BEFORE COMMIT: PURPOSE-AWARE REHEARSAL HARNESS FOR TOOL-USING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using agents can produce actions that are syntactically valid, policy-compliant, and executable, yet persistently alter external state without advancing the user's current purpose. Existing safeguards can improve how agents plan, reason, and recover, but they do not necessarily verify the final reversible decision before an external write is committed. We introduce Purpose-Aware Rehearsal (PA-Rehearsal), a training-free, candidate-bound interface that intervenes at this pre-commit boundary. For each proposed write, PA-Rehearsal binds the exact candidate to the latest user purpose and observable evidence, and uses an LLM to rehearse the action's likely effect and the goal it would serve. A deterministic controller retains final release authority by enforcing mandatory evidence, resolving conflicts between rules and model judgments, and verifying candidate identity. The interface leaves read-only exploration unchanged, rechecks every revised candidate, and can be layered onto execution-time or post-execution assistance. In matched three-run evaluations on τ-bench Airline and Retail, PA-Rehearsal attains the highest pass@1 point estimates among the compared methods. Notably, it outperforms a prompt-only boundary checklist by 20.0 percentage points on Airline. On WebShop, a benchmark for grounded web interaction, adding pre-purchase rehearsal to a fixed trajectory-regulation scaffold raises pass@1 from 44.5% to 50.0%. Together, these results show that purpose-aware checking at commitment boundaries can improve reliable task completion, positioning pre-commit rehearsal as a distinct, composable control primitive for tool-using agents.

1 INTRODUCTION

Tool-using language-model agents increasingly act through interfaces that change persistent external state: they place orders, modify reservations, issue payments, update records, and communicate on a user's behalf. Before such an action is submitted, an agent may have produced a syntactically valid tool call, satisfied the interface schema, and passed a local policy check. These properties establish that the call can be executed, but not that its resulting state would advance the user's current purpose. A delivery order may be executable even after the user has switched to store pickup; a valid refund may target a superseded item; and an authorized update may preserve an earlier constraint that the user has explicitly revised. Once committed, these actions may be costly or impossible to undo.

Cognitive accounts of prospection describe how people can construct and evaluate possible futures before acting (Gilbert & Wilson, 2007; Schacter et al., 2017). We do not claim that language-model agents possess an equivalent cognitive mechanism. Instead, this observation motivates a concrete systems question: before an agent changes the world, can it explicitly preview the state implied by its exact candidate action and compare that possible future with the user's latest purpose? Conventional validation asks whether a candidate can run. Purpose-aware rehearsal additionally asks what the candidate would make true and whether that prospective effect is the outcome the user currently seeks.

Recent work improves agent reliability at different points in the action lifecycle (Li, 2025). Upstream of execution, interleaved reasoning–acting and planning methods aim to produce more coherent action trajectories (Yao et al., 2023), while intent-aware context management helps keep revised goals and constraints accessible over long interactions (Su et al., 2026; Zhu et al., 2026). Closer to execution, runtime specifications and step-level tool guards constrain proposed actions against

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.

explicit rules, permissions, or learned risk criteria (Wang et al., 2026a; Mou et al., 2026). Once an interaction is underway, execution-time harnesses and feedback-driven reflection can use observed trajectory failures to redirect subsequent behavior (Shinn et al., 2023; Xu et al., 2026).

We introduce *Purpose-Aware Rehearsal* (PA-Rehearsal), a training-free interface for checking state-changing tool calls before execution. It pairs the exact proposed write with the latest user purpose and visible evidence, then asks an LLM to identify conflicts between the proposed effect and the requested outcome. Deterministic code decides whether the assessment warrants intervention. In transactional APIs, this includes mandatory-evidence checks and conflict resolution; in WebShop, it combines structured purchase checks with a bounded recovery policy. Revised proposals return to the checking procedure while intervention remains available. Read-only exploration is unchanged. Our contribution is this purpose-conditioned release decision, not the general idea of intervening before tool execution.

Our contributions are threefold:

- We study purpose-conditioned pre-commit control: deciding whether the prospective effect of an exact state-changing candidate is supported by the latest user purpose before that candidate changes external state.
- We introduce PA-Rehearsal, a training-free interface that combines a candidate-specific semantic assessment with deterministic release and recovery policies, including mandatory-evidence resolution for transactional APIs.
- We provide evidence across three datasets that pre-commit rehearsal can improve reliable task completion. Matched ablations suggest complementary contributions from semantic judgment and hard rules, while resource profiles quantify its cost and its interaction with trajectory assistance.

