymbol{\beta}^\top \mathbf{f}_{q,\Gamma}\big)}{\prod_{i \in \mathcal{G}_q} \big[1 + \exp(\alpha_q + \boldsymbol{\beta}^\top \mathbf{x}_{q,i})\big]}, \end{aligned} \tag{11}$$

where conditioning on the group's feature vectors is implicit. The denominator is common to all candidate sets. After conditioning on the observed success count, this denominator cancels, followed by the common factor $\exp(s_q \alpha_q)$:

$$\begin{aligned} \Pr\left(\mathcal{Y}_q \,\middle|\, \sum_{i \in \mathcal{G}_q} y_{q,i} = s_q\right) &= \frac{\Pr(\mathcal{Y}_q \mid \alpha_q, \boldsymbol{\beta})}{\sum_{\Gamma \in \mathcal{A}_q} \Pr(\Gamma \mid \alpha_q, \boldsymbol{\beta})} \\ &= \frac{\exp(s_q \alpha_q) \exp(\boldsymbol{\beta}^\top \mathbf{f}_{q,\mathcal{Y}_q})}{\exp(s_q \alpha_q) \sum_{\Gamma \in \mathcal{A}_q} \exp(\boldsymbol{\beta}^\top \mathbf{f}_{q,\Gamma})} \\ &= \frac{\exp(\boldsymbol{\beta}^\top \mathbf{f}_{q,\mathcal{Y}_q})}{\sum_{\Gamma \in \mathcal{A}_q} \exp(\boldsymbol{\beta}^\top \mathbf{f}_{q,\Gamma})} = \Pr_q(\boldsymbol{\beta}). \end{aligned} \tag{12}$$

This recovers Eq. (3) using only within-group comparisons. If all trajectories succeed or all fail, $\mathcal{A}_q$ contains one set and $\Pr_q(\boldsymbol{\beta}) = 1$, so the group contributes neither loss nor gradient to this estimator. The binary fit therefore uses $\mathcal{Q}_{\mathrm{info}}$, the groups containing both outcomes.

B.3.2 Regularized Coefficient Estimation

Binary outcomes. Write $\ell(\boldsymbol{\beta}) = \sum_{q \in \mathcal{Q}_{\mathrm{info}}} \log \Pr_q(\boldsymbol{\beta})$. For each candidate set, let $p_{q,\Gamma}(\boldsymbol{\beta})$ denote its conditional probability from Eq. (12), with Γ in place of $\mathcal{Y}_q$. Differentiating the log-sum-exp

normalizer gives

$$\begin{aligned}\boldsymbol{\mu}_q(\boldsymbol{\beta}) &= \sum_{\Gamma\in\mathcal{A}_q} p_{q,\Gamma}(\boldsymbol{\beta})\mathbf{f}_{q,\Gamma},\\ \nabla\ell(\boldsymbol{\beta}) &= \sum_{q\in\mathcal{Q}_{\text{info}}} \big(\mathbf{f}_{q,\mathcal{Y}_q} - \boldsymbol{\mu}_q\big),\\ \mathcal{I}(\boldsymbol{\beta}) = -\nabla^2\ell(\boldsymbol{\beta}) &= \sum_{q\in\mathcal{Q}_{\text{info}}}\sum_{\Gamma\in\mathcal{A}_q} p_{q,\Gamma}(\boldsymbol{\beta})\big(\mathbf{f}_{q,\Gamma} - \boldsymbol{\mu}_q\big)\big(\mathbf{f}_{q,\Gamma} - \boldsymbol{\mu}_q\big)^\top.\end{aligned} \tag{13}$$

Thus the information matrix is the sum of the within-group covariances of candidate feature sums. To stabilize estimation from limited data, we fit

$$\begin{aligned}\widehat{\boldsymbol{\beta}}(\boldsymbol{\beta}_{\text{ref}}) &= \operatorname*{arg\,min}_{\|\boldsymbol{\beta}\|_\infty\le\beta_{\max}} \mathcal{L}_{\text{price}}(\boldsymbol{\beta}),\\ \mathcal{L}_{\text{price}}(\boldsymbol{\beta}) &= -\ell(\boldsymbol{\beta}) - \tfrac{1}{2}\log\det\mathcal{I}(\boldsymbol{\beta}) + \rho\|\boldsymbol{\beta}-\boldsymbol{\beta}_{\text{ref}}\|_2^2.\end{aligned} \tag{14}$$

The first term fits within-task outcomes; the Firth correction addresses small-sample bias and separation (Firth, 1993; Heinze & Schemper, 2002); and the final term shrinks uncertain coefficients toward $\boldsymbol{\beta}_{\text{ref}}$. The log-determinant requires positive-definite information on the fitted coordinates; regularization does not make indistinguishable error classes identifiable. We enumerate candidate sets exactly and solve the bounded objective with L-BFGS-B (Byrd et al., 1995; Zhu et al., 1997). The resulting $\widehat{\boldsymbol{\beta}}$ is the regularized numerical estimate used by the method, whereas $\boldsymbol{\beta}^\star$ in Eq. (3) denotes the unpenalized estimator.

Graded outcomes. For graded rewards, let $z_{q,i} = \text{clip}((R^{\text{out}}_{q,i} - R_{\min})/\Delta R, 0, 1)$. We replace the binary likelihood by squared-error regression with unpenalized group intercepts:

$$\min_{\boldsymbol{\beta},\{\alpha_q\}} \sum_{q,i}\big(z_{q,i} - \alpha_q - \boldsymbol{\beta}^\top\mathbf{x}_{q,i}\big)^2 + \rho\|\boldsymbol{\beta}-\boldsymbol{\beta}_{\text{ref}}\|_2^2, \tag{15}$$

where the sum covers the retained trajectories in the fitting data. For fixed $\boldsymbol{\beta}$, setting the derivative with respect to α_q to zero gives $\widehat{\alpha}_q = \bar{z}_q - \boldsymbol{\beta}^\top\bar{\mathbf{x}}_q$, where the bars denote means within group q. Substituting this expression removes the intercepts by centring both features and rewards: $\mathbf{x}^c_{q,i} = \mathbf{x}_{q,i} - \bar{\mathbf{x}}_q$ and $z^c_{q,i} = z_{q,i} - \bar{z}_q$. Let $\mathbf{X}_c$ contain the centred feature rows and $\mathbf{z}_c$ the centred rewards. The reduced objective and its normal equations yield

$$\begin{aligned}\mathcal{L}_{\text{ridge}}(\boldsymbol{\beta}) &= \|\mathbf{z}_c - \mathbf{X}_c\boldsymbol{\beta}\|_2^2 + \rho\|\boldsymbol{\beta}-\boldsymbol{\beta}_{\text{ref}}\|_2^2,\\ (\mathbf{X}_c^\top\mathbf{X}_c + \rho\mathbf{I})\widehat{\boldsymbol{\beta}}(\boldsymbol{\beta}_{\text{ref}}) &= \mathbf{X}_c^\top\mathbf{z}_c + \rho\boldsymbol{\beta}_{\text{ref}},\\ \widehat{\boldsymbol{\beta}}(\boldsymbol{\beta}_{\text{ref}}) &= (\mathbf{X}_c^\top\mathbf{X}_c + \rho\mathbf{I})^{-1}(\mathbf{X}_c^\top\mathbf{z}_c + \rho\boldsymbol{\beta}_{\text{ref}}).\end{aligned} \tag{16}$$

Here $\mathbf{I}$ is the identity matrix and $\rho > 0$ makes the system invertible. This ridge estimator replaces the likelihood and Firth terms for graded outcomes; the price mapping and update rules below are shared by both branches.

B.3.3 From Coefficients to Prices

Negative coefficients associate higher error rates with lower outcomes within a task. We use their magnitudes as relative costs and assign no reward to positive coefficients:

$$\widetilde{w}_k = \max(-\beta_k, 0)\,\sigma_k, \qquad w_k = \begin{cases}\widetilde{w}_k/\max_j\widetilde{w}_j, & \max_j\widetilde{w}_j > 0,\\ 0, & \text{otherwise.}\end{cases} \tag{17}$$

At cold start, σ_k is the population standard deviation of error rate k over all cold-start trajectories, including zeros and groups excluded from the binary likelihood. Then $|\beta_k|\sigma_k$ measures the change in the linear predictor associated with one standard deviation of that rate. For online refits, $\sigma_k = 1$, recovering Eq. (4). When any price is positive, the most expensive class has price one; λ_u controls the overall penalty strength (Appendix B.4). These prices summarize predictive associations with outcomes, not causal effects.

B.3.4 Cold-Start Fit and Initialization

Cold-start fitting uses the teacher-diagnosed trajectories of Appendix B.1, with $M_{q,i} = 1$ in Eq. (9). Each group q consists of the rollouts of one task from one checkpoint. To separate error density from the additional association between trajectory length and outcome, the initial fit includes a length covariate:

$$\widetilde{\mathbf{x}}_{q,i} = \begin{pmatrix} \mathbf{x}_{q,i} \\ T^{\mathrm{fit}}_{q,i} \end{pmatrix}, \qquad T^{\mathrm{fit}}_{q,i} = \frac{T_{q,i}}{T^{\max}}, \qquad \widetilde{\boldsymbol{\beta}} = \begin{pmatrix} \boldsymbol{\beta} \\ \beta_T \end{pmatrix}. \tag{18}$$

Unlike the group intercept, this covariate can vary within a group and is therefore retained after intercept removal. We apply the appropriate estimator of Appendix B.3.2 to these extended features, using a zero reference and fixed shrinkage strength ρ; the binary optimizer also starts at zero and bounds the length coefficient. The coefficient β_T controls for length in fitting but never enters the price vector.

Let $\widehat{\boldsymbol{\beta}}^{(0)}$ contain only the K error coefficients from this fit. Equation (17), with the cold-start scales, gives the initial prices $\mathbf{w}^{(1)}$. We also retain the non-positive part of the fitted error coefficients as a fixed reference for online estimation:

$$\boldsymbol{\beta}^{(0)} = \min\left(\widehat{\boldsymbol{\beta}}^{(0)}, \mathbf{0}\right), \tag{19}$$

where the minimum is coordinate-wise. Thus $\widehat{\boldsymbol{\beta}}^{(0)}$ is the initial fit, $\boldsymbol{\beta}^{(0)}$ is the frozen reference, and $\mathbf{w}^{(1)}$ is the initial price vector. Online refits use only the K error rates, without the length covariate.

B.3.5 Online Refits and Price Updates

Fitting recent trajectories. Batch u uses the fixed prices $\mathbf{w}^{(u)}$. After its actor update, the verified records $(q, \mathbf{x}_{q,i}, y_{q,i})$ with non-imputed scores enter a sliding window of the last W updates; graded outcomes use $z_{q,i}$ in place of $y_{q,i}$. Refits begin at u_0 and recur every Δu updates. The binary fit requires enough mixed-outcome groups and successes; only classes with sufficient non-zero observations and within-group variation are fitted (thresholds in Table 3). If these checks fail or a fit is rejected, the previous prices remain in use. Numerical binary fits require convergence, a finite objective, and finite, feasible coefficients; implementation differences are recorded in Appendix B.7.

For the v-th accepted refit, we use the current window to obtain $\widehat{\boldsymbol{\beta}}^{(v)} = \widehat{\boldsymbol{\beta}}(\boldsymbol{\beta}^{(0)})$ with shrinkage strength ρ_u, which decreases over training. The reference stays fixed at $\boldsymbol{\beta}^{(0)}$; each binary optimization also starts there. The previous online coefficients enter only the subsequent smoothing step, not the shrinkage target.

Updating the deployed coefficients. Let $\mathcal{K}_v$ be the classes included in the accepted refit. Starting from $\boldsymbol{\beta}^{(0)}$, update the deployed coefficients by

$$\beta_k^{(v)} = \begin{cases} \nu\beta_k^{(v-1)} + (1-\nu)\hat{\beta}_k^{(v)}, & k \in \mathcal{K}_v \text{ and } \hat{\beta}_k^{(v)} \le 0, \\ \beta_k^{(v-1)}, & \text{otherwise,} \end{cases} \qquad \nu = 0.7. \tag{20}$$

This exponential moving average gives weight 0.7 to the previous value and 0.3 to the new fit. An unfitted class or a positive new coefficient keeps its previous value. We map $\boldsymbol{\beta}^{(v)}$ to prices with $\sigma_k = 1$ in Eq. (17); the new prices take effect from batch $u + 1$, so a batch is never scored using a fit to its own outcomes. The smoothing acts on coefficients; the change in scaling after cold start means that it does not guarantee continuity of the normalized prices.

B.4 Score, Penalty Schedule and Order-Preserving Budget

With error prices fixed within a batch, we compute a trajectory score, scale it over training, and cap the resulting penalty before distributing it across steps. The cap preserves the ordering of binary outcomes in the episode channel, as shown below.

B.4.1 TRAJECTORY SCORE

For a trajectory that passes the diagnosis gate, the batch prices $\mathbf{w} = \mathbf{w}^{(u)}$ and soft counts from Eq. (9) give

$$\begin{aligned} S_{q,i}^{\mathrm{raw}} &= \sum_{k=1}^{K} w_k n_{q,i,k} + (b-1) w_{k^*_{q,i}}, \\ S_{q,i} &= \mathrm{clip}\left(\frac{S_{q,i}^{\mathrm{raw}}}{T_{q,i}}, 0, 1 \right) \\ &= \mathrm{clip}\left(\sum_{k=1}^{K} w_k x_{q,i,k} + \frac{(b-1) w_{k^*_{q,i}}}{T_{q,i}}, 0, 1 \right). \end{aligned} \tag{21}$$

Here $b \geq 1$ adds weight to the earliest verified error class $k^*_{q,i}$; the extra term is zero when no verified error exists. We use a first-error penalty factor of $b = 1.5$ in all experiments, adding $0.5\, w_{k^*_{q,i}}$ to $S_{q,i}^{\mathrm{raw}}$. Setting $b = 1$ recovers the score in Eq. (4). The rate term is independent of length at fixed error rates; the first-error term retains an explicit $1/T_{q,i}$ dependence. Both successful and failed trajectories are scored; trajectories that fail the diagnosis gate use the imputed score of Appendix B.2.

