DON'T BLAME YOUR AGENT (YET): ENVIRONMENT-AWARE ERROR RECOVERY FOR LONG-HORIZON AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon agents show promise on complex tasks that require many interdependent steps, yet reliable execution remains an open challenge. Existing approaches improve agent components through training, context management, and harness design. We study the complementary problem of online error recovery, which redirects an agent away from a suboptimal intermediate state before it produces a wrong task outcome. Online recovery is particularly important for long-horizon tasks because one error can propagate through dependent later steps. Yet recovery is difficult because an agent's expected behavior depends on environmental preconditions that it can only partially observe. A failed attempt can therefore reflect either inadequate execution or an inaccurate premise about the environment. We propose ALTER, an environment-aware strategy selection framework for online agent error recovery. ALTER separates environmental premises from the behavior expected under them, allowing it to calibrate the actor or revise a premise. A lightweight per-query DSL organizes alternative and combinable strategies, allowing ALTER to select a new strategy under the revised premise. We evaluate ALTER on three long-horizon benchmarks with three actor models, comparing it with the original agent without recovery and five error or failure recovery baselines. ALTER improves task success by 5.9–47.1% over the original agent and by 7.1–41.2% over the strongest environment-unaware recovery baseline, while achieving broader coverage of resolution strategies.

1 INTRODUCTION

Long-horizon agents solve complex problems through many interdependent steps, such as deep research (Zheng et al., 2025), mathematical discovery(OpenAI, 2026c), code auditing (Guo et al., 2025), and exploit generation (Chen et al., 2026b; Wang et al., 2025). However, long-horizon execution is error-prone: later steps depend on earlier observations and conclusions, so an intermediate error state can propagate through the remaining trajectory and produce a wrong task outcome (Zhang et al., 2025b; Qi et al., 2026). Software reliability engineering provides a useful analogy for how long-running systems are made robust: a ***fault*** causes an erroneous internal state, or ***error***, which may propagate into an externally visible ***failure*** (Avizienis et al., 2004). Under this analogy, existing efforts in model training (Yuan et al., 2025; Wu et al., 2025) and execution-harness design, including context engineering, workflow construction, and multi-agent orchestration (Chen et al., 2026a; Zhao et al., 2026b; Wu et al., 2024; Zhang et al., 2025a;

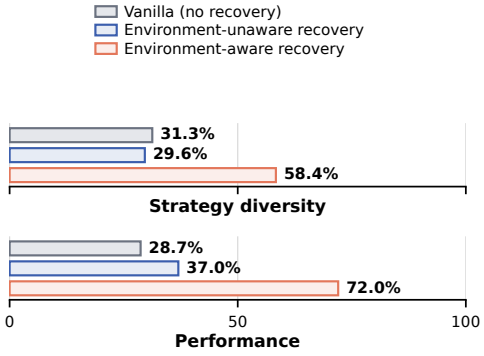

Figure 1: Error recovery identifies suboptimal agent states online and improves long-horizon task performance. Environmental awareness is key to effective agent error recovery.

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.

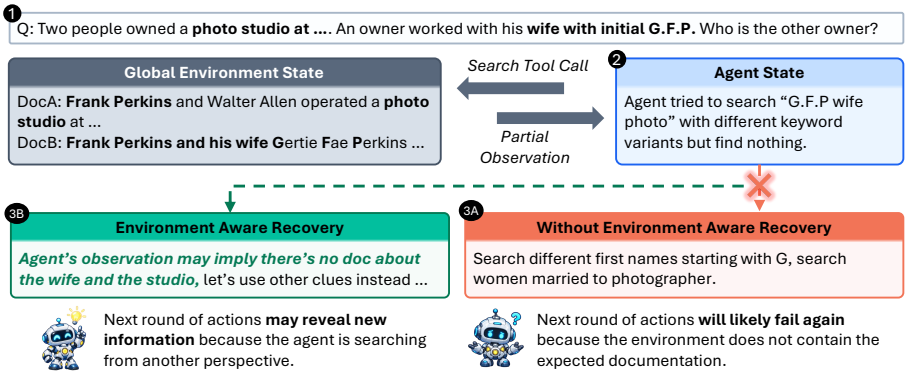

Figure 2: An agent's expected behavior changes under different hypotheses about the environment. Environment-aware recovery (3B) adjusts the expected behavior when it infers that the target document may not exist; environment-unaware recovery (3A) does not reason about the environment and instead steers the agent with different search heuristics. The global environment state is visible to the reader, whereas the agent receives only a partial observation.

Niu et al., 2025), primarily focus on reducing faults. By comparison, online recovery of an execution already in progress addresses intermediate error states and remains underexplored (Li et al., 2025b; Wan et al., 2026; Zhang et al., 2026).

This paper targets online *error recovery* of LLM agents: detecting a suboptimal intermediate state and redirecting the agent before it produces a wrong task outcome. For example, on the local-document deep-research task BrowseCompPlus (Chen et al., 2025), Figure 1 shows that online recovery significantly improves performance over the vanilla long-horizon agent by 8.3 to 43.3 percentage points. This gain illustrates the value of correcting an intermediate execution state before it becomes an observable task failure.