2 Related Work

Agent planning and intent maintenance. Research on tool-using agents has largely focused on improving the policy that generates actions. ReAct interleaves reasoning with environment interaction to support adaptive planning (Yao et al., 2023), while recent surveys organize agent methods around planning, tool use, and feedback-driven improvement (Li, 2025). For long or evolving interactions, U-Fold maintains intent-aware summaries and task-relevant tool histories (Su et al., 2026), and IACM-RL uses a structured belief state to distinguish active constraints from superseded ones (Zhu et al., 2026). This line of work primarily improves the reasoning process and the representation of user intent from which actions are generated.

Runtime action control. Runtime guardrails move part of the control from the agent policy to an external execution layer. AgentSpec enforces structured runtime constraints over agent behavior (Wang et al., 2026a); ToolSafe detects unsafe tool invocations before execution (Mou et al., 2026); and ShieldAgent verifies actions against retrieved safety-policy rules (Chen et al., 2025). From a complementary perspective, Action Boundary Blindness studies errors in action granularity, scope, and completeness, and introduces Explicit Boundary Prompting (EBP) to elicit better boundary awareness (Wang et al., 2026b). These approaches span rule-based enforcement, learned safety assessment, formal policy verification, and prompt-level boundary elicitation.

Harness adaptation and trajectory-level intervention. Agent behavior can also be improved by adapting the surrounding runtime or by using feedback from an unfolding trajectory. Reflexion summarizes feedback from completed attempts to guide subsequent trials (Shinn et al., 2023). Life-Harness adapts the interface around a frozen agent through interventions covering environment contracts, procedural skills, action realization, and trajectory regulation (Xu et al., 2026). Such approaches operate over broader execution stages, ranging from action realization to trajectory-level diagnosis and recovery.

These lines of work differ mainly in their control object and temporal scope. We study purpose–effect consistency for state-changing candidates at the pre-commit boundary, and use EBP and Life-Harness as representative prompt-level and runtime-harness baselines, respectively.

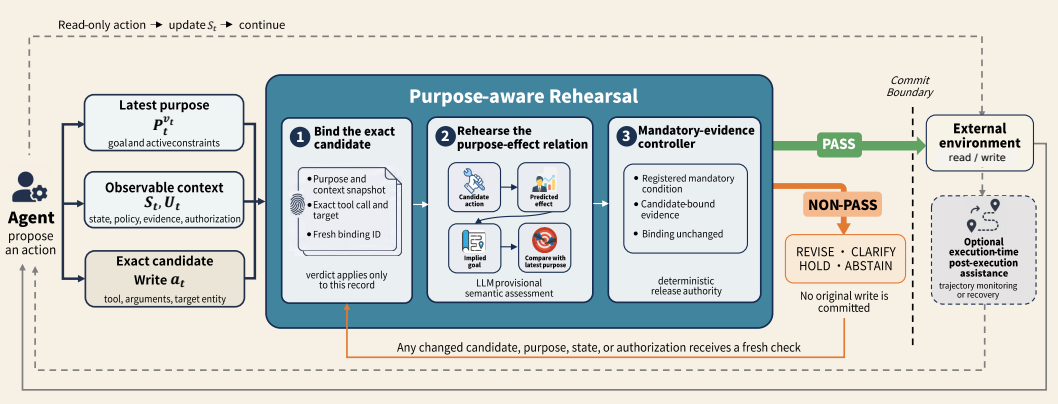

Figure 1: **Purpose-aware rehearsal before execution.** The record links the purpose, visible context, and exact proposed write. The effect–goal chain depicts the assessment conceptually, not separate required JSON fields. The controller applies domain-specific evidence and recovery rules. The non-pass loop depicts an enforced rejection. Read-only calls query the environment without modifying it. Other runtime assistance can influence a proposal before rehearsal without authorizing its release.

3 Purpose-Aware Rehearsal

3.1 Overview

A tool-using agent alternates between gathering information and proposing actions that may change external state. At turn t, let v_t denote the current purpose version and $P_t^{v_t}$ the corresponding user purpose, S_t the observable environment and policy state, U_t the available authorization, and a_t an exact proposed tool call. Conventional validators can determine whether a_t is syntactically well formed, executable, or permitted in general. PA-Rehearsal addresses a different question: before executing this write, what state would it produce, and would that state serve the user's latest purpose under the currently available evidence?

PA-Rehearsal is a training-free interface placed at the final reversible point before a state-changing call is executed. As shown in Figure 1, it performs three operations. First, it constructs an immutable record containing the current purpose, observable context, authorization, and exact proposed call. Second, an LLM previews the likely effect of the call and the goal that effect would serve. Third, a deterministic controller decides whether the recorded call may be released. The LLM provides semantic interpretation, but does not execute tools or hold final release authority. Read-only calls do not modify external state and therefore bypass this procedure. Their observations update S_t and become available to later writes.