B.4.2 PENALTY SCHEDULE

Let $u = 1, \ldots, U$ index actor updates and $0 \leq U_{\mathrm{w}} < U$ be the warm-up length. We measure $\overline{T}^{(0)}$, the mean executed trajectory length in the first rollout batch, before any actor update. The non-negative strengths λ_{warm} and λ_{peak} are fixed multiples of $\overline{T}^{(0)}$ (Table 3; benchmark variants in Appendix C.4). This scales a per-step error score to a trajectory-level penalty using the initial trajectory length. We keep $\overline{T}^{(0)}$ fixed so that later changes in trajectory length do not change this global scale. The penalty strength is

$$\lambda_u = \begin{cases} \lambda_{\mathrm{warm}}, & u \leq U_{\mathrm{w}}, \\ \dfrac{\lambda_{\mathrm{peak}}}{2} \Big[1 + \cos\Big(\pi \, \dfrac{u - U_{\mathrm{w}}}{U - U_{\mathrm{w}}} \Big) \Big], & u > U_{\mathrm{w}}. \end{cases} \tag{22}$$

The warm-up strength is constant; the remaining updates follow the cosine branch, with $\lambda_U = 0$ so that the diagnostic penalty vanishes at the final update.

B.4.3 BOUNDED BUDGET AND ORDER PRESERVATION

Let $\Delta R = R_{\max} - R_{\min} > 0$ be the terminal reward range, equal to the success–failure reward gap for binary outcomes. For $0 \leq \eta < 1$, the trajectory budget is

$$P_{q,i} = \min\big(\lambda_u S_{q,i}, \eta \Delta R \big), \qquad 0 \leq P_{q,i} \leq \eta \Delta R. \tag{23}$$

This budget is nondecreasing in $S_{q,i}$, with ties once the cap is reached; Appendix B.5 distributes it across steps without changing its total.

Proposition 1 (Episode-channel order preservation)**.** *Fix a task q with binary terminal rewards $R_{\min} + \Delta R\, y_{q,i}$ and episode scores $E_{q,i,t} = R_{q,i}^{\mathrm{out}} - \delta_{q,i,t}$. For any successful trajectory i, failed trajectory j, and their respective steps t and r,*

$$A^{\mathrm{ep}}_{q,i,t} - A^{\mathrm{ep}}_{q,j,r} = E_{q,i,t} - E_{q,j,r} \geq (1 - \eta) \Delta R > 0.$$

Proof. The conserved split has non-negative shares, so $0 \leq \delta_{q,i,t} \leq P_{q,i}$. Using Eq. (23),

$$\begin{aligned} E_{q,i,t} - E_{q,j,r} &= \Delta R - \delta_{q,i,t} + \delta_{q,j,r} \\ &\geq \Delta R - P_{q,i} \\ &\geq (1 - \eta) \Delta R > 0. \end{aligned}$$

Both episode advantages subtract the same task-group mean in Eq. (6), leaving their difference unchanged. □

The proposition concerns the diagnostic episode channel; additional reward terms in Appendix B.6 are outside its scope. For graded outcomes, the same argument gives

$$E_{q,i,t} - E_{q,j,r} \geq R_{q,i}^{\mathrm{out}} - R_{q,j}^{\mathrm{out}} - \eta \Delta R.$$

This bound guarantees strict order when the original reward gap exceeds the penalty cap.

B.5 Conserved Step Redistribution

Given the bounded budget $P_{q,i}$ from Appendix B.4, we use the diagnosed error locations to determine each step's share without changing the total penalty.

B.5.1 Localization Weights

Let $\mathcal{V}_{q,i,t}$ collect the verified entries citing step t across all parsable diagnoses, retaining repeated reports as separate entries. Each entry contributes its class price, giving

$$h_{q,i,t} = \sum_{e \in \mathcal{V}_{q,i,t}} w_{k(e)} \geq 0.$$

Thus a class reported at the same step by three diagnoses contributes $3w_k$; the relative support across steps determines their allocation weights. Quote length and severity hints do not affect these weights. When at least one weight is positive, we boost the earliest such step:

$$t^*_{q,i} = \min\{t : h_{q,i,t} > 0\}, \qquad \widetilde{h}_{q,i,t} = \big[1 + (b_{\mathrm{loc}} - 1)\mathbb{1}\big[t = t^*_{q,i}\big]\,\big] h_{q,i,t}. \tag{24}$$

If all weights are zero, we set $\widetilde{h}_{q,i,t} = 0$ for every step. We use $b_{\mathrm{loc}} = 1.5$ (Table 3). Unlike the score-level boost b in Appendix B.4, which can increase the total budget, b_{loc} changes only its allocation. The boosted step is the earliest with positive price, which need not be the earliest verified error.

B.5.2 Normalized Allocation and Conservation

The complete allocation rule is

$$\delta_{q,i,t} = \begin{cases} P_{q,i} \dfrac{\widetilde{h}_{q,i,t}}{\sum_{r=1}^{T_{q,i}} \widetilde{h}_{q,i,r}}, & \displaystyle\sum_{r=1}^{T_{q,i}} \widetilde{h}_{q,i,r} > 0, \\ P_{q,i}\mathbb{1}[t = T_{q,i}], & \text{otherwise.} \end{cases} \tag{25}$$

This is Eq. (5) with the boosted weights and a fallback for missing locations. A positive budget without a positive localization weight can arise from score imputation (Appendix B.2); it is assigned to the last active step. Both branches use non-negative proportions summing to one, so

$$0 \leq \delta_{q,i,t} \leq P_{q,i}, \qquad \sum_{t=1}^{T_{q,i}} \delta_{q,i,t} = P_{q,i}. \tag{26}$$

With localized evidence, zero-weight steps receive no penalty; when $P_{q,i} = 0$, all shares vanish. Redistribution therefore changes where the budget is charged, while preserving the total set by the rate-based score, schedule and cap.

B.6 Dual-Channel Advantage and Policy Update

The step shares $\delta_{q,i,t}$ enter two comparisons: an episode channel across the task's rollouts and a step channel across similar pre-response contexts. Their combined advantage then supervises the response tokens of each step.

B.6.1 Comparison Groups and Local Scores

The task group contains all active step samples,

$$\Omega_q = \{(j, r) : 1 \leq j \leq G,\ 1 \leq r \leq T_{q,j}\}, \qquad |\Omega_q| = \sum_{j=1}^{G} T_{q,j}.$$

Each step counts once, so longer trajectories contribute more samples to the episode baseline. For the step channel, $\mathcal{H}(o_{q,i,t}) \subseteq \Omega_q$ contains samples clustered with $o_{q,i,t}$ by pre-response context similarity. A context joins a cluster when its text similarity to the representative meets the threshold in Table 3;

otherwise it starts a new cluster. These groups approximate comparable contexts rather than asserting identical latent states.

We first compute the return from the environment rewards, then subtract the diagnostic share locally:

$$Z^{\mathrm{env}}_{q,i,t} = \sum_{r=t}^{T_{q,i}} \gamma^{r-t} R^{\mathrm{env}}_{q,i,r}, \qquad E_{q,i,t} = R^{\mathrm{out}}_{q,i} - \delta_{q,i,t}, \qquad \widetilde{Z}_{q,i,t} = Z^{\mathrm{env}}_{q,i,t} - \delta_{q,i,t}. \tag{27}$$

For purely terminal rewards, $Z^{\mathrm{env}}_{q,i,t} = \gamma^{T_{q,i}-t} R^{\mathrm{out}}_{q,i}$, recovering Eq. (7). The share is not added to the environment reward and discounted back to earlier steps: the diagnosis has already identified where to charge it. When the base recipe includes a rule-based invalid-action penalty, it also enters $E_{q,i,t}$; the displayed equations omit this additional term.

B.6.2 Centred Advantages and the Effect of Diagnosis

Following the two comparison groups, we define

$$\mu^{\mathrm{ep}}_q = \frac{1}{|\Omega_q|} \sum_{(j,r)\in\Omega_q} E_{q,j,r}, \qquad A^{\mathrm{ep}}_{q,i,t} = E_{q,i,t} - \mu^{\mathrm{ep}}_q,$$
$$\mu^{\mathrm{step}}(o) = \frac{1}{|\mathcal{H}(o)|} \sum_{(j,r)\in\mathcal{H}(o)} \widetilde{Z}_{q,j,r}, \quad A^{\mathrm{step}}_{q,i,t} = \widetilde{Z}_{q,i,t} - \mu^{\mathrm{step}}(o_{q,i,t}), \tag{28}$$
$$A_{q,i,t} = A^{\mathrm{ep}}_{q,i,t} + \omega A^{\mathrm{step}}_{q,i,t}.$$

On ALFWorld and WebShop, both channels subtract a mean without dividing by a standard deviation, keeping advantages in reward units; Search-based QA also divides by the standard deviation (Table 3). A singleton anchor group has $A^{\mathrm{step}} = 0$, while its episode advantage can remain non-zero.

To isolate the effect of diagnosis, hold the rollouts, comparison groups and other reward terms fixed, and let $A^{(0)}_{q,i,t}$ be the joint advantage with $\delta = 0$. For a non-empty comparison set $\mathcal{G} \subseteq \Omega_q$, write $\bar{\delta}_{\mathcal{G}} = |\mathcal{G}|^{-1} \sum_{(j,r)\in\mathcal{G}} \delta_{q,j,r}$. Subtracting the two versions of Eq. (28) gives

$$A_{q,i,t} - A^{(0)}_{q,i,t} = -\big(\delta_{q,i,t} - \bar{\delta}_{\Omega_q}\big) - \omega\big(\delta_{q,i,t} - \bar{\delta}_{\mathcal{H}(o_{q,i,t})}\big). \tag{29}$$

Thus each channel lowers a step's advantage relative to its own comparison group when its penalty exceeds that group's mean; a uniform penalty within a group cancels. The two channels compare the same local cost against different reference sets, rather than assigning two separate budgets.

B.6.3 Token-Level Policy Update

For each response-token position $\ell \in \mathcal{U}_{q,i,t}$, we use the same joint advantage, $A_\ell = A_{q,i,t}$, in Eq. (8). The loss averages over the $N = \sum_{(q,i,t)} |\mathcal{U}_{q,i,t}|$ response tokens in the batch; context, feedback and diagnosis tokens receive no policy loss. The importance ratio is $\varrho_\ell = \pi_\theta(a_\ell \mid c_\ell)/\pi_{\mathrm{old}}(a_\ell \mid c_\ell)$; both π_{old} and the advantages are held fixed during the update. In the experiments, the clipped surrogate additionally uses dual clipping for negative advantages (Ye et al., 2020), and the loss includes the low-variance KL term $\lambda_{\mathrm{KL}} D^{\mathrm{LV}}_\ell$ relative to the frozen policy π_{ref} (Schulman, 2020). The clipping and KL settings are listed in Table 3.

B.7 Algorithm

Algorithm 1 organizes the RL stage into four steps: rollout and diagnosis, penalty allocation, policy update, and online price refitting. Its initial policy, taxonomy and prices come from the cold start in Appendices B.1 and B.3.4; hyper-parameters are listed in Table 3.

Implementation notes. The reported runs have the following differences from the algorithmic specification. With exactly two parsable diagnoses, the trajectory gate accepts a one-to-one tie; a finite fit with a false solver-success flag is logged but not rejected. The schedule strengths are read from the configuration after being set from the measured $\overline{T}^{(0)}$. Earlier development runs only warned when the penalty cap was exceeded; the specified algorithm enforces it.

Algorithm 1 FAULT: self-evolving RL with diagnosis-guided credit

Require: Diagnosis-SFT policy π_θ; frozen reference policy π_{ref}; task distribution $\mathcal{D}$
Require: Frozen taxonomy and pricing classes $\mathcal{C}$; initial fit $\widehat{\boldsymbol{\beta}}^{(0)}$ and prices $\mathbf{w}^{(1)}$
Require: Reward bounds $R_{\min}, R_{\max}$; training settings of Table 3
Ensure: Trained policy π_θ and price sequence $\{\mathbf{w}^{(u)}\}_{u=1}^{U+1}$

$\boldsymbol{\beta}^{(0)} \leftarrow \min(\widehat{\boldsymbol{\beta}}^{(0)}, \mathbf{0})$; $v \leftarrow 0$; $\mathcal{W} \leftarrow \emptyset$; $\Delta R \leftarrow R_{\max} - R_{\min}$
for $u = 1, \ldots, U$ **do**
  $\pi_{\text{old}} \leftarrow \pi_\theta$; $\mathbf{w} \leftarrow \mathbf{w}^{(u)}$ ▷ fixed throughout the current update
  1. Rollout and self-diagnosis
  Sample tasks from $\mathcal{D}$ and G trajectories per task using π_{old}; record outcomes and rewards
  Generate M diagnoses per trajectory with π_{old}; verify entries and apply the gate (App. B.2)
  Retain verified entries mapped to pricing classes in $\mathcal{V}_{q,i,t}$ for all trajectories
  For gate-passing trajectories, compute $n_{q,i,k}, \mathbf{x}_{q,i}, k^*_{q,i}$ and $S_{q,i}$ by Eqs. (9), (21)
  Impute the remaining scores from the batch's same-outcome, non-imputed scores (App. B.2)
  2. Bounded penalty allocation
  if $u = 1$ **then**
    Measure $\overline{T}^{(0)}$; fix λ_{warm} and λ_{peak} as specified in App. B.4
  end if
  Compute λ_u by Eq. (22); set $P_{q,i} \leftarrow \min(\lambda_u S_{q,i}, \eta \Delta R)$ for every trajectory
  Compute $h_{q,i,t} \leftarrow \sum_{e \in \mathcal{V}_{q,i,t}} w_{k(e)}$ and its boosted form $\widetilde{h}_{q,i,t}$ by Eq. (24)
  Allocate $P_{q,i}$ using Eq. (25), including its zero-weight fallback, to obtain $\delta_{q,i,t}$
  3. Dual-channel policy update
  Form the task groups Ω_q and context-based anchor groups $\mathcal{H}(o_{q,i,t})$ (App. B.6)
  Compute $Z^{\text{env}}_{q,i,t}$ from environment rewards, then $E_{q,i,t}$ and $\widetilde{Z}_{q,i,t}$ by Eq. (27)
  Centre both channels and combine $A_{q,i,t} \leftarrow A^{\text{ep}}_{q,i,t} + \omega A^{\text{step}}_{q,i,t}$ by Eq. (28)
  Broadcast $A_{q,i,t}$ to its response tokens and update θ using Eq. (8) (App. B.6)
  4. Price refitting for the next batch
  Append non-imputed records $(q, \mathbf{x}_{q,i}, y_{q,i})$ to $\mathcal{W}$; use $z_{q,i}$ for graded outcomes (App. B.3.2)
  Preserve rollout-group identities in $\mathcal{W}$ and retain only the last W updates
  $\mathbf{w}^{(u+1)} \leftarrow \mathbf{w}^{(u)}$ ▷ retain prices unless this refit is accepted
  if $u \geq u_0$, $(u - u_0) \bmod \Delta u = 0$, and the refit checks of App. B.3.5 hold **then**
    Set $\boldsymbol{\beta}_{\text{ref}} \leftarrow \boldsymbol{\beta}^{(0)}$ and $\rho \leftarrow \rho_u$; fit $\widehat{\boldsymbol{\beta}}(\boldsymbol{\beta}^{(0)})$ on the supported classes in $\mathcal{W}$
    Binary: Eq. (14), initialized at $\boldsymbol{\beta}^{(0)}$; graded: Eq. (16)
    if the current fit is accepted (App. B.3.5) **then**
      $v \leftarrow v + 1$; $\widehat{\boldsymbol{\beta}}^{(v)} \leftarrow \widehat{\boldsymbol{\beta}}(\boldsymbol{\beta}^{(0)})$; let $\mathcal{K}_v$ be the fitted classes
      Update $\boldsymbol{\beta}^{(v)}$ from $\boldsymbol{\beta}^{(v-1)}$ and $\widehat{\boldsymbol{\beta}}^{(v)}$ by the sign-gated EMA in Eq. (20)
      Set $\mathbf{w}^{(u+1)}$ from $\boldsymbol{\beta}^{(v)}$ using Eq. (17) with $\sigma_k = 1$
    end if
  end if
end for

C Additional Experimental Setup

This appendix expands the benchmarks, baselines, evaluation protocols and implementation of §3.1.