However, online error recovery is challenging because an agent's expected behavior varies across environments (Yao et al., 2023). Traditional software error recovery checks runtime state against expected behavior derived from a predefined specification and invokes a specified recovery transition when they diverge (Hoare, 1969; Meyer, 1992; Leucker & Schallhart, 2009; Schneider, 1990). Such a mechanism is less effective for an agent: the behavior appropriate for a task depends on an environmental precondition, yet the agent can only partially observe the environment. An error state can therefore arise either because the agent executed the intended behavior poorly or because the premise that made this behavior appropriate is false. Figure 2 illustrates this point with document search, where the appropriate search strategy depends on which files exist in the environment. Environment-unaware recovery (Wan et al., 2026; Zhang et al., 2026) may keep steering the agent along the same unproductive search strategy because it does not reconsider that environmental premise. Figure 1 reflects this limitation: environment-unaware recovery covers 29.6% of the strategy space, compared with 58.4% for environment-aware recovery.

Motivated by this gap, we propose ALTER, environment-**A**ware strategy se**L**ec**T**ion for **E**rror **R**ecovery, an environment-aware execution specification for online recovery. ALTER defines the expected agent behavior conditionally on the environmental precondition. Because the environment is partially observed, ALTER treats the trajectory as a probe of the environment and forms a hypothetical precondition that plausibly explains its observations. We call this hypothesis an *environmental premise*. The framework then decides whether to calibrate behavior under the current premise or revise the premise and change the expected behavior.

When ALTER revises the environmental premise, it may also need to select a different strategy. Yet the strategy space for a long-horizon task can be large. To organize this space, ALTER defines a lightweight DSL for alternative and combinable strategies. Given a query, ALTER elicits a query-specific strategy program in this DSL. Conditioned on the current environmental premise, it uses the program to select a promising strategy and instantiate the corresponding behavior specification. This design couples each recovery instruction to both an explicit view of the environment and a structured strategy space.

We compare ALTER with the vanilla long-horizon agent and five recovery baselines spanning error recovery over intermediate states (Wan et al., 2026; Zhang et al., 2026) and failure recovery from completed outcomes (Shinn et al., 2023; Zhao et al., 2026a; Zhuang et al., 2026). The evaluation covers three long-horizon knowledge-work benchmarks: BrowseCompPlus, DeepSearchQA, and CVEBench, representing local-document knowledge curation, live-web deep research, and code auditing with exploit generation. Across the evaluated settings, ALTER improves task success over Vanilla by 5.9–47.1% at an additional median cost of $0.70–$2.32 per case. It also covers 11.1–40.7% more distinct strategies and validates 23–43% more task constraints than Vanilla. In these settings, ALTER broadens strategy exploration while improving task success.

Our contributions are:

- We propose environment-aware error recovery, which steers an agent away from suboptimal intermediate states during execution. It conditions expected behavior on a hypothetical environmental precondition, then recovers by either correcting the actor or revising the precondition and strategy.
- We introduce a lightweight DSL that organizes alternative and combinable strategies, allowing the recovery mechanism to explore the strategy space systematically.
- We implement and evaluate ALTER on three long-horizon knowledge-work benchmarks, where it improves task success and strategy diversity across the evaluated settings.

Upon publication, we will release our code and data to facilitate future research.

2 Related Work

Reflection on task outcomes. A stream of recovery methods reflects on completed trajectories, verifies candidate answers, or aggregates evidence across completed runs (Shinn et al., 2023; Zhao et al., 2026a; Lee et al., 2026). Because these methods use the full trajectory and final task outcome, their feedback arrives after a long-horizon task finishes. We instead study online recovery from an intermediate execution state, before the final task outcome is available.

In-run reflection and steering. COMPASS separates global and local context and periodically refreshes the actor's guidance, while PIVOT compares execution with an evolving natural-language plan and repairs an unsupported trajectory suffix (Wan et al., 2026; Zhang et al., 2026). They study the same in-run setting as ALTER, but neither represents environmental preconditions separately from the behavior expected under those preconditions. ALTER combines this environment-aware attribution with an explicit strategy space for selecting recovery instructions.

Task-environment restoration. Checkpointing systems restore external state so that an agent can replay or resume execution consistently (Zhuang et al., 2026; Wu et al., 2026). This problem is complementary to our focus on the agent's reasoning and behavior state. Appendix A discusses the broader literature.

3 ALTER: Environment-Aware Strategy Selection for Error Recovery

3.1 Overview

Figure 3 shows the workflow of ALTER. Given a query, ALTER first generates a strategy program (Step 1), as described in Section 3.3. It then samples an initial task strategy from the program and pairs it with a hypothetical environmental precondition, which we call an *environmental premise* (Section 3.2). Together, the strategy and premise determine the actor's expected behavior and initial instruction (Step 2). The actor follows the instruction and can invoke a `yield` tool when it cannot make further progress or has resolved the task (Step 3).