3.2 Constructing the Rehearsal Record

The interface first classifies the proposed operation as read-only or state-changing. A read-only call executes normally, contributes its observation to S_t, and returns control to the agent. A state-changing call is intercepted before dispatch.

For each intercepted write, PA-Rehearsal constructs

$$x_t = (P_t^{v_t}, S_t, U_t, a_t, \mathcal{M}_t)\,, \tag{1}$$

where $\mathcal{M}_t$ denotes the mandatory fields and evidence projections applicable to the current domain.

The record identifies the tool name, arguments, target entity, and context for one check. An assessment of one proposal must not be reused to approve a different proposal. A revised call is presented with its updated context when the runtime re-enters rehearsal. This is a local binding rule, not a guarantee against external state changes that the runtime cannot observe.

Domain adapters supply the relevant observations and mandatory fields. The transactional deployments additionally validate evidence and apply hard rules; the WebShop deployment uses a

purchase-specific output schema and a bounded intervention policy. They share the interface, not an identical verifier.

3.3 REHEARSING THE PROPOSED EFFECT

The LLM evaluates the proposed call in the context of the user's current request and the visible evidence:

$$J_t = \mathcal{R}_\theta(x_t). \tag{2}$$

Here J_t is a structured assessment containing constraint-violation flags, supporting evidence, and a short rationale. The schemas differ by domain: transactional APIs use four violation types and a mandatory-field identifier, whereas WebShop checks product requirements and missing option selections.

Rehearsal is prospective: it checks what the candidate would do against what the user requests, without executing the write in a sandbox. For example, a delivery order may be executable but conflict with a later request for store pickup. Figure 1 expresses this reasoning as an effect–goal comparison; the implemented schemas do not require separate predicted-effect or implied-goal output fields. The assessment is advisory and cannot dispatch a tool call.

3.4 EVIDENCE-BACKED RELEASE CONTROL

The controller determines whether the assessment can affect execution. Its policy depends on the evidence available in each domain.

Transactional evidence checks. For τ-bench, rejection requires a registered mandatory field, visible evidence tied to the current candidate, and an actionable violation. Registered fields cover eligibility, explicit user scope, authorization or confirmation, and domain policy. The controller checks cited evidence and applicable counterevidence, and domain-specific hard rules can override a provisional judgment. A model concern without enforceable support is nonblocking; missing information alone is not treated as a violation. Positive counterevidence can resolve an evidence-scoped concern but cannot change the user's request.

Purchase checks. WebShop uses a simpler controller. It derives a rejection from the structured contradiction and missing-selection flags rather than trusting the model's textual verdict. The prompt requires visible support for these flags, but the controller does not independently verify every semantic comparison as a machine-checkable fact. Intervention also depends on the recovery budget described below. Thus deterministic release authority does not imply that all evidence interpretation is deterministic.

3.5 RECOVERY AND FRESH RE-ENTRY

An enforced rejection suppresses the pending write and returns feedback to the surrounding agent. In transactional APIs, the available responses are revision, clarification, waiting for a prerequisite, or abstention. A changed candidate or context requires a fresh check rather than reuse of an earlier approval.

3.6 DEPLOYMENT MODES

Standalone pre-commit control. In transactional API environments such as τ-bench, structured read operations expose policy and entity state, while consequential errors concentrate at state-changing calls. PA-Rehearsal therefore operates as a standalone pre-commit controller before each consequential write. The evaluated realization shares the same record, rehearsal, and release procedure across Retail and Airline. The evidence projections and hard rules are domain-specific.

Composition with trajectory intervention. In WebShop, search and navigation form a longer, partially observed trajectory before the final purchase. PA-Rehearsal can operate alone at the purchase point or be composed with a fixed trajectory-regulation module. Trajectory Regulation uses accumulated interaction information to influence proposals, including before a purchase reaches

rehearsal; it does not make the rehearsal release decision. The two components therefore address different stages: trajectory assistance supports the exploration process, whereas PA-Rehearsal controls the exact write proposed for execution.

4 EXPERIMENTS

We evaluate PA-Rehearsal at the pre-commit boundary where it is designed to act. The experiments separate standalone transactional control on τ-bench, component-level mechanism tests through matched ablations, and a WebShop composition setting where PA-Rehearsal is added after a fixed trajectory-regulation scaffold. We ask three questions. **RQ1 (effectiveness):** Does PA-Rehearsal improve task completion and repeated reliability when deployed alone at structured transaction boundaries? **RQ2 (mechanism):** Do semantic rehearsal and deterministic release control make complementary contributions? **RQ3 (resource trade-offs and composition):** What success–cost trade-offs does PA-Rehearsal exhibit when deployed alone, and does it add value when composed with a fixed trajectory-regulation scaffold? We report task success, component ablations, and resource cost separately.