C.1 Benchmarks and Baselines

Benchmarks. ALFWorld comprises six families of household tasks (Pick, Look, Clean, Heat, Cool and Pick2); we train on its standard training games and evaluate on the unseen split. WebShop tests product search, inspection and purchase against natural-language requests. Search-based QA follows Search-R1 (Jin et al., 2025) and draws questions from NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022) and Bamboogle (Press et al., 2023).

Baselines. Vanilla uses the instruction-tuned backbone without task-specific post-training. GRPO and GiGPO start from that backbone; GiGPO follows its upstream recipe, including 150 updates on ALFWorld. SEED uses its own teacher-written hindsight-skill SFT data and skill-distillation objective. FAULT starts from diagnosis SFT, and the Qwen3-4B-Instruct-2507 GRPO (diag-SFT)

and GiGPO (diag-SFT) controls share this checkpoint with diagnostic shaping disabled but diagnosis monitoring retained.

C.2 Evaluation Protocols

All reported metrics are percentages, with higher values better; each result comes from one training run, so differences are point estimates rather than claims of statistical significance.

ALFWorld. The unseen split has 134 tasks: Pick 24, Look 18, Clean 31, Heat 23, Cool 21 and Pick2 17. Each task is evaluated three times by a single evaluation worker, with greedy decoding (temperature 0) and at most 30 active steps per evaluation. We average the three success indicators for each task, then average across all 134 tasks, so task families contribute in proportion to their size.

WebShop. We use the small catalogue of $1{,}000$ products and 500 held-out goals. Evaluation runs 8 batches of 32 goals (256 episodes) with greedy decoding, at most 15 steps and a two-step history window. Goals are sampled without replacement within a batch but may recur across batches, so the strict success rate counts successful episodes, not distinct goals. The task score is the environment's graded score, averaged over episodes and scaled to $[0, 100]$; the strict success rate is the fraction of episodes ending in a purchase of score one.

Search-based QA. The agent gathers evidence with a search tool before answering questions from the seven datasets above. We report exact-match accuracy for each dataset and their unweighted mean, rather than pooling all questions into one accuracy.

C.3 Cold-Start Data and Initialization

Each benchmark has its own frozen error taxonomy, diagnosis-SFT data and initial price fit, constructed as in Appendix B.1. We describe the data and benchmark-specific choices below; training hyper-parameters are collected in Appendix C.4.

C.3.1 ALFWorld

The following cold-start setup describes Qwen3-4B-Instruct-2507.

Data and teacher diagnosis. We sample eight rollouts for each of 180 training tasks from two actor checkpoints: the base model and a checkpoint after 20 GRPO updates. Each checkpoint contributes $1{,}440$ trajectories, giving $2{,}880$ in total. GPT-5.6 Sol diagnoses them at temperature 0.

Taxonomy construction. Using GPT-5.6 Sol for both induction passes, we construct a frozen taxonomy with eight L1 families and 25 L2 mechanisms. From the L2 mechanisms and eight L1 fallback categories, we retain classes with at least ten verified occurrences and frequency ranks in the top 60% of candidates, yielding $K = 20$ pricing classes.

Diagnosis-SFT data. We reserve 200 trajectories, stratified by checkpoint, outcome and task type, for assessment. The remaining $2{,}680$ examples, supplemented by a few action-replay examples from successful trajectories, train the diagnosis-SFT model with the settings in Table 3.

Initial price fit. The fit uses 99 informative groups of eight rollouts from the same task and checkpoint, or 792 trajectories, with the extended features of Appendix B.3.4. Cold-start feature scales σ_k are computed over all $2{,}880$ trajectories. The resulting prices are reported in Table 7.

C.3.2 WebShop

The recorded Qwen3-4B-Instruct-2507 cold-start collection samples 180 tasks and eight rollouts per task from each selected early checkpoint, producing up to $4{,}320$ trajectories. WebShop uses its own error taxonomy and diagnosis-SFT data. Its graded training reward uses the within-group ridge estimator of Appendix B.3.2. The two model sizes use different penalty strengths, listed in Table 3.

C.3.3 SEARCH-BASED QA

Search-based QA also uses a benchmark-specific taxonomy and diagnosis-SFT data. Its SFT data combine diagnosis examples with task responses, including successful solving trajectories from Qwen3.8-27B (Qwen Team, 2026a) that contain at least one search. The 1.7B SFT set contains 6,974 examples: 4,179 task-response examples and 2,795 diagnosis examples. The 4B set contains 6,498 examples: 3,699 task-response examples and 2,799 diagnosis examples.

C.4 TRAINING CONFIGURATION AND HYPER-PARAMETERS

Table 3 lists the six FAULT configurations in Table 1, separating shared settings from benchmark and model-size differences. The backbones are Qwen3-1.7B and Qwen3-4B-Instruct-2507 (Yang et al., 2025; Qwen Team, 2025); both use full-parameter diagnosis SFT, and the actor and diagnoser share weights throughout RL.

Diagnoser system prompt (`DIAG_SYSTEM`)

```
You are the trajectory diagnostician for an AI agent. You will read ONE complete
trajectory of the agent solving a task in {env_name}, together with the FINAL OUTCOME,
and identify the ERRORS the agent made, using ONLY the error table below.

Error table (use these ids only):
{table_text}

Rules:
1. Report at most {max_errors} errors, most important first. One entry per distinct
   error mechanism; a cascade after the first error counts once.
2. Every entry MUST cite the step number (1-based, exactly as shown as [Step k]) and a
   VERBATIM quote of at most {max_quote_words} words copied character-for-character from
   that step's text (the agent's own text or the environment feedback shown under that
   step). Never paraphrase, never quote from another step.
3. Use l1/l2 ids from the table. If you see a real error that fits no row, put it in
   "off_table" (label, step, quote, one-line mechanism) instead of forcing an id.
4. A SUCCESSFUL trajectory can still contain errors (e.g., wasted retries). A FAILED
   trajectory may have no identifiable table error other than running out of steps:
   do NOT invent errors.
5. severity_hint is your judgement (minor|major|fatal); it does not change the outcome
   you were told.
6. Output ONLY one JSON object, no prose, exactly this schema:
{schema_text}
```

Output schema (`DIAG_SCHEMA_TEXT`, injected as `{schema_text}`)

```
{"outcome_seen": "success" | "failure",
 "errors": [{"l1": "E1", "l2": "E1.2", "step": 7, "quote": "...", "severity_hint": "major"}],
 "dominant": {"l1": "E1", "step": 7} | null,
 "off_table": [{"label": "...", "step": 12, "quote": "...", "mechanism": "..."}]}
```

Figure 6: **The frozen-table diagnosis prompt.** Each reported error needs a frozen-table id (or `off_table`), a 1-based step and a verbatim quote. The user turn gives the task, a one-line environment description, the final outcome with its reward and the rendered trajectory; only the environment name and description and the table text depend on the benchmark.

Table 3: Key training and method hyper-parameters for the six FAULT runs. Shared values span model or benchmark columns.

Setting	ALFWorld		WebShop		Search-based QA	
	1.7B	4B	1.7B	4B	1.7B	4B
Supervised initialization						
SFT learning rate	10^{-5}					
SFT learning-rate schedule	10% warm-up; cosine decay					
SFT epochs / global batch	2 / 32		7 / 176		2 / 32	
Policy optimization and rollout generation						

Continued on next page

Table 3 (continued)

Setting	ALFWorld		WebShop		Search-based QA	
	1.7B	4B	1.7B	4B	1.7B	4B
RL learning rate / updates	10^{-6}; $U = 160$; one PPO epoch per update					
Policy clipping	Lower/upper clipping $0.2/0.2$; dual-clip parameter 3.0					
Rollout sampling	$G = 8$ trajectories per task; temperature 1.0, top-$p = 1$, no top-k filtering; thinking disabled					
Response limit (tokens)	512					
Tasks / trajectories per update	16 / 128		16 / 128		128 / 1,024	
PPO mini-batch	256		256		512	
RL learning-rate warm-up fraction	0		0		0.1	
KL coefficient	0.01		0.01		0.001	
Entropy coefficient	0.001		0.001		0	
Invalid-action penalty	0.1		0.1		0.01	
Environment reward	$\{0, 10\}$		$10\times$ matching score, in $[0, 10]$		$\{0, 1\}$	
Discount factor	$\gamma = 0.95$					
Step-channel weight ω	0.8		0.8		0.9	
Advantage normalization	Mean centring		Mean centring		Mean centring and standard-deviation scaling	
Anchor similarity threshold	0.95		0.95		0.9	
Self-diagnosis and penalty allocation						
Diagnosis sampling	$M = 3$ per trajectory; temperature 0.7, top-$p = 1$					
Maximum errors per diagnosis	5					
Diagnosis agreement threshold	≥ 2 parsable diagnoses; ≥ 2 votes per L1 hit					
Score features	L2 error rates		L2 error rates		L1 hit indicators	
First-error boost in score	$b = 1.5$		$b = 1.5$		Not used	
First-error boost in allocation	$b_{\mathrm{loc}} = 1.5$ for all six runs					
Rule-detector allocation weights	Repeated failed action: 1; invalid action: 0.5; action cycle: 1					
Warm-up strength λ_{warm}	28.156	26.5	8.141	12.37	0.2	0.2
Peak strength λ_{peak}	84.468	79.5	8.141	37.11	0.6	0.6
Penalty schedule	$U_{\mathrm{w}} = 10$ warm-up updates; cosine schedule with final strength 0 at update 160					
Budget cap $\eta\Delta R$ ($\eta = 0.9$)	9		9		0.9	
Initial error-price estimation						
Pricing classes K	24	20	32	18	10	8
Estimator	Conditional logit with Firth correction		Within-group ridge		Conditional logit with Firth correction	
Cold-start shrinkage ρ	0.5		0.003		0.5	
Length covariate in initial fit	$T_{q,i}/T_{\max}$		Included		None	
Initial price scaling	Feature standard deviation σ_k		Feature standard deviation σ_k		No feature scaling	
Coefficient bounds	$[-8, 8]$		Unbounded		$[-8, 8]$	
Online error-price estimation						
Refit window / schedule	$W = 50$ updates; first refit at $u_0 = 30$, then every $\Delta u = 10$ updates					
Refit support	At least 30 nonzero records per fitted class and 20 informative groups; binary fits also require total successes in these groups $\geq 10\times$ the number of fitted classes					
Online shrinkage	$\rho_u = 1$ through update 20; linear decay to 0.1 at update 100					
Coefficient EMA	$\nu = 0.7$ (old); $1 - \nu = 0.3$ (new)					
Online price scaling	$\sigma_k = 1$					

C.5 Diagnosis Prompt

The prompt sequence follows Appendix B.1: the teacher first names errors without a taxonomy, an induction prompt builds and freezes the two-level table, and the diagnosis prompt then uses that table to label trajectories. Only the last prompt runs during RL (Figure 6). A single rendering function supplies the same prompt format for diagnosis SFT, online self-diagnosis, acceptance tests and frozen-checkpoint evaluations. This prompt serves the role of the SEED analyzer (their Figure 10), using a learned, verifiable error table in place of free-text analysis. The teacher and induction prompts, one-line environment descriptions and actor prompts are released with our code; the ALFWorld actor prompt is taken unchanged from SEED's Figure 11.

D Additional Experimental Results

We examine signal coverage and strength, step-level credit allocation, diagnostic quality, and ablations of the design. Unless stated otherwise, results use ALFWorld training rollouts from one run per method; GRPO (diag-SFT) and GiGPO (diag-SFT) share FAULT's diagnosis-SFT initialization. Apart from the ablations, these analyses describe diagnosis and training behavior; task performance is reported in Tables 1 and 4.

D.1 Learning Signal in Same-Outcome Groups

We first define the measurements used in §3.3.1, then separate the algebraic effect of equal outcomes from the observed coverage.

D.1.1 Coverage and Relative Signal Strength

Groups and measured quantities. A group g contains $G = 8$ rollouts of one task at one update. Each run has 16 groups per update over 160 updates. Write $c(g) \in \{\mathrm{mix}, \mathrm{succ}, \mathrm{fail}\}$ for mixed, all-success and all-fail groups, respectively, and let N_c be the number of groups of type c, with $N = \sum_c N_c$. For coverage, the measured credit is

$$\kappa_s = \begin{cases} A_{q,i,t}, & \text{GRPO/GiGPO: step sample } s = (i,t), \\ S_{q,i}, & \text{FAULT: trajectory sample } s = i. \end{cases} \tag{30}$$

Here $A_{q,i,t}$ is the optimizer advantage, whereas $S_{q,i}$ is the diagnostic score of Appendix B.4, before multiplication by λ_u. Thus FAULT's coverage measures available diagnostic contrast, not the magnitude of its final policy advantage.