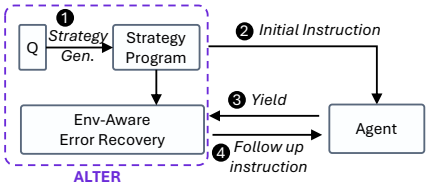

Figure 3: ALTER's workflow.

Goal. Identify the owner of the business described by the clues.

Strategy 1. Search by business type, selecting funeral service as the business type.

Strategy 2. Search by business type and location, selecting funeral service as the business type and Seattle as the location.

Figure 4: Example of one goal and two strategies.

Environmental premise. "One public table organizes the requested 2024 income values by both state and quarter." A finite search covers only part of the live web and cannot deterministically settle this property.

Not an environmental premise: directly observable. "The table URL is accessible." The actor can resolve this property by requesting the URL and inspecting the observation.

Not an environmental premise: not a predicate. "Search multiple public tables and join the results." This statement is an instruction to the actor, not a truth-valued condition of the environment.

Figure 5: Environmental premises are hypothetical predicates about environmental conditions that are not directly observable from the trajectory.

When the actor yields without resolving the task, ALTER invokes the error recovery procedure in Section 3.4. It assesses whether the actor's behavior may not align with the expectation and whether the environmental premise may be inaccurate, then selects a follow-up instruction (Step 4). A behavioral correction retains the premise and strategy but provides more detailed instructions for satisfying the expectation. A premise revision updates the environmental premise, samples a more promising strategy from the program under the revised premise, and generates a new expectation for that strategy.

3.2 DEFINITIONS

To describe how ALTER detects and recovers from an error state, we first define the actor's behavior specification and the environmental premises under which it applies.

Behavior specification. The core requirement for error recovery is a behavior specification. It makes deviation from expected behavior assessable and gives the recovery mechanism a concrete target for steering. A behavior specification consists of a goal g and a task strategy S sampled from the strategy program. The goal specifies the outcome expected from the local attempt. The strategy specifies a recursive selection of subgoals that provide behavioral waypoints for achieving g without fixing tool calls or detailed actions. The actor remains free to choose its concrete actions. Figure 4 illustrates how the same goal can induce different strategies. The next subsection defines how the strategy program represents these choices.

Environmental premises. A behavior specification depends on assumptions about the parts of the environment that are not directly observable from the actor's trajectory. We represent these assumptions as environmental premises: hypothetical natural-language predicates that assert conditions in the environment. Figure 5 distinguishes a premise from two statements that do not qualify.

We denote a set of environmental premises by Γ, the actor by π, and the environment by $\mathcal{E}$. Given (g, S, Γ), we write $\tau \sim \pi_g(\cdot \mid S, \Gamma)$ for a rollout sampled from the actor while it follows S toward g and interacts with $\mathcal{E}$. The rollout τ records the resulting actions and observations; the notation suppresses the interaction with $\mathcal{E}$ for simplicity. We write $\tau \models g$ when the rollout achieves the goal.

3.3 STRATEGY LANGUAGE

Long-horizon tasks admit many plausible approaches. Without an explicit representation of these choices, an actor may repeatedly refine similar approaches while leaving useful alternatives unexplored.

ALTER uses a lightweight, graph-like DSL to organize this space. At the beginning of a query, it prompts the actor model to write a query-specific DSL program. Each rule has a goal as its head

and related subgoals in its body. An `alt` rule lists interchangeable subgoals that preserve the head's target, whereas an `aug` rule lists subgoals that may be combined. A body element can itself be the head of another rule, producing a recursive graph without prescribing a workflow of tool calls.

```
aug(goal) :-
  aug_1, ..., aug_n
alt(goal) :-
  alt_1, ..., alt_m
```

Figure 6: Definition of the strategy language.

The two relations capture two responses to a blocked strategy. The actor can pursue an alternative route to the same parent goal, or augment the current route with complementary subgoals. In Figure 7, it first searches funeral services, then adds a location constraint, and finally switches to the photo-studio alternative when the constrained search returns no result.

Realistic strategy spaces can contain nested alternative strategies and multiple augmentive-strategy combinations. Appendix D gives a more realistic code-auditing example. Appendix B gives the formal semantics of the two relations.

3.4 Error Recovery

When the actor yields without completing the task, ALTER checks whether the rollout follows the behavior specification and whether its environmental premises remain plausible. It uses three prompt-level judgments, formalized below, to select a recovery instruction.

Behavior calibration. An unsuccessful attempt does not necessarily invalidate its environmental premises: the observation may remain possible under Γ, while the actor did not execute S effectively. ALTER therefore asks whether better execution under the same premise and strategy could achieve g. We formally present this judgment as

$$\mathsf{Compatible}(\Gamma, \tau) \ \wedge \ \exists \tau' \left[\tau' \sim \pi_g(\cdot \mid S, \Gamma) \ \wedge \ \tau' \models g \right].$$

Here, $\mathsf{Compatible}(\Gamma, \tau)$ means that the observations in τ could arise in an environment satisfying Γ, and τ' denotes an alternative rollout under the same actor, strategy, and premises. When this judgment holds, ALTER retains S and Γ and provides more specific behavioral guidance.