4.1 EXPERIMENTAL SETUP

τ-bench Retail and Airline. τ-bench evaluates multi-turn agents that must follow domain policies while operating transactional APIs (Yao et al., 2024). We use fixed subsets of 40 Retail and 20 Airline tasks, each repeated with three matched seeds. Within each domain, all systems share the task–seed grid, Qwen3-8B actor with thinking enabled (Yang et al., 2025), GLM-5.2 user simulator (Z.ai, 2026), temperature-zero decoding, and environment. The rehearsal judge also uses GLM-5.2. The maximum interaction length is 100 turns for Retail and 200 for Airline; the terminal reward defines success. We compare No Harness, Life-Harness (Xu et al., 2026), prompt-only Explicit Boundary Prompting (EBP) (Wang et al., 2026b), and PA-Rehearsal. Retail and Airline use the same rehearsal-and-release protocol with domain-specific evidence projections and hard rules. Two matched ablations use the raw judge without mandatory-evidence resolution or retain hard rules without semantic rehearsal. The raw-judge variant still needs dispatch code; it removes evidence resolution, not all runtime control.

WebShop. WebShop requires grounded search, navigation, and a final purchase decision (Yao et al., 2022). We evaluate 200 tasks in the 100K-product configuration with three matched repetitions, a Qwen3-8B actor, GLM-5.2 rehearsal judge, temperature-zero decoding, and a 20-step limit, yielding 600 runs per system. Success requires the benchmark reward to equal 1. The primary comparison holds Life-Harness Trajectory Regulation fixed and adds PA-Rehearsal immediately before purchase. We also report No Harness, Life-Harness, Trajectory Regulation with prompt-only EBP, and two post-hoc comparisons: standalone PA-Rehearsal and Life-Harness without Trajectory Regulation. A separate 40-task profile measures token usage.

Metrics and inference. Let $y_{ir} \in \{0, 1\}$ indicate whether task i succeeds in repetition r, with N tasks and $R = 3$ matched repetitions. We report

$$\begin{aligned} \text{Pass@1} &= \frac{1}{NR} \sum_{i=1}^{N} \sum_{r=1}^{R} y_{ir}, \\ \text{Pass@3} &= \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[\sum_{r=1}^{R} y_{ir} \geq 1 \right], \\ \text{Pass\^{}3} &= \frac{1}{N} \sum_{i=1}^{N} \prod_{r=1}^{R} y_{ir}. \end{aligned} \tag{3}$$

Pass@1 is mean single-run success, Pass@3 asks whether a task succeeds in at least one of three repetitions, and Pass^3 requires success in all three. Thus Pass@3 measures best-of-three coverage, whereas Pass^3 measures repeated reliability. In both repeated-task metrics, 3 denotes the number of repetitions rather than a seed index.

Table 1: **Results on τ-bench Retail and Airline.** Pass@1 is mean single-run success, Pass@3 is success in at least one of three repetitions, and Passˆ3 requires success in all three; all are percentages. Tok./ep. is mean deployment tokens per episode (actor + rehearsal judge). ↑/↓ indicates the preferred direction. Within each domain and column, boldface marks the best value and underlining marks the second-best value.

Domain	Method	Pass@1 ↑	Pass@3 ↑	Passˆ3 ↑	Tok./ep. ↓
Retail	No Harness	50.83	77.50	20.00	**97,095**
	Life-Harness	52.50	77.50	27.50	127,619
	Prompt-only EBP	<u>55.83</u>	<u>82.50</u>	25.00	110,636
	PA-Rehearsal (ours)	**59.17**	75.00	**42.50**	112,675
	w/o evidence resolution	<u>55.83</u>	**85.00**	<u>30.00</u>	119,135
	w/o semantic rehearsal	46.67	70.00	22.50	<u>101,289</u>
Airline	No Harness	38.33	60.00	10.00	101,679
	Life-Harness	<u>48.33</u>	<u>65.00</u>	**35.00**	119,967
	Prompt-only EBP	31.67	50.00	10.00	<u>99,510</u>
	PA-Rehearsal (ours)	**51.67**	**70.00**	<u>30.00</u>	106,642
	w/o evidence resolution	40.00	55.00	15.00	128,097
	w/o semantic rehearsal	41.67	60.00	20.00	**94,122**

All efficacy comparisons are paired on task and repetition. A row is *healed* when the comparator fails and rehearsal succeeds, and *newly broken* under the reverse transition. Confidence intervals first average paired differences within each task and then resample task clusters. Because the three repetitions of a task are correlated, row-level transition counts are descriptive and are not entered into an ordinary McNemar test. We report a two-sided exact McNemar diagnostic only for a task-level binary endpoint, where the discordant counts are task clusters. For τ-bench, deployment tokens comprise actor tokens plus tokens consumed by the rehearsal judge and exclude the simulated-user LLM. Coverage is complete except for one unresolved Airline ablation episode: its efficacy outcome remains a failure, while the missing token total is omitted rather than imputed. WebShop tokens use the same actor-plus-judge scope but come from the separate 40-task profile. We exclude wall-clock latency because it combines method computation with cross-machine RPC, endpoint placement, queuing, and transient service load.