Coverage. Define the within-group range and the usable-signal indicator by

$$r_g = \max_{s \in g} \kappa_s - \min_{s \in g} \kappa_s, \qquad \mathrm{usable}(g) = \mathbb{1}[r_g > \tau \bar{r}_{\mathrm{mix}}], \tag{31}$$

where each method uses its own mixed-group mean:

$$\bar{r}_{\mathrm{mix}} = \frac{1}{N_{\mathrm{mix}}} \sum_{g:\, c(g)=\mathrm{mix}} r_g, \tag{32}$$

$$\mathrm{Coverage}(\tau) = \frac{1}{N} \sum_g \mathbb{1}[r_g > \tau \bar{r}_{\mathrm{mix}}]. \tag{33}$$

The measured runs have $N_{\mathrm{mix}} > 0$ and $\bar{r}_{\mathrm{mix}} > 0$. Mixed groups supply outcome contrast and set a within-method reference scale. We use $\tau = 0.05$; "usable" denotes a contrast above this threshold, not independently verified correctness or a guaranteed policy improvement.

Signal-retention Index. For signal strength, let r_g^{I} use the same range definition, but replace $S_{q,i}$ by the injected budget $P_{q,i}$ for FAULT; the baselines still use $A_{q,i,t}$. In a 20-update window w, let $N_{w,c}$ count groups of type c and let $\bar{r}^{\mathrm{I}}_{w,c}$ be their mean range. For a positive mixed-group mean, define

$$I_w = \sum_{c \in \{\mathrm{mix},\mathrm{succ},\mathrm{fail}\}} \frac{N_{w,c}}{N_w} \frac{\bar{r}^{\mathrm{I}}_{w,c}}{\bar{r}^{\mathrm{I}}_{w,\mathrm{mix}}}, \qquad N_w = \sum_c N_{w,c}. \tag{34}$$

An absent group type contributes zero. Each term combines a type's frequency with its contrast relative to mixed groups; the mixed contribution is exactly its frequency. Figure 3 normalizes each window separately. The run-pooled Index is recomputed from all groups, rather than averaged across windows. Because FAULT uses trajectory budgets and the baselines use step advantages, this is a comparison of contrast proxies, not a common-scale measure of optimizer credit. Redistribution and centring can change how budget differences enter the update, so the proxy alone does not provide an upper bound on advantage contrast.

D.1.2 Credit under Equal Terminal Rewards

ALFWorld gives terminal reward $R_{\max} = 10$ for success and $R_{\min} = 0$ for failure. The following result isolates terminal rewards and diagnostic penalties; it omits the shared invalid-action penalty and other reward terms (Appendices B.6 and F.1). For a fixed task q, use the step set Ω_q and anchor groups of Appendix B.6. For any non-empty comparison set $\mathcal{G} \subseteq \Omega_q$, write $\bar{\delta}_{\mathcal{G}} = |\mathcal{G}|^{-1} \sum_{(j,r)\in\mathcal{G}} \delta_{q,j,r}$. For $\mathcal{H} = \mathcal{H}(o_{q,i,t})$, define the centred discount factor

$$\psi_{q,i,t} = \gamma^{T_{q,i}-t} - \frac{1}{|\mathcal{H}|} \sum_{(j,r)\in\mathcal{H}} \gamma^{T_{q,j}-r}. \tag{35}$$

Proposition 2 (Credit in same-outcome groups)**.** *Fix a rollout group with purely terminal rewards $R^{\text{out}}_{q,i} = \bar{R}$ for all i, discount $0 < \gamma < 1$, and step-channel weight $\omega \geq 0$. Assume that the non-empty anchor groups form a fixed partition of Ω_q, and that normalization scales are finite and strictly positive. Omitting additional reward terms, the following hold.*

(i) GRPO and the GiGPO episode channel have zero outcome advantage:

$$A^{\text{GRPO}}_{q,i} = A^{\text{GiGPO,ep}}_{q,i} = 0. \tag{36}$$

(ii) Within an anchor group $\mathcal{H}$, GiGPO's step advantage is

$$A^{\text{GiGPO,step}}_{q,i,t} = \frac{\bar{R}}{\xi_{\mathcal{H}}}\, \psi_{q,i,t}, \qquad \xi_{\mathcal{H}} > 0, \tag{37}$$

where $\xi_{\mathcal{H}}$ is the group's common normalization scale. It vanishes everywhere when $\bar{R} = 0$. For $\bar{R} \neq 0$, it vanishes throughout $\mathcal{H}$ if and only if every member has the same remaining length $T_{q,j} - r$.

(iii) FAULT has the channel and joint advantages

$$\begin{aligned} A^{\text{ep}}_{q,i,t} &= -\big(\delta_{q,i,t} - \bar{\delta}_{\Omega_q}\big), \\ A^{\text{step}}_{q,i,t} &= \bar{R}\,\psi_{q,i,t} - \big(\delta_{q,i,t} - \bar{\delta}_{\mathcal{H}}\big), \\ A_{q,i,t} &= \omega\bar{R}\,\psi_{q,i,t} - \big(\delta_{q,i,t} - \bar{\delta}_{\Omega_q}\big) - \omega\big(\delta_{q,i,t} - \bar{\delta}_{\mathcal{H}}\big). \end{aligned} \tag{38}$$

When $\bar{R} = 0$, the joint advantage is zero at every sample if and only if $\delta_{q,i,t}$ is constant over Ω_q.

Proof. For (i), let $\mu_q = G^{-1}\sum_i R^{\text{out}}_{q,i}$ and $\sigma_q = \text{sd}(R^{\text{out}}_{q,1}, \ldots, R^{\text{out}}_{q,G})$. Equal rewards give $\mu_q = \bar{R}$ and $\sigma_q = 0$, hence

$$A^{\text{GRPO}}_{q,i} = \frac{R^{\text{out}}_{q,i} - \mu_q}{\sigma_q + \zeta} = \frac{\bar{R} - \bar{R}}{\zeta} = 0, \qquad \zeta > 0.$$

The GiGPO episode channel uses the same centred outcome. For (ii), purely terminal rewards give

$$Z_{q,i,t} = \sum_{r=t}^{T_{q,i}} \gamma^{r-t} R^{\text{env}}_{q,i,r} = \bar{R}\gamma^{T_{q,i}-t}, \qquad \frac{Z_{q,i,t} - \bar{Z}_{\mathcal{H}}}{\xi_{\mathcal{H}}} = \frac{\bar{R}}{\xi_{\mathcal{H}}}\psi_{q,i,t},$$

where $\bar{Z}_{\mathcal{H}}$ is the anchor-group mean. For $\bar{R} \neq 0$, all centred returns vanish exactly when all discount factors agree. Since $d \mapsto \gamma^d$ is strictly decreasing for $0 < \gamma < 1$, this is equivalent to equal remaining lengths.

For (iii), substituting the common reward into Eqs. (6)–(7) gives $E_{q,i,t} = \bar{R} - \delta_{q,i,t}$ and $\widetilde{Z}_{q,i,t} = \bar{R}\gamma^{T_{q,i}-t} - \delta_{q,i,t}$. Centring these scores over Ω_q and $\mathcal{H}$, respectively, and adding the channels gives Eq. (38). When $\bar{R} = 0$, write $m = \bar{\delta}_{\Omega_q}$ and $m_{\mathcal{H}} = \bar{\delta}_{\mathcal{H}}$, so

$$A_{q,i,t} = -(m_{\mathcal{H}} - m) - (1 + \omega)(\delta_{q,i,t} - m_{\mathcal{H}}).$$

If all advantages are zero, averaging within each anchor group gives $m_{\mathcal{H}} = m$; since $1 + \omega > 0$, every share then equals m. The converse follows by substitution. □

The result concerns advantage contrast, not semantic correctness or a guaranteed gradient improvement. A singleton anchor group has zero step-channel advantage for both GiGPO and FAULT, although FAULT's episode channel can remain non-zero. Equal outcomes do not force diagnostic contrast to vanish, but uniform shares do; different error labels need not yield different shares. In all-success groups, discount and diagnostic terms can also cancel.

D.1.3 Observed Coverage and Strength

Coverage and its source. At $\tau = 0.05$, every mixed group is usable for all three methods, so coverage differences come from same-outcome groups. GRPO's residual format contribution never exceeds the threshold in these groups. GiGPO can distinguish all-success samples at different remaining distances from a shared anchor context to success, favoring shorter paths. Its overall coverage falls below GRPO's near $\tau = 0.25$ and approaches its mixed-group share at larger thresholds. Its all-fail groups have no usable contrast. Under the diagnostic-score measurement, FAULT covers every all-fail group and 90% of all-success groups. Of the 125 all-success groups below threshold, 53 have zero scores in all eight rollouts. These are empirical coverage results, not guarantees implied by different diagnoses. Figure 7 expands Figure 1(a) over training windows; Figure 8(a) varies the threshold.

Available versus applied contrast. Replacing $S_{q,i}$ by $P_{q,i}$ in the coverage calculation reduces FAULT's run-wide coverage from 95.1% to 84.6%. The two measurements differ by at most two percentage points through the first 120 updates. In the last 20 updates, as λ_u approaches zero, only 20% of groups retain a budget range above threshold. Thus diagnosis can still distinguish trajectories while the applied penalty becomes small.

Interpreting the Index. Figure 8(b,c) gives the pooled Index and its same-outcome contributions; Figure 3 gives the windowed values. FAULT's pooled 0.821 is a budget-based proxy, subject to the measurement distinction above. For sensitivity only, hold group frequencies and same-outcome mean ranges fixed while multiplying the mixed-group mean range by a hypothetical factor $k \in [2, 3]$. The Index then becomes approximately $0.372 + 0.449/k \in [0.52, 0.60]$, still above the baselines' approximately 0.42. This is neither a measured advantage Index nor a proven bound.

D.2 Penalties on Successful Trajectories

Measurement. Success certifies the terminal outcome but does not exempt a trajectory from diagnostic penalties. Figure 9(a) separates three rates: verified-error reports and L1-majority reports are fractions of valid diagnoses, whereas the charge rate is the fraction of successful trajectories with $P > 0.001$ reward units. Verification checks the cited quote and step; it is not an independent judgment of semantic correctness.

Results and the budget cap. Across training, 61.85% of successes ($8{,}719/14{,}097$) receive a charge above this threshold, decreasing from 80.5% in the first window to 52.2% in the last. Failures have a higher median budget than successes in every window (Figure 9(b)). The cap of Eq. (23) preserves success–failure ordering in the diagnostic episode channel by Proposition 1. It binds on $56/20{,}480$ trajectories (0.27%, all successes) and never removes the entire diagnostic-score contrast of a complete group (Figure 9(c)).

D.3 Step-Level Localization: Measurement and Additional Views

This subsection defines the localization and concentration measures used in §3.3.2, Figures 1(b) and 4, and the additional views in Figure 10.

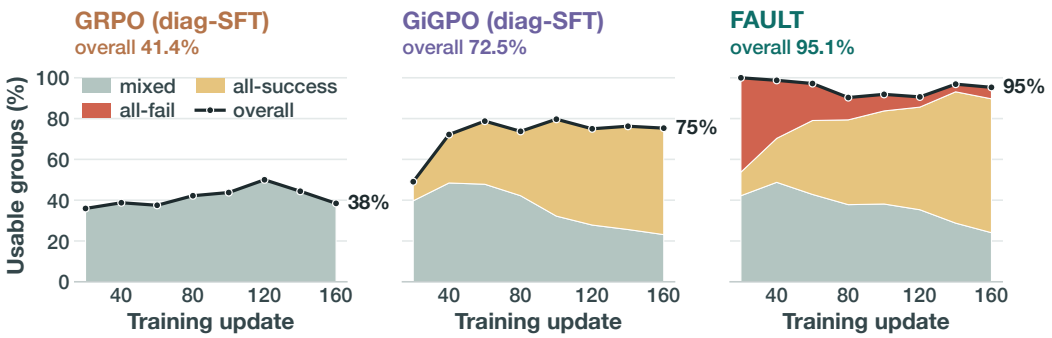


Figure 7: **Usable signal over training.** Stacked shares of all groups with a usable range at $\tau = 0.05$, measured in 20-update windows at their final update. The dark line is total coverage; titles report run-wide values. Same-outcome groups (red and gold) dominate and grow, with FAULT retaining broad coverage of them. FAULT uses scores S and the baselines use optimizer advantages. Single-seed runs; both baselines start from diagnosis SFT.

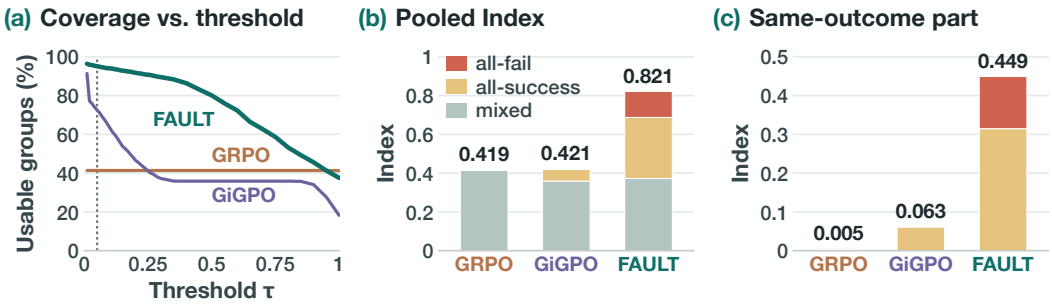


Figure 8: **Threshold sensitivity and pooled signal strength.** (a) Coverage as τ increases, using each method's mixed-group mean; the dotted line marks $\tau = 0.05$. (b) Pooled Index split by group type; the mixed contribution equals its group share. (c) Same-outcome contributions on a finer scale. Coverage uses S for FAULT, whereas its Index uses P; baselines use advantages for both. These measure contrast, not independently verified correctness.

D.3.1 Evaluation Protocol

Step penalties. For a trajectory of $T = T_{q,i}$ steps, define the non-negative credit assigned against step t as

$$b_t = \begin{cases} \delta_{q,i,t}, & \text{FAULT}, \\ \max(-A_{q,i,t}, 0), & \text{GRPO and GiGPO}, \end{cases} \tag{39}$$

where $A_{q,i,t}$ is the optimizer advantage. GRPO broadcasts its outcome advantage across steps; the additional invalid-action penalty makes b_t constant or two-valued.