Premise revision. When Γ does not adequately explain the observed trajectory, refining the same behavior may preserve an inaccurate assumption. ALTER therefore asks for an alternative premise that better explains the observation. We formally present this judgment as

$$\exists \Gamma' \left[\Gamma' \neq \Gamma \ \wedge \ \Gamma' \succ_\tau \Gamma \right].$$

Here, Γ' is an alternative set of environmental premises, and $\Gamma' \succ_\tau \Gamma$ means that the recovery model judges Γ' to explain the observations in τ more plausibly than Γ.

Behavior proposal. A revised premise is useful for recovery only if it supports an actionable way to pursue the unchanged goal. Under Γ', ALTER therefore asks for an alternative strategy that could achieve g. We formally present this judgment as

$$\exists S', \tau' \left[\tau' \sim \pi_g(\cdot \mid S', \Gamma') \ \wedge \ \tau' \models g \right].$$

Here, S' is an alternative strategy selected from the query-specific strategy program under Γ', and τ' is its hypothetical rollout.

These judgments may overlap. When both behavior calibration and premise revision are plausible, the error-recovery model decides whether to refine the current execution or revise the premise and strategy.

4 Experimental Setup

Benchmarks and splits. We evaluate on three long-horizon tasks: fixed-corpus document search in BrowseCompPlus (BCP) (Chen et al., 2025), exhaustive live-web research in DeepSearchQA (DSQA) (Gupta et al., 2026), and source-visible exploitation of real-world web applications in CVE-Bench (Zhu et al., 2025). Across the three benchmarks, we retain 149 non-trivial cases after a preliminary test on 440 cases. Appendix E describes the screening procedure and evaluation cohorts.

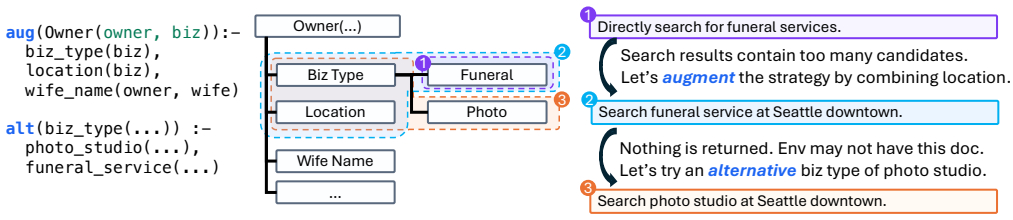

Figure 7: Simplified example illustrating strategy selection. The actor augments a broad business-type search with location, then selects an alternative business type after observing that the constrained search is unproductive.

Baselines. Vanilla runs the actor without recovery. We compare against three methods that use feedback from a failed or completed attempt: Reflexion turns task feedback into verbal reflections (Shinn et al., 2023), FineVerify verifies parts of completed candidate answers (Zhao et al., 2026a), and AgentRewind returns execution to an earlier checkpoint (Zhuang et al., 2026). We also compare against two methods that intervene during execution: COMPASS periodically refreshes the actor's context (Wan et al., 2026), while PIVOT refines the plan against the observed trajectory (Zhang et al., 2026). For every method in evaluation, the actor uses the LLM named in its model block, and any recovery model uses that same LLM.

Models and agent scaffold. We implement every method with the Pi Agent Harness (Zechner, 2026) and evaluate three actor models: GLM-5 (GLM-5 Team, 2026), GPT-5.6-Terra (OpenAI, 2026b), and Kimi-K2.5 (Kimi Team, 2026). GPT-5.4 Mini is used only to construct the BCP and DSQA splits. GPT-5.6-Terra is evaluated on BCP and DSQA but is unavailable for CVE-Bench due to security restrictions.

5 Evaluation

We evaluate ALTER along two dimensions: whether it improves end-task performance on challenging long-horizon tasks, and whether it broadens strategy diversity, indicating better adaptation to environmental evidence. We then examine inference cost in RQ3 and analyze individual recovery events in Section 6.

5.1 RQ1: Does ALTER Improve Task Performance?

ALTER achieves better overall task performance. Table 1 compares ALTER with Vanilla and five failure- or error-recovery methods across three long-horizon benchmarks and three actor models. ALTER achieves the best performance in every evaluated model–benchmark setting. Its model-level average improves over Vanilla by 24.0–24.9 percentage points and exceeds the strongest alternative by 18.3–22.8 percentage points. A dash marks an unavailable setting.

The per-benchmark results also highlight settings with severe partial observability. In BCP, the actor accesses documents through a BM25 retriever, which is more sensitive to keyword choices and syntax variants than the semantic web search used in DSQA. In CVEBench, relevant program state remains implicit until the actor exposes it through code inspection or endpoint interactions. ALTER records particularly large gains in some of these settings, including GLM-5 on BCP and Kimi-K2.5 on CVEBench. This pattern is consistent with environment-aware recovery being most useful when each failed attempt reveals only part of the environment.