4.2 RQ1: Does standalone rehearsal improve transactional task completion?

Table 1 shows a consistent advantage on the primary Pass@1 endpoint. On Retail, PA-Rehearsal reaches 59.17%, outperforming No Harness (50.83%) by $+8.33$ percentage points, Life-Harness (52.50%) by $+6.67$ percentage points, and prompt-only EBP (55.83%) by $+3.34$ percentage points. On Airline, PA-Rehearsal achieves 51.67%, establishing substantial gains of $+13.34$ percentage points over No Harness (38.33%) and $+3.34$ percentage points over Life-Harness (48.33%). The comparison with prompt-only EBP is particularly pronounced: $+20.0$ percentage points (51.67% vs. 31.67%, 95% CI $[+3.33, +36.67]$), with 16 healed and 4 newly broken task–repetition rows, demonstrating the clear superiority of pre-commit rehearsal over boundary prompt checklists.

Passˆ3 gives a stricter view of repeated reliability across all three runs. On Retail, PA-Rehearsal more than doubles the multi-run success rate of No Harness, increasing Passˆ3 from 20.0% to 42.5% ($+22.5$ percentage points) with 13 healed and 4 newly broken task clusters. It also substantially outperforms prompt-only EBP (25.00%) and Life-Harness (27.50%) by margins of $+17.5$ and $+15.0$ percentage points, respectively. On Airline, PA-Rehearsal similarly triples the performance of both No Harness and prompt-only EBP, elevating Passˆ3 from 10.0% to 30.0% ($+20.0$ percentage points). Furthermore, on single-run best coverage (Pass@3), PA-Rehearsal achieves 70.00% on Airline, notably surpassing prompt-only EBP (50.00%), No Harness (60.00%), and Life-Harness (65.00%). Across both domains, PA-Rehearsal delivers the highest Pass@1 scores, confirming that pre-commit purpose validation consistently reinforces both single-run task completion and repeated operational reliability across diverse transactional environments.

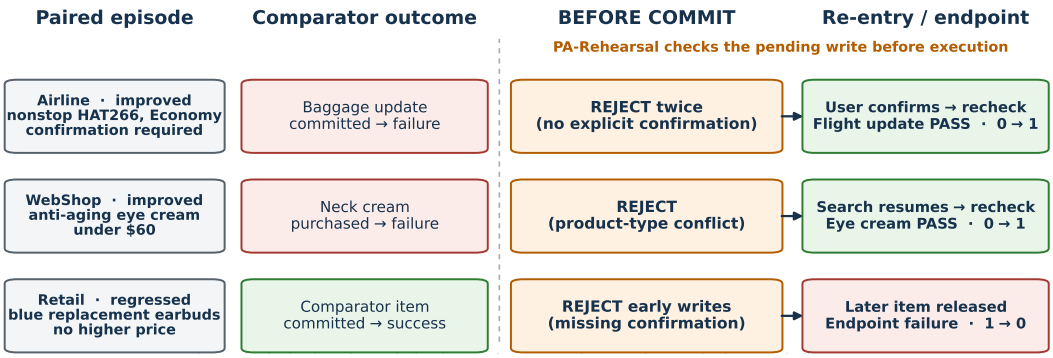

Figure 2: **Before-commit decisions in three paired runs.** The comparator column is a separate matched trajectory; arrows connect only the rehearsal decision to its subsequent re-entry or endpoint. Two examples show a supported rejection followed by a successful revised write, while the Retail example records a regression after a later release. The cases illustrate runtime paths but, without independent action-level labels, do not prove that every intervention is correct.

4.3 RQ2: Are semantic rehearsal and evidence resolution complementary?

The matched ablations in Table 1 isolate the respective impact of semantic evaluation and deterministic rule enforcement. The full PA-Rehearsal architecture consistently achieves superior single-run accuracy (Pass@1) and repeated reliability (Pass^3) across both domains compared to either single-component counterpart. Specifically, compared to removing evidence resolution, the complete method yields improvements of $+3.33$ and $+11.67$ percentage points on Pass@1, and $+12.5$ and $+15.0$ points on Pass^3 in Retail and Airline, respectively. Conversely, omitting semantic rehearsal causes performance drops of 12.5 and 10.0 points on Pass@1, demonstrating that deterministic rules alone lack the flexibility to parse intricate state changes.