Blind reference labels. We use 300 failed training trajectories of FAULT, 300 from a same-configuration GiGPO rerun, and 48 from GRPO, sampled evenly across eight 20-update windows and from both all-fail and mixed groups. Every failure reaches the $T = 30$ step limit. An independent LLM, Qwen3.8-Max (Qwen Team, 2026b), sees only the action and feedback text, without method names or penalties, and selects the earliest decisive step j whose correction it judges most likely to turn failure into success. These are model judgments, not human or counterfactually verified labels. A blind relabel of 48 trajectories agrees within one step in 75% of cases.

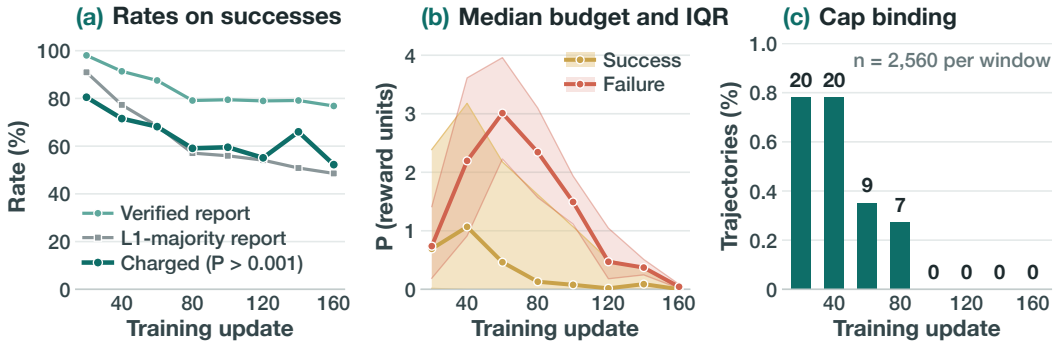


Figure 9: **Diagnostic penalties on successful trajectories.** Each point summarizes 20 updates. (a) Verified and L1-majority report rates among valid diagnoses, and charge rate ($P > 0.001$) among all successes. (b) Median budget P and interquartile range for successes and failures. (c) Fraction of the 2,560 trajectories per window reaching the cap; labels give counts. Windows are placed at their final update. Diagnoser and prices evolve during training; single seed.

D.3.2 Localization Metrics and Results

Rank steps by decreasing b_t, breaking ties uniformly at random. Allowing one step of error around the judge's label, define

$$\mathcal{J} = \{t \in \{1, \dots, T\} : |t - j| \le 1\}, \qquad \text{rank}^\star = \min_{t \in \mathcal{J}} \text{rank}(t). \tag{40}$$

The ranking and signed-offset metrics are

$$\text{MRR} = \mathbb{E}\left[\frac{1}{\text{rank}^\star}\right], \qquad \text{hit@}k = \Pr(\text{rank}^\star \le k), \qquad \text{off} = \min\left(\arg\max_t b_t\right) - j. \tag{41}$$

MRR and hit@k average over trajectories and 20,000 random tie-breaks per trajectory; the offset instead uses the first maximizer. Positive offsets place the maximum penalty after the judged step. Constant b_t gives the uniform random-ranking reference.

Localization results. FAULT achieves MRR 0.494, hit@1 34.1%, and 82 exact step matches; GiGPO achieves 0.303, 11.9%, and 21 exact matches. On its 48 trajectories, GRPO achieves MRR 0.306 and hit@1 15.3%, compared with 0.266 and 9.9% for random ranking. Figure 1(b) places GRPO at the random reference for its uniform outcome credit; the measured value includes the invalid-action penalty, which sometimes marks the decisive step. In 73% of GRPO trajectories, the maximum penalty is tied, so its offset distribution is omitted from Figure 4(a). GiGPO's median offset is +5 steps. Figure 10(a) extends hit@k to $k = 1, \dots, 12$: FAULT leads most at small k, while the curves approach each other as k grows. Even random nomination of 10 out of 30 steps covers the tolerance window about 72% of the time.

D.3.3 Concentration of Step Credit

For FAULT, set $c_t = \delta_{q,i,t}$; for the baselines, set $c_t = A_{q,i,t}$. For $T \ge 2$ and $\sum_t |c_t| > 0$, define

$$w_t = \frac{|c_t|}{\sum_{r=1}^{T} |c_r|}, \qquad H^* = -\frac{1}{\ln T} \sum_{t: w_t > 0} w_t \ln w_t, \qquad \text{ENS} = \left(\sum_{t=1}^{T} w_t^2\right)^{-1}. \tag{42}$$

The normalized entropy $H^* \in [0, 1]$ is zero for concentration on one step and one for a uniform allocation. The effective number of steps, ENS, is the inverse Herfindahl index: it equals 1 and T in these two cases. For FAULT these statistics describe the diagnostic allocation, not its full advantage, which also depends on returns and comparison-group baselines; the baseline statistics use absolute optimizer advantages.

Excluded trajectories and training trends. Zero-total-credit trajectories have no normalized distribution and are excluded. They account for 31.4% of FAULT's same-outcome trajectories, mainly

because no verified error with a non-zero price was found, versus 7.9% for GRPO and 0.1% for GiGPO. In same-outcome groups, GRPO has at most two step-advantage values separated by 0.1, reflecting the invalid-action penalty. Figure 10(b) tracks mean ENS in all-fail groups, where $T = 30$ and a uniform allocation has ENS 30. Over training, ENS falls from 26.5 to 15.6 for GRPO, 24.0 to 11.3 for GiGPO, and 11.3 to 3.3 for FAULT. The decline for FAULT partly reflects fewer diagnosed errors as penalized behaviors recede (Appendix E.2); concentration alone does not establish more accurate localization.

D.4 Penalties on Wasted Steps

Step classes and measurement. We use two mechanically identified classes: invalid or missing actions, and format-valid no-ops whose next feedback is "Nothing happens." Invalid actions take precedence; the last step cannot be labeled a no-op without a subsequent observation. All failed trajectories reach the 30-step limit, so these steps consume the interaction budget without recorded progress. The labels identify wasted interactions, not whether correcting one step would have changed the outcome. For the steps $\mathcal{B}$ of one class in a trajectory, its targeting ratio is

$$\frac{\sum_{t\in\mathcal{B}} b_t / \sum_{t=1}^{T} b_t}{|\mathcal{B}|/T},$$

using Eq. (39). Ratios and top-1 rates use failed trajectories with positive total negative credit and class counts strictly between 0 and T; eligibility therefore depends on the method and class. We average the ratio equally over these trajectories, with one denoting uniform allocation. Top-1 hits average uniformly over tied maximizers of b_t.

Invalid actions. Among eligible trajectories, both baselines achieve 100% top-1 targeting of invalid actions in all-fail groups; over the pooled all-fail and mixed-group sample, GRPO reaches 100% compared with 48.6% for FAULT. The shared invalid-action penalty directly marks these steps, so this result alone does not demonstrate diagnostic understanding. In the recorded recipe, an invalid action receives -0.1 and a valid action receives zero before advantage construction (Appendix F.1).

Format-valid no-ops. FAULT assigns $3.67\times$ the uniform share of negative credit to no-ops, compared with $0.48\times$ for GRPO and $0.57\times$ for GiGPO (Figure 11). The baselines' no-op top-1 hit rates are 0.1% and 4.3%, below their uniform references of 7.2% and 7.7%. In the analysed all-fail groups, their no-op negative credit is zero, although a valid step can still have a non-negative advantage after centring. Thus FAULT targets a class of wasted steps not directly flagged by the invalid-action reward; the contrast goes beyond locating format violations.

D.5 Diagnostic Quality Before and During RL

Assessment protocol. We assess teacher diagnoses and taxonomy coverage on the $2{,}880$ ALFWorld cold-start trajectories collected with Qwen3-4B-Instruct-2507, and evaluate the diagnosis-SFT checkpoint on the 200 stratified held-out trajectories (Appendix C.3.1). Separately, we test for degradation during RL as λ anneals using 17 frozen checkpoints at updates 0–160 and $n{=}600$ diagnoses per checkpoint. This assessment uses single-seed training rollouts and a fixed reference from the original teacher's diagnoses, unchanged throughout training.

Cold-start results. Evidence verification retains $13{,}820$ of $14{,}484$ teacher error mentions (95.4%); every rejected mention fails the quote check. Successful and failed trajectories contain an average of 3.4 and 5.1 verified errors, respectively, showing that the teacher reports process errors even in successful trajectories. The frozen taxonomy covers 90.6% of accepted mentions. On the held-out set, the initial self-diagnoser achieves L1 precision 0.81 and recall 0.56, above the respective thresholds of 0.75 and 0.50. Dominant-error step accuracy is 0.48 under a ±2-step tolerance, below the 0.60 threshold, and the off-table rate is 0.31, above the 0.10 limit. Only precision and recall meet their criteria; step localization and off-table predictions remain limitations.

Results during RL. L1 error-slot recall increases from 0.66 to 0.71 (Spearman $+0.93$), while the off-table hallucination rate decreases from 0.32 to 0.16 (Spearman -0.94). Both metrics are strongest in the late $\lambda \to 0$ window (updates 100–160). L2 micro-F1 remains stable, and L2 precision and recall do not decline through this window (Figure 12); L2 macro-F1 decreases slightly (-0.015 over

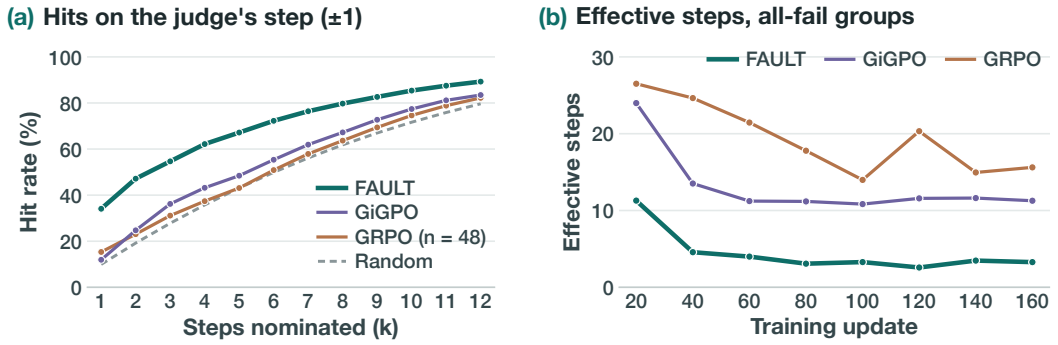


Figure 10: **Additional localization and concentration results.** (a) Hit@k within one step of the blind judge's label, using the trajectories of Figure 4(a) and 48 GRPO trajectories; the dashed curve is random ranking. (b) Mean ENS on all-fail groups in 20-update windows, placed at their final update. Both baselines start from diagnosis SFT.

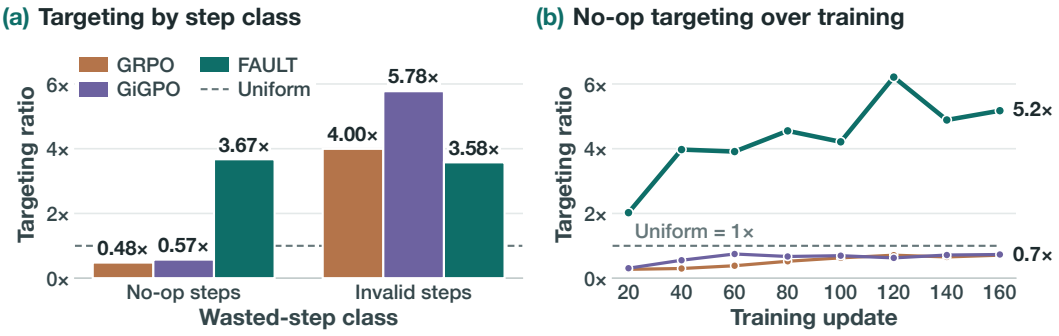


Figure 11: **Negative-credit allocation to wasted steps.** (a) Mean targeting ratio for invalid actions and format-valid no-ops under shared mechanical labels; $1\times$ is uniform allocation. (b) No-op targeting over training. Credit is δ_t for FAULT and $\max(-A_t, 0)$ for the baselines. All analysed failures reach the step limit. Baselines: GRPO (diag-SFT) and GiGPO (diag-SFT); offline analysis of single-seed runs.

the run). Macro-F1 weights classes equally and is more sensitive to rare classes; stable micro-F1 therefore does not rule out class-specific degradation.

D.6 Ablation Studies and Cold-Start Data Scale

Ablation variants. We compare FAULT with four design variants on ALFWorld with Qwen3-4B-Instruct-2507 (Table 4); several differ in more than their named components. *W/o online pricing* keeps cold-start prices fixed throughout RL while still fitting monitoring coefficients, and removes the cap in Eq. (23). *Cosine λ → constant λ* fixes the penalty strength instead of letting it rise from a low value to a peak and decay to zero. *Shared diagnoser → fixed judge* takes diagnoses from the frozen diagnosis-SFT checkpoint; this recorded variant also uses constant λ, online prices and the cap, so it does not isolate the judge. *GRPO + diagnostic penalty* is an earlier design that broadcasts a trajectory-level diagnostic penalty to every step rather than targeting diagnosed steps; it also changes score mixing and penalty scale, and experienced a refit failure. The variant without online pricing and the full method come from later runs than the other variants, so these runs are not matched controls.

Ablation results. With actor, diagnoser, and prices co-evolving, FAULT achieves the best overall success (91.0%), above fixed prices/no cap (89.6%), constant λ (79.1%), fixed judge (82.8%), and earlier GRPO-based diagnostic shaping (77.6%). These comparisons favor the full design but assess recorded design variants rather than isolating localization, pricing or the judge one factor at a time.