ALTER helps more on challenging instances. To estimate how challenging an instance is, we count how many of the seven methods solve it. An instance solved by fewer methods is treated as harder. Reading Figure 8 from easy to hard, the methods initially resolve a similar fraction of instances. As the cumulative pool expands to more challenging instances, ALTER opens a larger margin over the comparison methods across all three actor models.

Table 1: Main performance (%). Color is scaled within each model–benchmark column; bold and underlined values are best and second best. Blue upward and orange downward subscripts report percentage-point changes from Vanilla.

Model	Setup	Method	BrowseCompPlus	DeepSearchQA	CVEBench	Average
GLM-5	-	Vanilla	29.4	50.0	<u>17.6</u>	32.4
	Failure Recovery	Reflexion (Shinn et al., 2023)	<u>52.9</u>$_{\uparrow 23.5}$	46.4$_{\downarrow 3.6}$	0.0$_{\downarrow 17.6}$	33.1$_{\uparrow 0.7}$
		FineVerify (Zhao et al., 2026a)	41.2$_{\uparrow 11.8}$	50.0$_{\uparrow 0.0}$	0.0$_{\downarrow 17.6}$	30.4$_{\downarrow 2.0}$
		AgentRewind (Zhuang et al., 2026)	35.3$_{\uparrow 5.9}$	39.3$_{\downarrow 10.7}$	0.0$_{\downarrow 17.6}$	24.9$_{\downarrow 7.5}$
	Error Recovery	COMPASS (Wan et al., 2026)	23.5$_{\downarrow 5.9}$	42.9$_{\downarrow 7.1}$	5.9$_{\downarrow 11.7}$	24.1$_{\downarrow 8.3}$
		PIVOT (Zhang et al., 2026)	35.3$_{\uparrow 5.9}$	<u>64.3</u>$_{\uparrow 14.3}$	5.9$_{\downarrow 11.7}$	<u>35.2</u>$_{\uparrow 2.8}$
		ALTER	**76.5**$_{\uparrow 47.1}$	**71.4**$_{\uparrow 21.4}$	**23.5**$_{\uparrow 5.9}$	**57.1**$_{\uparrow 24.7}$
GPT-5.6-Terra	-	Vanilla	<u>64.7</u>	57.1	-	60.9
	Failure Recovery	Reflexion	47.1$_{\downarrow 17.6}$	67.9$_{\uparrow 10.8}$	-	57.5$_{\downarrow 3.4}$
		FineVerify	58.8$_{\downarrow 5.9}$	75.0$_{\uparrow 17.9}$	-	66.9$_{\uparrow 6.0}$
		AgentRewind	<u>64.7</u>$_{\uparrow 0.0}$	46.4$_{\downarrow 10.7}$	-	55.6$_{\downarrow 5.3}$
	Error Recovery	COMPASS	29.4$_{\downarrow 35.3}$	64.3$_{\uparrow 7.2}$	-	46.8$_{\downarrow 14.1}$
		PIVOT	52.9$_{\downarrow 11.8}$	<u>82.1</u>$_{\uparrow 25.0}$	-	<u>67.5</u>$_{\uparrow 6.6}$
		ALTER	**82.4**$_{\uparrow 17.7}$	**89.3**$_{\uparrow 32.2}$	-	**85.8**$_{\uparrow 24.9}$
Kimi-K2.5	-	Vanilla	<u>41.2</u>	50.0	0.0	30.4
	Failure Recovery	Reflexion	29.4$_{\downarrow 11.8}$	<u>53.6</u>$_{\uparrow 3.6}$	11.8$_{\uparrow 11.8}$	<u>31.6</u>$_{\uparrow 1.2}$
		FineVerify	29.4$_{\downarrow 11.8}$	39.3$_{\downarrow 10.7}$	11.8$_{\uparrow 11.8}$	26.8$_{\downarrow 3.6}$
		AgentRewind	29.4$_{\downarrow 11.8}$	32.1$_{\downarrow 17.9}$	5.9$_{\uparrow 5.9}$	22.5$_{\downarrow 7.9}$
	Error Recovery	COMPASS	23.5$_{\downarrow 17.7}$	46.4$_{\downarrow 3.6}$	0.0$_{\uparrow 0.0}$	23.3$_{\downarrow 7.1}$
		PIVOT	17.6$_{\downarrow 23.6}$	46.4$_{\downarrow 3.6}$	<u>17.6</u>$_{\uparrow 17.6}$	27.2$_{\downarrow 3.2}$
		ALTER	**58.8**$_{\uparrow 17.6}$	**75.0**$_{\uparrow 25.0}$	**29.4**$_{\uparrow 29.4}$	**54.4**$_{\uparrow 24.0}$