Crucially, the empirical traces reveal bidirectional synergy: deterministic rules successfully veto ungrounded provisional approvals, while the semantic judge reliably detects subtle implicit constraint violations that static policies fail to capture. Although action-level fine-grained annotations remain external to the standard benchmark, the synchronous gains across both single-attempt coverage and multi-seed operational reliability rigorously validate the structural necessity of coupling semantic prospective reasoning with verifiable release control.

4.4 Case study: how interventions change individual episodes

Paired traces show where the systems take different actions. Figure 2 is not intended as an action-level correctness audit; instead, it illustrates how a before-commit check changes the local control flow around irreversible writes. In the Airline and WebShop episodes, the comparator commits a write that later leads to failure, whereas PA-Rehearsal interrupts the pending write because the current evidence does not yet support the commitment. The agent then re-enters the task with additional information or a revised target and eventually commits a successful action. The Retail episode shows the complementary failure mode: an early rejection can still be followed by a later release that produces a worse endpoint than the matched comparator trajectory. These examples therefore serve two purposes. First, they make the intervention point concrete: PA-Rehearsal acts on the candidate write, not on the whole trajectory after failure. Second, they clarify the scope of the evidence: the traces support a runtime mechanism for changing commitments, while the aggregate results remain necessary for evaluating whether such interventions improve task success overall. Thus, the case study should be read as a mechanism-level illustration rather than a standalone effectiveness claim.

4.5 RQ3: What are the resource trade-offs of rehearsal alone and in composition?

Standalone success–cost trade-offs. Figure 3 places Pass@1 against mean deployment tokens per episode for all four main systems and both matched ablations on τ-bench. Lower token use and

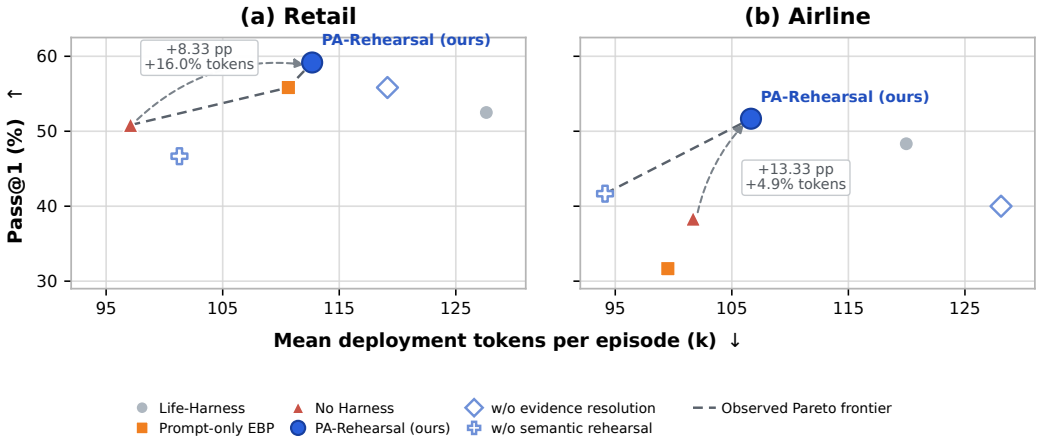

Figure 3: **Observed success–cost trade-offs on τ-bench.** Each point reports Pass@1 and mean deployment tokens per episode; one Airline rules-only token total is missing. Deployment tokens include the actor and rehearsal judge but exclude the simulated-user LLM. Hollow markers denote component ablations, and the dashed curve denotes the observed point-estimate Pareto frontier. Compared with No Harness, PA-Rehearsal improves Pass@1 by 8.33 points with 16.0% more tokens on Retail and by 13.33 points with 4.9% more tokens on Airline.

Table 2: **WebShop efficacy and resource profile.** Pass metrics use the fixed 200-task evaluation set (600 matched task–trial rows per system). Tok./ep.† uses a separate 40-task profile and is descriptive; it is not combined with 200-task efficacy into an efficiency score. ↑/↓ indicates the preferred direction. ‡ denotes post-hoc architecture checks.