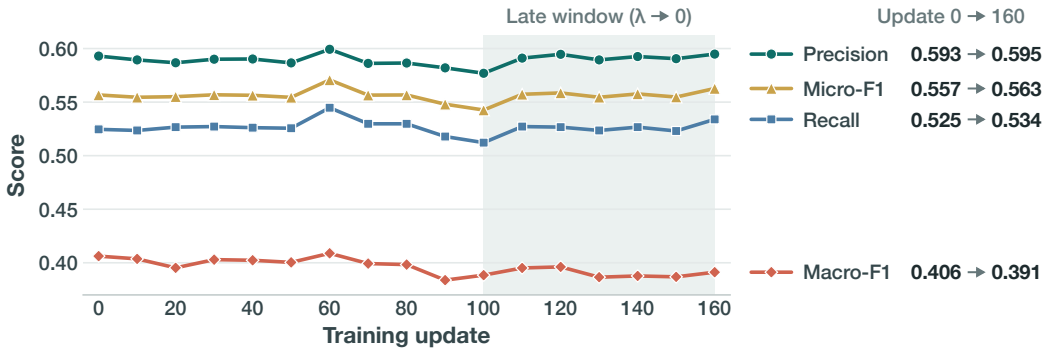


Figure 12: **Diagnostic quality during RL.** Concrete-L2 precision, recall, micro-F1 and macro-F1 against the fixed teacher reference. Shading marks the late $\lambda \to 0$ window (updates 100–160). Single seed; training rollouts.

Table 4: Ablations and design variants on ALFWorld with Qwen3-4B-Instruct-2507: success rate (%) on the 134 unseen tasks, evaluated as in Table 1.

	ALFWorld						
Variant	Pick	Look	Clean	Heat	Cool	Pick2	All
FAULT	**91.7**	**100.0**	**93.5**	87.0	85.7	88.2	**91.0**
w/o online pricing	79.2	**100.0**	83.9	**95.7**	**90.5**	**94.1**	89.6
Cosine $\lambda \to$ constant λ	70.8	**100.0**	77.4	73.9	85.7	70.6	79.1
Shared diagnoser $\to$ fixed judge	**91.7**	94.4	77.4	78.3	81.0	76.5	82.8
GRPO + diagnostic penalty	79.2	**100.0**	71.0	87.0	85.7	41.2	77.6

Cold-start scale protocol. Table 5 compares three cold-start pool sizes with Qwen3-4B-Instruct-2507. The ALFWorld pools are nested, adding 1,440 trajectories each from the base model, a checkpoint after 20 GRPO updates and a checkpoint after 40 updates; pool size and policy composition therefore vary together. Diagnosis F1 is evaluated on 600 trajectories from 25 unseen ALFWorld tasks using the same 17-class reference set. The number of non-zero prices counts pricing classes with a negative cold-start coefficient. Pre-RL success is measured with sampled decoding at temperature 0.4; the raw backbone scores 17.9. The ALFWorld RL runs are evaluated after 160 updates using an earlier protocol: 134 draws sampled with replacement from the 134 unseen tasks, with greedy decoding for each draw. Their success rates therefore count successful draws, rather than distinct tasks. The three WebShop runs use the earlier binary-reward recipe and report post-RL task score and strict success rate.

Table 5: Cold-start data scale with Qwen3-4B-Instruct-2507. ALFWorld reports named-class diagnosis F1 of the SFT model, non-zero initial prices, pre-RL SFT success rate and success rate after 160 RL updates. WebShop reports post-RL task score and strict success rate. Evaluation protocols are specified above.

	ALFWorld				WebShop	
Cold-start pool	Diag. F1	Non-zero prices	SFT SR	RL SR (draws)	Score	Succ.
1,440	0.617	6/20	24.6	74.6	73.8	60.2
2,880	0.648	8/20	31.3	82.1	85.6	67.6
4,320	0.668	3/20	30.6	83.6	82.3	70.3

E TRAINING DYNAMICS

We examine training trajectory length, the frequency of penalized behaviors, and the evolution of error prices on ALFWorld. These analyses use training rollouts and fitted coefficients, rather than additional test evaluations. The frozen-price variant is the w/o online pricing variant in Table 4.

E.1 TRAINING TRAJECTORY LENGTH

Measurement. Trajectory length $T_{q,i}$ counts active environment interactions, including invalid actions but excluding padding; it measures neither token count nor elapsed time. Figure 5 reports the mean over successful and failed trajectories at each update; Table 6 averages these update means over the final 5, 10, 20 or 40 updates. The four runs are raw-initialized GRPO, SEED, the frozen-price variant and the full method, all observed through update 160 with a 30-step horizon. Equal update counts do not imply equal numbers of interactions or matched training configurations.

Table 6: Mean training trajectory length over four windows ending at update 160, the last update shared by all runs. Each value equally averages the per-update mean number of active interactions. All runs use a 30-step horizon; these are single-run training summaries, not test-time estimates or confidence intervals.

Updates	GRPO raw-init	SEED	FAULT	w/o online pricing
156–160	21.89	14.54	11.43	12.03
151–160	21.69	13.77	12.03	11.38
141–160	22.09	14.34	12.56	11.70
121–160	22.21	14.38	12.64	12.00

Results across windows. The full method produces shorter trajectories than GRPO and SEED in all four windows. The frozen-price variant, which also removes the cap, is shorter than the full method by about 0.6–0.9 steps in three windows; the ordering reverses by about 0.6 steps in the final five updates. It has the shortest mean over updates 151–160, but it solves fewer tasks in the separate greedy evaluation (120 versus 122 of 134). Thus, shorter training trajectories need not correspond to higher task success. These single-run comparisons show sensitivity to the summary window, not statistical significance or the effect of online pricing alone.

Interpreting mean length. A lower mean can reflect shorter successful paths or more successes replacing horizon-truncated failures. These aggregate summaries do not separate lengths by outcome, so they cannot distinguish the two effects. The curves therefore describe training behavior without establishing that diagnostic redistribution alone improves path efficiency.

E.2 PENALIZED BEHAVIORS OVER TRAINING

Measurement. We track invalid or missing actions and format-valid no-ops using the mechanical labels of Appendix D.4. In each 20-update window, Figure 13(a) aggregates FAULT's diagnostic penalty mass by class, while panels (b,c) divide each class's step count by the total recorded step count. The comparisons include FAULT, GRPO (diag-SFT) with the same initialization, and SEED as a separate reference; GiGPO lacks the rollout text needed for this analysis.

Observed changes. The combined share of FAULT's penalty mass on invalid actions and no-ops falls from 91% to 18% as these behaviors become less frequent. Invalid actions, which receive a shared format penalty, decline under all three methods. Format-valid no-ops are not directly identified by this penalty: their rate falls from 3.0% to 1.7% under FAULT, but rises from 2.6% to 3.8% under GRPO and from 2.6% to 4.8% under SEED. Together with the allocation results in Appendix D.4, this pattern is consistent with diagnostic penalties targeting behaviors beyond invalid output. The comparison is observational: panel (a) measures only FAULT's penalty allocation, and these single-seed trends do not isolate a causal effect of the penalties.

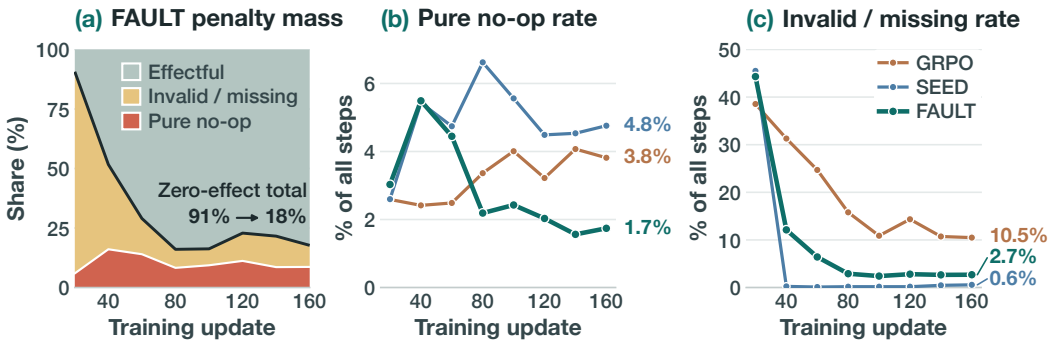


Figure 13: **Penalty allocation and behavior frequencies during training.** Points summarize 20-update windows and are placed at the final update. (a) Share of FAULT's diagnostic penalty mass assigned to each mechanical step class. (b) Format-valid no-op rate. (c) Invalid or missing action rate. Rates use all recorded steps as the denominator; the remaining-step category in (a), labeled effectful, includes steps without subsequent feedback. GRPO starts from diagnosis SFT; SEED is a reference with a different training mechanism. GiGPO is omitted because its original logs lack rollout text. Single-seed observational comparisons.

E.3 Initial Prices and Online Price Dynamics

Cold-start prices. Table 7 lists the initial prices for ALFWorld with Qwen3-4B-Instruct-2507 (Appendix C.3.1). Eight of the 20 error coefficients are negative and receive non-zero prices; the other twelve remain in the model with zero price. Success is shorter in 1,085 of 1,107 within-task success–failure pairs and never longer. The length coefficient reaches its lower bound, so initial error coefficients should be interpreted in light of this strong outcome–length association.

Table 7: Cold-start prices on ALFWorld with Qwen3-4B-Instruct-2507. $\widetilde{w}_k = \max(-\beta_k, 0)\,\sigma_k$ and $w_k = \widetilde{w}_k / \max_j \widetilde{w}_j$ (Eq. (17)). The remaining 12 pricing classes have $\beta_k > 0$ and $w_k = 0$.

Pricing class	β_k	$\sigma_k = \mathrm{sd}(x_{\cdot,k})$	$\widetilde{w}_k$	w_k
E7/_other (non-executable output, other)	−0.10791	0.017713	1.9115×10^{-3}	1.0000
E1.state_requirement_misread	−0.07593	0.015166	1.1515×10^{-3}	0.6024
E2.failure_cause_misattribution	−0.06719	0.013516	9.0816×10^{-4}	0.4751
E6/_other (premature abandonment, other)	−0.04978	0.012275	6.1109×10^{-4}	0.3197
E3.redundant_state_toggle_repeat	−0.02184	0.010325	2.2551×10^{-4}	0.1180
E1.state_history_loss	−0.01125	0.008723	9.8129×10^{-5}	0.0513
E2.absence_as_evidence_inference	−0.00459	0.013469	6.1783×10^{-5}	0.0323
E1.command_semantics_misbelief	−0.00045	0.025040	1.1284×10^{-5}	0.0059

Tracking and deployment. We track coefficients in GRPO (diag-SFT), GiGPO (diag-SFT), the frozen-price variant and the full method, with GRPO + step penalty as a fifth reference in Figure 16. The tracker refits every ten updates from 30 to 160, yielding 14 snapshots per run, and applies the sign-gated moving average in Eq. (20). For the two baselines, the tracker only monitors coefficients; the frozen-price variant records new fits but continues to charge the initial prices without the cap. The full method deploys its updated prices, while the fifth reference adds explicit step penalties to GRPO (diag-SFT).

Class-specific changes. Figure 14 shows three illustrative classes selected after inspecting the runs. Search-state amnesia treats previously checked locations as unexplored; remote-interaction misbelief attempts to inspect or manipulate an object before reaching it; E7 residual groups unnamed errors within one broad family. The full method's amnesia coefficient becomes more negative, from -0.295 at update 30 to -5.976 at 160; the other two end at -1.034 and -1.571. Under GiGPO (diag-SFT), it declines until update 120 and then partially recovers, showing that fitted associations also change without deploying prices.

Figure 16 reports all 20 classes, including five that remain unchanged across runs. Flat non-zero curves can reflect initial values retained by the sign gate, while large negative values can reflect fits approaching the coefficient bound; neither pattern establishes calibration quality.

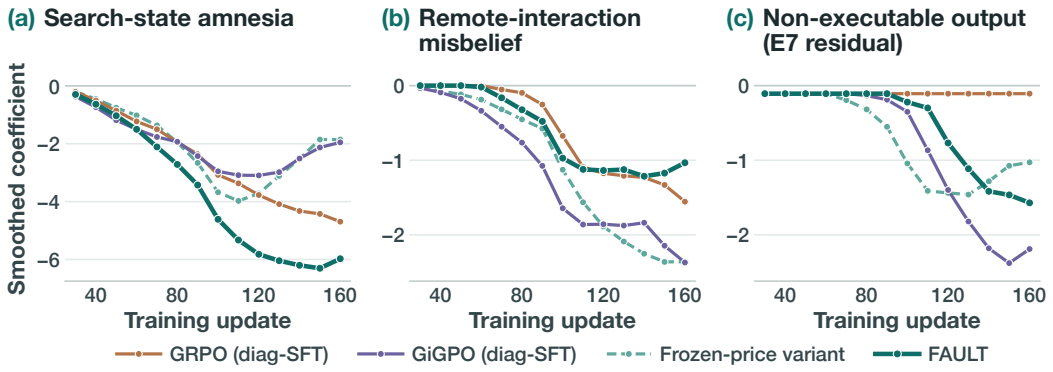


Figure 14: **Coefficient dynamics for three illustrative error classes.** Sign-gated moving averages at 14 refits per run; panels use different vertical scales. GRPO and GiGPO (diag-SFT) track coefficients without applying them, the frozen-price variant retains its initial prices, and the full method deploys updated prices. E7 residual collects unnamed errors within the non-executable-output family.

Revising the initial price allocation. Search-state amnesia has a zero cold-start price because its initial coefficient is positive ($+0.18$), yet it is the most frequent pricing class in every tracker window of every run and receives the most negative coefficient at the first online refit. At update 30, it occurs in 70% of the full method's tracker-window trajectories. More generally, the rank agreement between initial prices and online weights declines across runs; for the frozen-price variant, it drops from 0.63 at update 30 to -0.07 at 160 (Figure 15(a)). The comparison also reflects different price mappings: cold-start prices rescale coefficients by feature standard deviations, whereas online prices use unit scaling (Appendix B.3). The rankings thus describe changes in relative weights, not accuracy against known error costs.