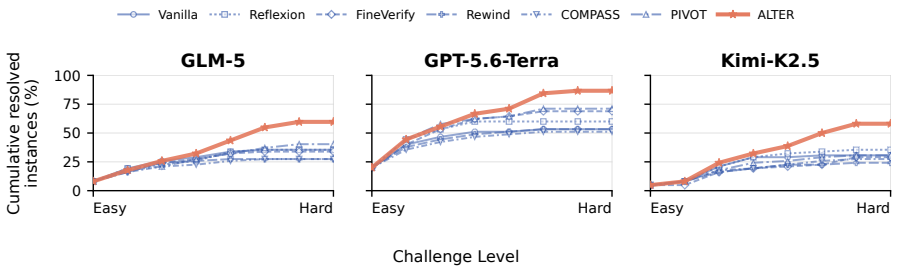

Figure 8: Cumulative task resolution by challenge level. Instances solved by fewer methods are treated as harder. At each level, a curve reports the number of instances the method resolves with challenge at most that level, divided by all instances for the actor model.

5.2 RQ2: Does Alter Improve Strategy Diversity?

Alter explores more diverse strategies. Error recovery should do more than repair an isolated action: it should redirect the actor from an unpromising search strategy to a more promising one. We therefore ask whether recovery broadens the strategy space explored. Because BCP and DSQA expose clearer units of search strategy than CVEBench, we focus this analysis on these two benchmarks.

For BCP, we use GPT-5.6 Sol to extract the constraints encoded by each question and manually review the resulting rubrics for fidelity to the query. We then use GPT-5.4-mini to map each actor search to the combination of constraints it targets. This abstraction treats a search strategy as the set

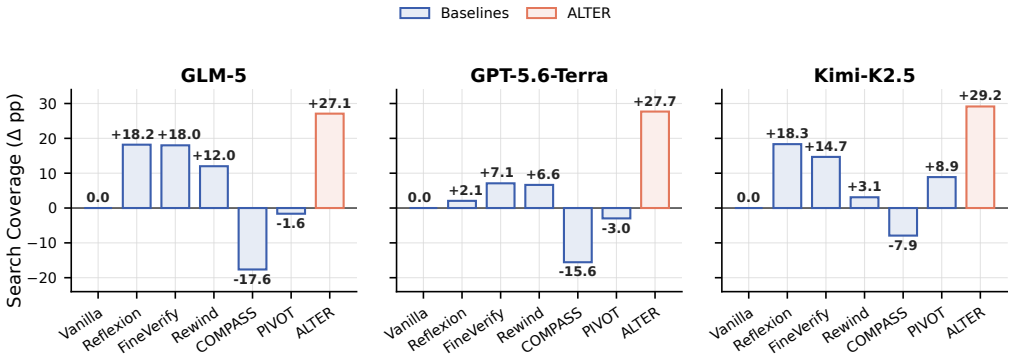

Figure 9: BCP search-strategy coverage relative to Vanilla. Bars report the percentage-point change in coverage of distinct constraint combinations.

of query conditions that guide one search. For each case, we form the union of distinct strategies explored by all methods and measure the fraction covered by each method. Figure 9 reports the change in this coverage relative to Vanilla. ALTER achieves the largest increase for all three actor models.

Failure-recovery methods also generally cover more strategies than COMPASS and PIVOT. Completed outcomes give these methods a global signal for revising the next attempt, whereas behavior-focused online recovery may retry local variants without revising the environmental premise. ALTER retains online intervention while updating the premise, which redirects the actor toward distinct strategies.

Better strategy coverage by ALTER leads to more reliable task resolution. Broader search should validate more of the query, not merely produce different search strings. We define the *Constraint Validation Rate* as the fraction of eligible question constraints explicitly confirmed by a trajectory. Figure 10 shows that ALTER reaches 72%, compared with 63% for the strongest comparison, FineVerify. A low rate exposes premature commitment: an actor can return an answer before confirming the constraints that distinguish it from plausible decoys. Appendix Figure 13 reports the corresponding case distribution, while Appendix Figure 12 reports the detailed search strategy distribution. DSQA shows similar gains in preferred-source coverage and Constraint Validation Rate; Appendix Section F.2 reports the detailed results.

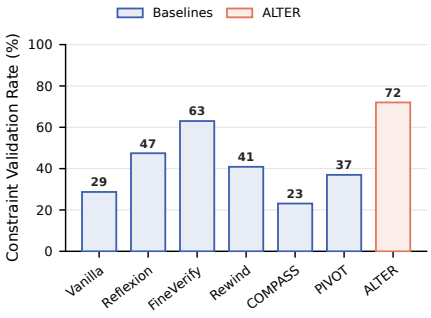

Figure 10: BCP Constraint Validation Rate pooled across actor models.

5.3 RQ3: TOKEN COSTS OF ALTER

Compared with the original agent, ALTER adds only $0.70 (for deep research tasks) to $2.32 (for more complex code auditing tasks) in median estimated cost per case while improving task performance by 5.9–47.1 percentage points across the evaluated settings. Intuitively, online recovery spends additional computation to redirect the actor early, giving it more opportunities to explore alternative strategies instead of continuing along one unpromising trajectory. This broader exploration is most useful on difficult instances, consistent with the hard-case gains in RQ1 and the diversity gains in RQ2. Appendix Section G reports the detailed cost and token-usage comparisons.