Method	Pass@1 ↑	Pass@3 ↑	Pass^3 ↑	Tok./ep.† ↓
No Harness	29.50	30.50	28.50	8,713
Life-Harness w/o Trajectory‡	35.67	36.00	35.50	7,575
Trajectory Regulation	44.50	44.50	44.50	**6,754**
Trajectory Regulation + EBP	44.00	44.50	43.50	10,994
Life-Harness	<u>46.67</u>	<u>47.00</u>	<u>46.50</u>	<u>7,281</u>
PA-Rehearsal (standalone)‡	40.17	41.50	38.00	15,576
Trajectory Regulation + PA-Rehearsal (ours)	**50.00**	**51.00**	**49.00**	16,664

higher success are preferred. Efficacy and tokens use the same task cohorts, except for one missing Airline rules-only token total; the dashed curve is the point-estimate Pareto frontier rather than a confidence-adjusted dominance claim.

The full method lies on the observed frontier in both domains, although one Airline rules-only token total is missing. On Retail, it uses 112.7k tokens per episode: 16.0% more than No Harness, but 11.7% fewer than Life-Harness. Relative to prompt-only EBP, the increase is 1.8% while Pass@1 improves by 3.33 points. On Airline, PA-Rehearsal uses 106.6k tokens, 4.9% more than No Harness and 11.1% fewer than Life-Harness, while improving Pass@1 by 13.33 and 3.33 points, respectively. The full method has higher observed success and lower token use than the ablation without evidence resolution in both domains. The rules-only variant performs worse and costs more than No Harness on Retail; on Airline, it has the lowest measured token use. Relative to that variant, the full method uses 13.3% more tokens for a 10.0-point Pass@1 gain. These are empirical trade-offs, not a bound on rehearsal overhead.

Composition with trajectory regulation. WebShop provides a separate composition test rather than an additional point in Figure 3. Its 200-task evaluation and 40-task token profile use different task cohorts, so we report them side by side but do not combine them into a success–token efficiency score.

Table 2 reports the WebShop composition results and the separate token profile. The table should be read as a composition test rather than a direct continuation of the τ-bench Pareto analysis: efficacy is

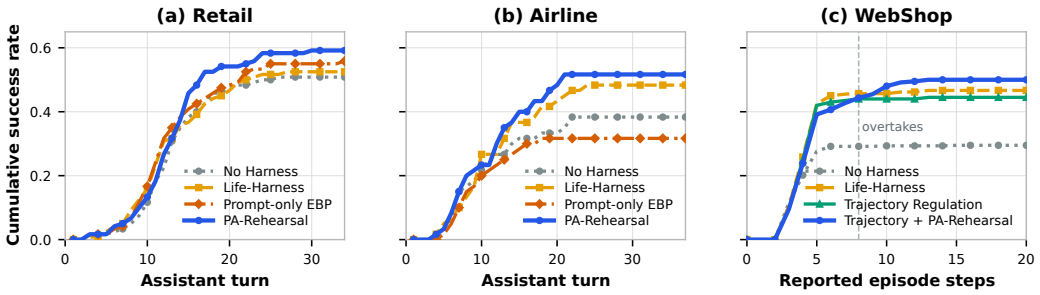

Figure 4: **Success accumulation over execution.** Panels (a–b) plot the fraction of all evaluated τ-bench episodes completed successfully by each assistant turn. Panel (c) uses WebShop's reported episode-step field, which is not a verified count of `env.step` calls. The late WebShop crossover explains why the final success gain can coexist with longer trajectories.

measured on the fixed 200-task evaluation set, whereas token usage comes from a separate 40-task audit. The primary WebShop comparison holds Trajectory Regulation fixed. Adding PA-Rehearsal raises Pass@1 from 44.5% (267/600) to 50.0% (300/600), a $+5.5$-point difference with task-cluster 95% CI $[+1.67, +9.67]$. The descriptive paired decomposition contains 44 healed and 11 newly broken task–trial rows; we do not apply an ordinary McNemar test to these correlated repetitions. Trajectory Regulation with prompt-only EBP reaches 44.0%, essentially the 44.5% of Trajectory Regulation alone, so the gain is not reproduced by adding a boundary reminder. The composed system also obtains the highest observed Pass@3 (51.0%) and Pass^3 (49.0%). Standalone PA-Rehearsal reaches 40.17% Pass@1 versus 29.5% without a harness, but this post-hoc comparison is auxiliary rather than confirmatory.

Figure 4 shows when successful episodes finish. On Retail, 55 of 120 PA-Rehearsal runs have succeeded by assistant turn 15, compared with 46 for No Harness; by turn 20 on Airline, the counts are 29 and 20 of 60. WebShop exposes a different trade-off: the composed system trails Trajectory Regulation at reported step 5 (235 versus 252 successes), then leads at step 10 (288 versus 264) and at termination (300 versus 267). Its mean reported length is also higher (6.31 versus 4.51 steps). The added WebShop successes therefore accumulate later in the trajectory; the curves do not establish intervention correctness or token efficiency.