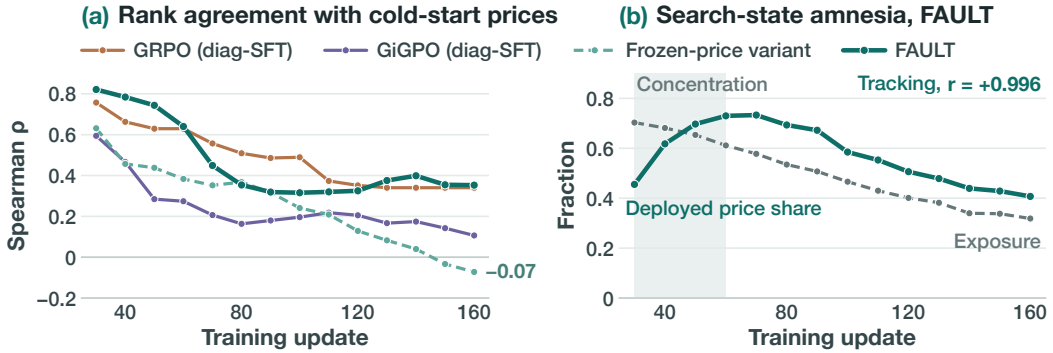


Figure 15: **Changes in price rankings and relative price allocation.** (a) Spearman correlation between cold-start prices and each run's online weights. Only the full method deploys the updated weights; the other curves are monitoring estimates. (b) Search-state amnesia under the full method: its share of total deployed class weight (solid) and its occurrence rate among tracker-window trajectories (dashed). Shading marks updates 30–60. The two series have Pearson correlation $+0.996$ over updates 70–160; the class's own normalized weight remains 1. Each curve represents 14 refits from one run.

Amnesia's share of total deployed class weight rises from 0.46 at update 30 to 0.73 at 70, then declines alongside its occurrence rate (Pearson $+0.996$ over updates 70–160; Figure 15(b)). Its own normalized weight remains 1 throughout, so the falling share reflects increased weight on other classes, not a reduction in its own price. From update 30 to 160, its occurrence rate falls by 55%

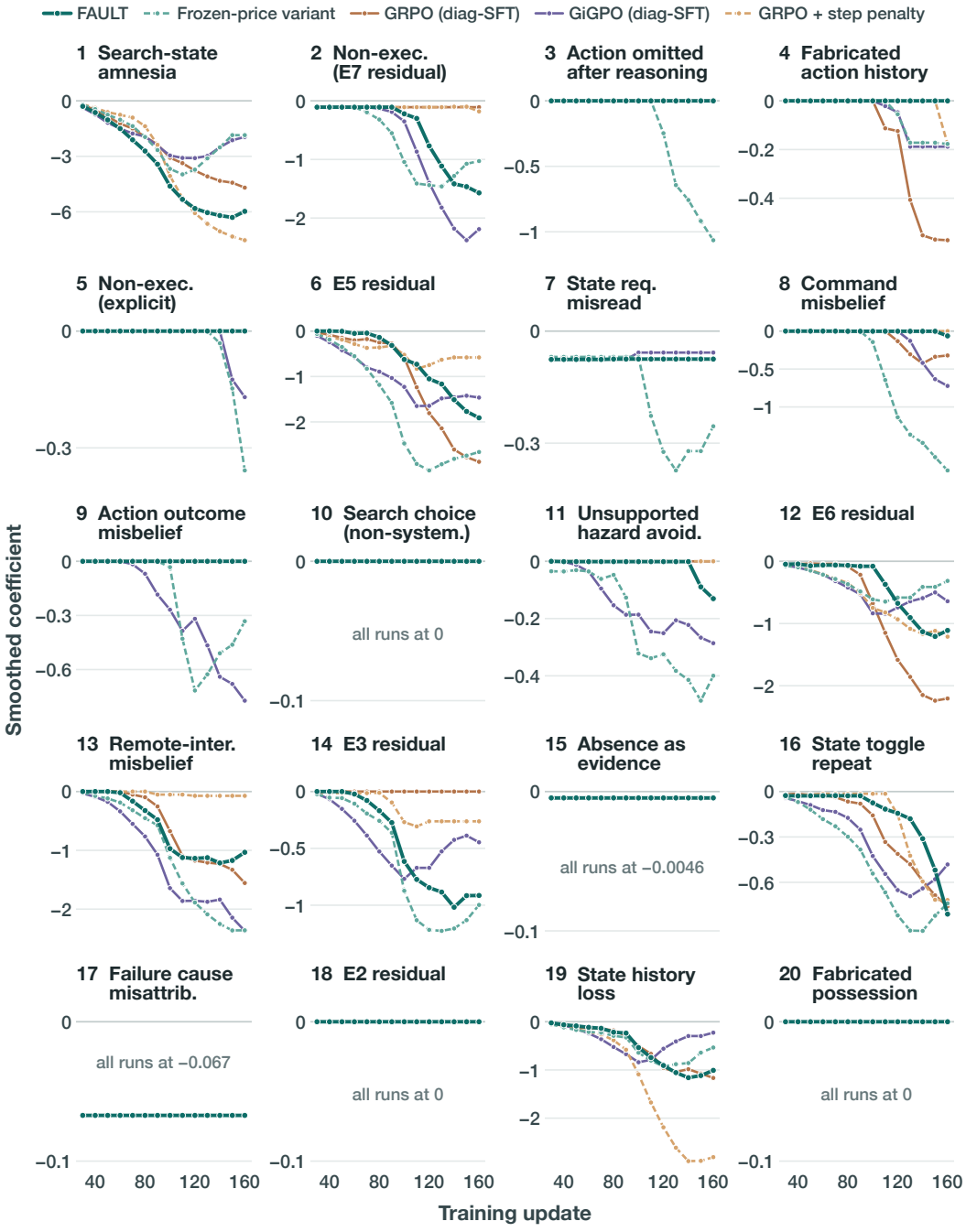


Figure 16: **Coefficient dynamics for all** 20 **pricing classes.** Sign-gated moving averages at 14 refits for five runs, with a separate vertical scale per panel. Coincident curves may be hidden by the full-method curve drawn last; several runs remain at zero in panels 3, 5 and 9. Residual classes collect errors not assigned to a named L2 class within an L1 family. Curves show estimator coefficients, not deployed prices.

under the full method, compared with 34% under GRPO (diag-SFT). By update 160, classes priced at zero initially account for 68% of the full method's deployed class-weight mass. These changes explain what online pricing adds to the initial allocation; their effect on policy performance is not isolated by these curves or the ablation configurations (Appendix D.6).

Interpretation. Each curve represents one run, and adjacent refits reuse data from overlapping 50-update windows. The coefficients describe associations that change with the policy, diagnoses and available samples, rather than causal error costs. Raw fits are provided with the code.

F Case Studies

We present two hand-selected ALFWorld training rollouts of FAULT, supplemented by baseline credit examples. These single-seed cases illustrate the mechanisms; population statistics are reported in Appendices D and E. Step numbers follow the zero-based training logs and judge labels: step 8 here corresponds to $t = 9$ in §2.1. As in Appendix D.3, we compare FAULT's diagnostic allocation δ_t with baseline negative optimizer advantages $\max(-A_t, 0)$; δ_t is not FAULT's full optimizer advantage. The blind judge is an independent LLM that sees only actions and feedback; its labels are model judgments, not independently verified ground truth.

F.1 Credit in an All-Fail Group

Selection and group. Among the 300 failed FAULT trajectories in Figure 4(a), 82 have their maximum diagnostic allocation at the judge's decisive step. From these matches, we select the most concentrated all-fail case without invalid actions or no-ops. Its task is *clean some lettuce and put it in sidetable* (update 70). All eight rollouts fail at the 30-step horizon, giving zero outcome-relative advantage, while their diagnostic budgets span 0.93–6.13 reward units (Figure 17(a)). The shared invalid-action reward can still provide a separate signal in all-fail groups.

Diagnostic allocation. In the selected trajectory ($P = 3.18$), step 8 returns to the fridge after a five-location sweep, marking the first revisit of a searched location; the agent subsequently cycles through searched locations and never finds the lettuce (Table 8). Approximately 82% of the budget falls on step 8 and 18% on the cabinet revisit at step 14 (Figure 17(b)). The blind judge also selects step 8; its verdict is reproduced in the table. Because this trajectory has no invalid actions or no-ops, its localization is not explained by mechanical flags. The group remains at $0/8$ success: the case demonstrates budget contrast and agreement with the judge within an unsuccessful group.

Baseline comparison. Figure 18 includes independently sampled baseline rollouts from different task instances; no paired same-instance capture is available. In the GRPO (diag-SFT) trajectory, step 13 revisits the fridge opened at step 4, and the blind judge identifies step 13 as decisive. Within its all-fail group, outcome-relative advantage is zero, but format shaping yields two per-step advantage values, with invalid steps 0.1 below valid steps. Negative advantage consists of three equal 0.087 spikes on unparseable steps and is zero at the judge's step. Rescaling these values cannot distinguish that step from other format-valid steps. The bottom strip shows an all-fail GiGPO (diag-SFT) trajectory whose negative advantage also falls only on format-flagged steps; no transcript is available because the original run did not store rollout text. These examples illustrate differences in the recorded credit assignments, without establishing a paired performance comparison.

F.2 Recovery in a Successful Trajectory

Errors and recovery. The second case is a 19-step success at update 30 on the task *put a hot tomato in sidetable* (Figures 19 and 20). Three premature heating attempts return "Nothing happens.": two occur before reaching the microwave and one after closing its door. The agent later assumes the tomato is hot, but the examination at step 15 returns "This is a cold tomato" in the next observation. It then revises its belief at step 16, heats the tomato successfully and completes delivery. Of the diagnostic budget $P = 5.52$, approximately 89% falls on the three ineffective attempts, but 0.56 also falls on the corrective examination, showing that localization is imperfect. A clean eight-step success in the same group has $P = 0$. The contrast illustrates how successful outcomes can coexist with different process errors and penalties.

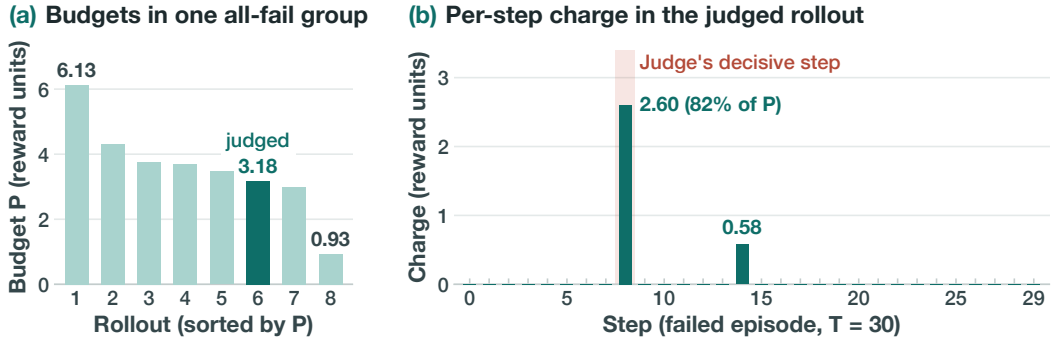


Figure 17: **Diagnostic penalties within an all-fail group.** (a) Budgets for eight rollouts, all failing at 30 steps. (b) Allocation for the selected trajectory: approximately 82% at the judge's decisive step and 18% at a later revisit (Table 8). The group contributes to Figure 7; the case is selected for illustration.

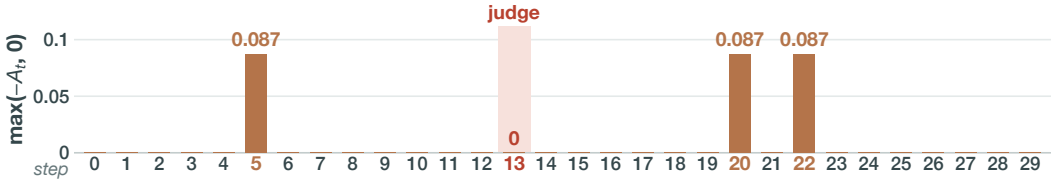
Table 8: Walkthrough of the selected trajectory in Figure 17(b), abridged from the raw episode. Values of δ_t are rounded shares of the diagnostic budget $P{=}3.18$; the two largest allocations fall on revisits of searched locations.

Step	Action	Environment feedback (abridged)	δ_t
0–1	go to / open fridge 1	fridge opened: a cup and a potato, no lettuce	0
2–7	first sweep	countertop 1; cabinets 2, 3 (opened, both empty); sinkbasin 1: four further locations, each visited for the first time	0
8	**go to fridge 1**	**first revisit of a searched location; contents unchanged since step 1**	**2.60 (82%)**
9–13	search starts to cycle	countertop 1 and cabinet 2 revisited; cabinet 4 and diningtable 1 new	0
14	**go to cabinet 3**	**revisit of a cabinet known to be empty since step 6**	**0.58 (18%)**
15–28	cycling continues	nine more steps on already-searched locations, four on new ones (diningtable 2, cabinet 1, sidetable 1), one `inventory` check ("not carrying anything")	0
29	go to cabinet 1	30-step limit reached; the episode fails; lettuce never found	0

Blind judge (independent LLM; saw only this action–feedback text): decisive step $= 8$; verdict: "Never found lettuce; repeatedly revisited fridge, countertop and cabinets already seen instead of new spots."

(a) GRPO (diag-SFT) failed training rollout · update 54 · all-fail group (0/8) · 30 steps

Task: put a clean pan in stoveburner. Steps 0–12 search seven locations; the agent then mostly revisits them, never checks the stoveburners and runs out of the 30-step budget.



Step 13 · judge's decisive step — **negative credit 0**

Action `go to fridge 1`

Feedback You arrive at fridge 1. The fridge 1 is open. In it, you see a bowl 1, a cup 1, a tomato 3, and a tomato 1.

Same contents as at step 4, an already-searched location. Like every valid step, its advantage is +0.013.

Steps 5, 20 and 22 · unparseable — **0.087 each**

Action `(no action emitted / unparseable)`

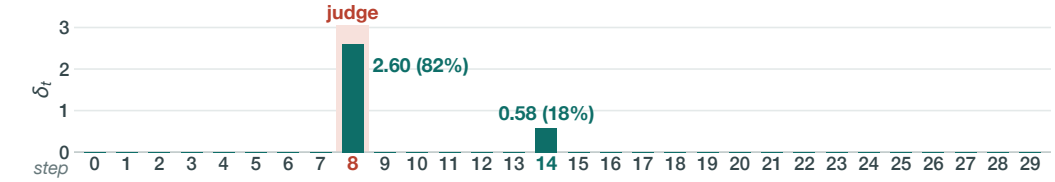
Feedback Nothing happens.

Format shaping sets these steps 0.1 below the valid ones: the only per-step contrast in an all-fail group.