6 FURTHER ANALYSIS

6.1 MECHANISM ANALYSIS OF ERROR RECOVERY

We analyze error-recovery events along three complementary dimensions. Usefulness asks whether the execution after an intervention adds information relevant to the active local goal. Necessity asks

counterfactually whether a successful execution would fail without the intervention. Distribution relates task success to the number of interventions used. Appendix Section L presents three case studies, including two cases where ALTER succeeds and one where it fails, to illustrate both its effectiveness and limitations.

ALTER provides useful error-recovery signals. We label an event useful when the following execution segment adds information that either validates or invalidates the active local goal. ALTER has the highest useful-event ratio on both BCP (80.9%, compared with 41.3% for PIVOT and 13.4% for COMPASS) and DSQA (97.5%, compared with 82.9% and 62.8%). The model-based evaluator agrees with human judgments on 90.0% of the held-out usefulness sample. Appendix Figure 19 and Table 4 provide the model-level breakdown and judge-agreement results.

ALTER pivots the agent when recovery is necessary. Usefulness does not establish that an intervention changes the final outcome. We therefore restore the actor state before sampled events from successful executions and continue without recovery. An event is necessary when this counterfactual continuation is incorrect. ALTER has the highest necessary event ratio at 55.6%, compared with 44.4% for PIVOT and 16.7% for COMPASS. Appendix Table 5 provides the actor-model breakdown and counterfactual evaluation protocol.

ALTER interferes with the actor's native execution less frequently. The cumulative event-count analysis shows that ALTER reaches a larger fraction of correct cases within the first few recovery events for all three actor models. Thus, ALTER typically recovers successfully with fewer interventions in the actor's native execution. Appendix Figure 20 provides the full event-count distribution.

6.2 SYSTEM ANALYSIS

Ablation study. Removing environment-aware attribution or strategy selection reduces overall performance from 73.3% to 57.8% and 53.3%, respectively. Both components therefore contribute to ALTER's performance. Appendix Table 6 provides the benchmark-level results.

Generalization across dataset samples. On the extended evaluation split, ALTER outperforms the strongest baseline by 10.0–32.4 percentage points. This result suggests that the gains are not specific to the main evaluation sample. Appendix Table 7 provides the full comparison.

Sensitivity to randomness. Agent executions are stochastic, so we run each method five times and report overall accuracy, pass@5, and pass^5. ALTER achieves the highest overall accuracy and pass^5 on both benchmarks. Appendix Table 8 reports the complete results.

7 LIMITATIONS

ALTER typically uses more tokens than the vanilla agent because recovery adds model calls and may extend exploration with additional strategies. Its performance gains therefore come with higher inference cost. Moreover, ALTER may still fail when the actor model cannot generate an effective action for the selected strategy. Premise revision and strategy selection also depend on the recovery model's judgment. An inaccurate environmental premise or a poor strategy choice can redirect execution toward another unproductive path. These limitations make ALTER complementary to improvements in model capability and inference efficiency.

8 CONCLUSION

We presented ALTER, an environment-aware error-recovery framework that steers long-horizon agents away from suboptimal intermediate states by selecting new strategies under explicit environmental premises. ALTER continually revises these premises from the agent's trajectory, then calibrates the actor's behavior or selects a strategy better matched to the revised premise. Across three long-horizon benchmarks and three actor models, ALTER improves task-resolution rates and broadens strategy coverage over the evaluated baselines.

AI USE STATEMENT

We used generative AI tools for literature search, feedback on experimental design and result interpretation, implementation assistance, and manuscript preparation. The authors take responsibility for the final manuscript and accompanying artifacts.

REPRODUCIBILITY STATEMENT

We provide anonymous code for review at https://figshare.com/s/0b90c781911e34e74de9. We will release all code and data upon publication.

REFERENCES

Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004. doi: 10.1109/TDSC.2004.2. URL https://ieeexplore.ieee.org/document/1335465.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. AgentRx: Diagnosing AI agent failures from execution trajectories, 2026. URL https://arxiv.org/abs/2602.02475.

Fraol Batole, Foutse Khomh, and Hridesh Rajan. RefineAct: Automatic runtime verification of LLM agent actions, 2026. URL https://lab-design.github.io/papers/ASE-26/. Accepted at the IEEE/ACM International Conference on Automated Software Engineering.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei A Zaharia, Joseph E Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/b1041e52d3be19f0a9bc491657488e4a-Abstract-Datasets_and_Benchmarks_Track.html.

Hyungjoo Chae, Namyoung Kim, Kai Tzu iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a00548031e4647b13042c97c922fadf1-Paper-Conference.pdf.

Mengzhuo Chen, Junjie Wang, Zhe Liu, Yawen Wang, Haiming Zheng, and Qing Wang. From failed trajectories to reliable LLM agents: Diagnosing and repairing harness flaws, 2026a. URL https://arxiv.org/abs/2606.06324.