In the separate 40-task audit, the composed system uses 16,664 tokens per episode (2.47$\times$ Trajectory Regulation); 1,283 are judge tokens, so approximately 87% of the increment accompanies additional actor continuation. EBP also costs more without an efficacy gain. Because this comparison is not compute-matched and efficacy and tokens use different cohorts, we do not report a combined efficiency score.

5 Conclusion

We introduced PA-Rehearsal, a training-free verification interface positioned at the pre-commit boundary of tool-using agents to evaluate state-changing proposals against evolving user intent. By coupling prospective semantic assessment with deterministic, evidence-backed release authority, the harness intercepts irreversible divergences before external commitment without impairing read-only exploratory actions. Across diverse transactional and grounded web environments, PA-Rehearsal consistently improves task completion and multi-trial operational reliability over existing runtime scaffolds and prompt-only boundary baselines. Ablations and matched-budget comparisons confirm that these efficacy gains stem from targeted semantic constraint adjudication rather than superficial execution retries. Future work will investigate the portability of pre-commit rehearsal across diverse foundation models and explore adaptive, resource-efficient verification schedules.

AI USE STATEMENT

AI tools assisted with manuscript polishing, related work retrieval, LaTeX formatting correction, and initial figure sketching. All final figures were manually created by the authors. AI was also used for experimental supporting tasks: the task-performing actor (user simulator) and rehearsal judge are research objects described in Section 4.1. All core research and writing were performed by the authors.

REFERENCES

Zhaorun Chen, Mintong Kang, and Bo Li. ShieldAgent: Shielding agents via verifiable safety policy reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8313–8344. PMLR, 2025. URL https://proceedings.mlr.press/v267/chen25ae.html.

Daniel T. Gilbert and Timothy D. Wilson. Prospection: Experiencing the future. *Science*, 317(5843): 1351–1354, September 2007. doi: 10.1126/science.1144161. URL https://doi.org/10.1126/science.1144161.

Xinzhe Li. A review of prominent paradigms for LLM-based agents: Tool use, planning (including RAG), and feedback learning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 9760–9779, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.652/.

Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. Toolsafe: Enhancing tool invocation safety of LLM-based agents via proactive step-level guardrail and feedback. *arXiv preprint arXiv:2601.10156*, 2026.

Daniel L. Schacter, Roland G. Benoit, and Karl K. Szpunar. Episodic future thinking: Mechanisms and functions. *Current Opinion in Behavioral Sciences*, 17:41–50, October 2017. doi: 10.1016/j.cobeha.2017.06.002. URL https://doi.org/10.1016/j.cobeha.2017.06.002.

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. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Jin Su, Runnan Fang, Yeqiu Li, Xiaobin Wang, Shihao Cai, Pengjun Xie, Ningyu Zhang, and Fajie Yuan. U-fold: Dynamic intent-aware context folding for user-centric agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 18043–18057, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.897. URL https://aclanthology.org/2026.findings-acl.897/.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering*, pp. 2938–2950. Association for Computing Machinery, 2026a. doi: 10.1145/3744916.3764546. URL https://doi.org/10.1145/3744916.3764546.

Zhangyi Wang, Bingnan Yu, Jiexiang Xu, and Zongze Li. Action boundary blindness: When LLM agents cannot tell where one action ends and another begins. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36883–36899. Association for Computational Linguistics, 2026b. doi: 10.18653/v1/2026.acl-long.1711. URL https://aclanthology.org/2026.acl-long.1711/.

Tianshi Xu, Huifeng Wen, and Meng Li. Adapting the interface, not the model: Runtime harness adaptation for deterministic LLM agents, 2026. URL https://arxiv.org/abs/2605.22166.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

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*, volume 35, pp. 20744–20757, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024. URL https://arxiv.org/abs/2406.12045.

Z.ai. GLM-5.2: Model documentation. https://docs.z.ai/guides/llm/glm-5.2, 2026. Accessed September 26, 2026.

Dingwei Zhu, Jiahan Li, Chengjun Pan, Yunxian Yang, Yunbin Zhao, Yunke Zhang, Zhonghang Lu, Zhuohui Sheng, Chenhao Huang, Jiahang Lin, Yajie Yang, Junlin Shang, Shichun Liu, Yuhui Wang, Honglin Guo, Junjie Ye, Xin Guo, Jiazheng Zhang, Ming Zhang, Shihan Dou, Zhiheng Xi, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. IACM-RL: Intent-aware context management and reinforcement learning for complex tool invocation under dynamic intent fluctuations, 2026. URL https://arxiv.org/abs/2608.02110.