Blind judge: *"Sustained revisit churn of seen fridge/cabinets/countertops; stoveburners never checked for a pan."*

(b) FAULT failed training rollout · update 70 · all-fail group (0/8) · 30 steps · P = 3.18

Task: clean some lettuce and put it in sidetable. Steps 0–7 search five locations; the agent then mostly revisits them and never finds the lettuce. No step is invalid or a no-op.



Step 8 · judge's decisive step — **δ = 2.60 (82%)**

Action `go to fridge 1`

Feedback You arrive at fridge 1. The fridge 1 is open. In it, you see a cup 1, and a potato 1.

Same contents as at step 1: the first revisit of a searched location.

Step 14 — **δ = 0.58 (18%)**

Action `go to cabinet 3`

Feedback You arrive at cabinet 3. The cabinet 3 is open. In it, you see nothing.

Revisits a cabinet known to be empty since step 6.

Blind judge: *"Never found lettuce; repeatedly revisited fridge, countertop and cabinets already seen instead of new spots."*

(c) GiGPO (diag-SFT) failed training rollout · update 54 · all-fail group (0/8) · 30 steps



Negative advantage falls only on its two format-flagged steps (6 and 24); the run stored no rollout text, so there is no transcript or judge label.

Figure 18: **Transcripts and step penalties.** (a) GRPO (diag-SFT) and (b) FAULT failures from separate all-fail groups, each with its per-step credit: GRPO's negative optimizer advantage $\max(-A_t, 0)$ and FAULT's diagnostic allocation δ_t. The blind judge selects steps 13 and 8, respectively (shaded); GRPO marks only three unparseable steps, while FAULT allocates approximately 82% to the judged step. (c) Negative advantage for a GiGPO (diag-SFT) all-fail trajectory, marking only format violations; no transcript was stored. Action and feedback text is verbatim; italic notes are ours. Task instances differ.

Task: put a hot tomato in sidetable. FAULT · update 30 · success in 19 steps · P = 5.52

0	Observation	-= Welcome to TextWorld, ALFRED! =- You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a ...
	Reasoning	*I need to put a hot tomato in the sidetable. To do this, I first need to obtain a tomato. Tomatoes are typically stored in cabinets, on countertops, or in the fridge. The most likely place to find a tomato is in one of the cabinets or in the fridge. I should start by exploring possible locations ...*
	Action	**`go to fridge 1`**
1	Observation	You arrive at fridge 1. The fridge 1 is closed.
	Reasoning	*I am currently at fridge 1, which is closed. My goal is to put a hot tomato in the sidetable. To do that, I first need to obtain the tomato. The tomato is likely to be stored in one of the cabinets, on a countertop, or in the fridge. Since the fridge is closed, I can either: • Open it to check if the ...*
	Action	**`open fridge 1`**
2	Observation	You open the fridge 1. The fridge 1 is open. In it, you see a bowl 3, a cup 1, a mug 1, and a tomato 2.
	Reasoning	*I am currently at fridge 1, which is open. Inside, I see a bowl 3, a cup 1, a mug 1, and a tomato 2. My goal is to "put a hot tomato in sidetable." This means I need to first obtain a tomato, then heat it, and finally place it on the sidetable. Currently, I have the tomato available in the fridge (via ...*
	Action	**`take tomato 2 from fridge 1`**
3	Observation	You pick up the tomato 2 from the fridge 1.
	Reasoning	*I currently have the tomato 2 in my hand, having just picked it up from the fridge 1. My goal is to "put a hot tomato in sidetable", which means I need the tomato to be hot before placing it on the sidetable. Currently, the tomato is cold (as it was taken from the fridge). To make it hot, I need ...*
	Action	**`heat tomato 2 with microwave 1`** *before reaching the microwave* **δ = 2.11**
4	Observation	Nothing happens.
	Reasoning	*I am currently holding the tomato 2, which I picked up from the fridge 1. My goal is to "put a hot tomato in sidetable", which means I need to heat the tomato first and then place it on the sidetable. So far, I have: • Taken the tomato from the fridge. • Attempted to heat it using the ...*
	Action	**`heat tomato 2 with microwave 1`** *repeats the failed heat* **δ = 1.41**
5	Observation	Nothing happens.
	Reasoning	*I am currently holding a tomato that I picked up from the fridge. ... Since I am currently in the fridge (as implied by the context of the last action), and I have not yet gone to the microwave, I should go to the microwave to perform the heating action. ...*
	Action	**`go to microwave 1`**
6	Observation	You arrive at microwave 1. The microwave 1 is closed.
	Reasoning	*I am currently at microwave 1, and the microwave is closed. The previous action of heating the tomato with the microwave did not yield any result, and I am now at this location. The admissible actions include "open microwave 1", which is the logical first step to proceed. Since the ...*
	Action	**`open microwave 1`**
7	Observation	You open the microwave 1. The microwave 1 is open. In it, you see nothing.
	Reasoning	*I am currently at the microwave, which is open and empty. ... The only way to heat the tomato is by placing it in the microwave, and the action 'heat tomato 2 with microwave 1' is admissible, but it may require the tomato to be physically inside the microwave first. ...*
	Action	**`move tomato 2 to microwave 1`**
8	Observation	You move the tomato 2 to the microwave 1.
	Reasoning	*I am currently at the microwave 1, and I have just moved the tomato 2 into it. The microwave is open, and there is no indication that it is already heating or that the heating process has started. The previous attempt to heat the tomato directly did nothing, likely because the tomato was ...*
	Action	**`close microwave 1`**
9	Observation	You close the microwave 1.
	Reasoning	*I am currently at the microwave 1, which I have just closed. The tomato 2 is now inside the microwave, and I have successfully placed it there. The next logical step is to heat the tomato using the microwave. The current observation confirms that I am at microwave 1, and I have ...*
	Action	**`heat tomato 2 with microwave 1`** *tomato left inside the closed microwave* **δ = 1.41**

Figure 19: **Recovery within a successful trajectory, steps 0–9.** The agent makes three ineffective heating attempts. Charge tags show the step allocation of budget $P = 5.52$; the case continues in Figure 20.

Task: put a hot tomato in sidetable. FAULT · update 30 · steps 10–18 (continued) · P = 5.52

10	Observation	Nothing happens.
	Reasoning	*I am currently at step 11, and the observation is "Nothing happens" after attempting to heat the tomato in the microwave. However, the most recent action taken ...*
	Action	`examine microwave 1` **δ = 0.03**
11	Observation	The microwave 1 is closed.
	Reasoning	*I am currently at microwave 1, which is closed. I have just heated the tomato in the microwave, but nothing happened when I tried to heat it. This suggests that the heating process may not have been successful or may have been interrupted. The last action I took was to heat the ...*
	Action	`examine microwave 1`
12	Observation	The microwave 1 is closed.
	Reasoning	*I am currently at step 13, and my observation is that "The microwave 1 is closed." This means that the microwave is currently sealed, and I cannot directly interact with it to heat the tomato without first opening it. The previous action ("heat tomato 2 with microwave 1") did not ...*
	Action	`open microwave 1`
13	Observation	You open the microwave 1. The microwave 1 is open. In it, you see a tomato 2.
	Reasoning	*I am currently at step 14, and I have just opened the microwave. Inside the microwave, I see the tomato (tomato 2) that I heated earlier. However, the heating process did not succeed ...*
	Action	`examine microwave 1`
14	Observation	The microwave 1 is open. In it, you see a tomato 2.
	Reasoning	*... The microwave 1 is open. In it, you see a tomato 2. • The goal is to "put a hot tomato in sidetable". • Currently, the tomato is in the microwave and is hot (as per observation 10, we heated it). • However, the microwave is open, and the tomato is inside it. • The next logical ...*
	Action	`take tomato 2 from microwave 1` *assumes the tomato is already hot*
15	Observation	You pick up the tomato 2 from the microwave 1.
	Reasoning	*I am currently holding the tomato 2, which was just taken from the microwave 1. The goal is to "put a hot tomato in sidetable." However, the tomato I currently hold is not hot yet ... The key missing piece is confirming that the tomato is hot. ...*
	Action	`examine tomato 2` *corrective check, still charged* **δ = 0.56**
16	Observation	This is a cold tomato 2.
	Reasoning	*I currently have a cold tomato 2 in my hand. The goal is to "put a hot tomato in sidetable." This means I need to heat the tomato first, then place it on the sidetable. Currently, I have the tomato, but it's cold. I need to heat it using a heat source. The only available heat source in the ...*
	Action	`heat tomato 2 with microwave 1` **false belief corrected**
17	Observation	You heat the tomato 2 using the microwave 1.
	Reasoning	*... The task at hand is "put a hot tomato in sidetable", which means we need to move the heated tomato to the sidetable. Currently, we are in a state where: • The tomato 2 is already hot (as confirmed by action 17). • We are not currently in the sidetable. • We are likely currently near ...*
	Action	`go to sidetable 1`
18	Observation	You arrive at sidetable 1. On the sidetable 1, you see a apple 1, a butterknife 1, a dishsponge 2, a lettuce 1, a peppershaker 3, a peppershaker 2, and a tomato 1.
	Reasoning	*I am currently at sidetable 1, and I have just heated the tomato 2 using the microwave. The goal is to "put a hot tomato in sidetable 1". Currently, on sidetable 1, I see a tomato 1 (not heated), which is likely the original unheated tomato. I have just heated tomato 2, which is now hot. The ...*
	Action	`move tomato 2 to sidetable 1` **episode succeeds (reward 10)**

Where the budget goes: 89% of P = 5.52 falls on the three ineffective heat attempts (steps 3, 4 and 9) and 0.56 (10%) on the corrective check at step 15, so the allocation is imperfect. A clean 8-step success in the same group has P = 0.

Figure 20: **Recovery within a successful trajectory, steps 10–18.** Continued from Figure 19. Examination reveals the tomato is still cold, after which the agent corrects its belief and completes the task. Most of the budget targets ineffective heating attempts, although the corrective examination also receives a penalty.

Reading the transcript. Each step shows the input observation, generated reasoning and action. Observations and actions follow the stored transcript, with long passages shortened by "..."; reasoning is excerpted model-generated text with chat-template artifacts removed, not a statement of environment facts. Charge tags show the diagnostic allocation δ_t of Eq. (25), and short notes beside the actions are ours.

G Extended Related Work

This section expands Section 4 along the same three lines.

Outcome-based agentic RL. Agentic RL treats an LLM as a policy that acts over many turns and learns from the rewards it observes (Zhang et al., 2026a). Following the success of GRPO for reasoning models (Shao et al., 2024; Guo et al., 2025), it has been applied with GRPO, PPO or their variants to search (Jin et al., 2025), tool use (Qian et al., 2025; Feng et al., 2026), web navigation (Wei et al., 2025), and games and embodied text environments (Wang et al., 2025; Feng et al., 2025). GRPO needs no critic: it normalizes terminal rewards across the rollouts of a task and assigns each rollout's advantage to every step. This is simple and scales well, but the advantage of a step reflects the whole trajectory rather than that step.

Two consequences follow for long-horizon tasks. With binary rewards, a group whose rollouts all succeed or all fail has zero advantage for every rollout; DAPO filters such groups out (Yu et al., 2025), while Yang et al. (2026a) give all-fail groups a fallback advantage from first-visit observation coverage. In our ALFWorld runs, these same-outcome groups make up between 59% and 64% of training groups (Section 3.3.1). Within a failed rollout, every step receives the same credit, so the decisive mistake is not singled out. Agent-specific methods such as ARPO (Dong et al., 2026) and GiGPO (Feng et al., 2025) change how rollouts are sampled or compared, but their credit is still computed from outcomes.

Numeric process signals. Process supervision assigns rewards to intermediate steps. In mathematical reasoning, step rewards from human labels (Lightman et al., 2024), automatic rollout labels (Wang et al., 2024) or progress estimates (Setlur et al., 2025) improve on outcome-only rewards. Agentic RL has adopted the idea: process reward models for agents learn step values from Monte Carlo rollout returns (Choudhury, 2025) or temporal-difference estimates (Xi et al., 2026), and recent methods score steps with prefix-aware reward models or the agent's own hindsight (Kim et al., 2026; Ma et al., 2026; Yao et al., 2026). These signals are dense and can separate rollouts that share an outcome.

Their difficulty is trust. A learned step reward is itself a model that the policy can exploit; DeepSeek-R1 avoids neural process rewards partly for this reason (Guo et al., 2025), and further optimizing an agent PRM can lower the true success rate (Choudhury, 2025). GiGPO avoids a learned model by comparing actions taken from the same environment state (Feng et al., 2025). Its outcome-based credit, however, comes from discounted returns, so it vanishes in all-fail groups and, when present, spreads over the steps after a wrong turn (Section 3.3.2). FAULT keeps the terminal reward as the only trusted signal: step penalties come from verified diagnoses, and their prices are a few coefficients fitted to outcomes.

Language analysis of trajectories. Language methods analyze completed trajectories to state what went wrong and where. The analysis can guide another attempt through context or memory (Shinn et al., 2023; Zhao et al., 2024; Hu et al., 2025; Yao et al., 2024; Ge et al., 2025), earn a reward when a retry succeeds (Bensal et al., 2025), produce corrected training trajectories (Yuan et al., 2025; Zhang et al., 2026b; Li et al., 2026), or be distilled into the policy (Yang et al., 2026b; Zheng et al., 2026; Bi et al., 2026; Wu et al., 2026). SEED is closest to our setting: one checkpoint acts, analyzes its rollouts into hindsight skills and learns from them by on-policy distillation, with strong gains on ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022). Such analysis carries the information that outcome credit lacks.

Three gaps keep it from solving the two credit-assignment problems above. First, the analysis rarely sets RL credit: it serves as context, a training target or an object of reward, even when pooled across trajectories as in SAMULE (Ge et al., 2025). GRSD (Zheng et al., 2026) does use pooled reflections to rescale each step's outcome advantage, but this advantage is still zero in same-outcome groups, so equal outcomes still receive equal outcome credit. Second, error claims are not checked

at the stated step; on TRAIL, the best evaluated model reaches only about 11% joint accuracy in identifying and locating errors in agent traces (Deshpande et al., 2025), even though error taxonomies are available (Cemri et al., 2025; Zhu et al., 2025). Third, the weight of the analysis is set by fixed loss coefficients, mixing schedules or a single retry, not learned from outcomes. FAULT closes these gaps with diagnoses restricted to a fixed taxonomy and checked by quotes (Appendix B.2), prices fitted to outcomes within each task (Appendix B.3), and a bounded penalty budget redistributed over verified error steps.