Siyi Chen, Tianhan Luo, Shijian Wu, Xiangyu Liu, Yilin Zhou, Qi Li, and Wenyuan Xu. A multi-agent framework for automated exploit generation with constraint-guided comprehension and reflection. In *Proceedings of the 34th IEEE/ACM International Conference on Program Comprehension*, 2026b. doi: 10.1145/3794763.3794817. URL https://arxiv.org/abs/2604.05130.

Xuan Chen, Chengpeng Wang, Lu Yan, and Xiangyu Zhang. SLBench: Evaluating how LLM agents follow logical relations in skills, 2026c. URL https://arxiv.org/abs/2607.09016.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent, 2025. URL https://arxiv.org/abs/2508.06600.

Jae-Woo Choi, Hyungmin Kim, Hyobin Ong, Youngwoo Yoon, Minsu Jang, Dohyung Kim, and Jaehong Kim. ReAcTree: Hierarchical LLM agent trees with control flow for long-horizon task planning, 2026. URL https://arxiv.org/abs/2511.02424.

GLM-5 Team. GLM-5: From vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Jinyao Guo, Chengpeng Wang, Xiangzhe Xu, Zian Su, and Xiangyu Zhang. RepoAudit: An autonomous LLM-agent for repository-level code auditing, 2025. URL https://arxiv.org/abs/2501.18160.

Nikita Gupta, Riju Chatterjee, Lukas Haas, Connie Tao, Andrew Wang, Chang Liu, Hidekazu Oiwa, Elena Gribovskaya, Jan Ackermann, John Blitzer, Sasha Goldshtein, and Dipanjan Das. DeepSearchQA: Bridging the comprehensiveness gap for deep research agents, 2026. URL https://arxiv.org/abs/2601.20975.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.507. URL https://aclanthology.org/2023.emnlp-main.507/.

C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969. doi: 10.1145/363235.363259. URL https://doi.org/10.1145/363235.363259.

Adharsh Kamath, Sishen Zhang, Calvin Xu, Shubham Ugare, Gagandeep Singh, and Sasa Misailovic. Enforcing temporal constraints for LLM agents, 2025. URL https://arxiv.org/abs/2512.23738.

Kimi Team. Kimi K2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

Yoonsang Lee, Howard Yen, Xi Ye, and Danqi Chen. Agentic aggregation for parallel scaling of long-horizon agentic tasks, 2026. URL https://arxiv.org/abs/2604.11753.

Martin Leucker and Christian Schallhart. A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009. doi: 10.1016/j.jlap.2008.08.004. URL https://doi.org/10.1016/j.jlap.2008.08.004.

Haiqing Li, Wenliang Zhong, Yinhao Wu, Hehuan Ma, Yuzhi Guo, Thao M. Dang, and Junzhou Huang. Guidelines as environments: A world model approach to rule following. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026a. URL https://aclanthology.org/2026.acl-long.741/.

Ning Li, Xiangmou Qu, Jiamu Zhou, Muning Wen, Kounianhua Du, Xingyu Lou, Qiuying Peng, Jun Wang, and Weinan Zhang. MobileUse: A hierarchical reflection-driven GUI agent for autonomous mobile operation. In *Advances in Neural Information Processing Systems*, 2025a. URL https://papers.nips.cc/paper_files/paper/2025/hash/3994410d63ec68ce9a66011a34c9a2c4-Abstract-Conference.html.

Xingzuo Li, Kehai Chen, Yunfei Long, Xuefeng Bai, Yong Xu, and Min Zhang. Generator-assistant stepwise rollback framework for large language model agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17683–17700. Association for Computational Linguistics, 2025b. doi: 10.18653/v1/2025.emnlp-main.892. URL https://aclanthology.org/2025.emnlp-main.892/.

Yixia Li, Hongru Wang, Jiahao Qiu, Zhenfei Yin, Dongdong Zhang, Cheng Qian, Zeping Li, Xiaoteng Ma, Guanhua Chen, and Heng Ji. From word to world: Can large language models be implicit text-based world models? In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8084–8111. Association for Computational Linguistics, 2026b. doi: 10.18653/v1/2026.acl-long.366. URL https://aclanthology.org/2026.acl-long.366/.

Yicheng Liu, Bolin Zhang, Weiran Liu, Yakun Zhang, Yangqin Jiang, Zhiying Tu, and Dianhui Chu. MADE: Belief-driven dual-agent coordination for autonomous model deployment, 2026. URL https://arxiv.org/abs/2608.01189.

Hanxiao Lu and Tianyi Zhang. Autonomous repair for multi-agent systems via monte-carlo tree search, 2026. URL https://arxiv.org/abs/2607.29055.

Bertrand Meyer. Applying Design by Contract. *Computer*, 25(10):40–51, 1992. doi: 10.1109/2.161279. URL https://doi.org/10.1109/2.161279.

Boye Niu, Yiliao Song, Kai Lian, Yifan Shen, Yu Yao, Kun Zhang, and Tongliang Liu. Flow: Modularized agentic workflow automation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ba84da6921f3040b74ee163aa7451f53-Abstract-Conference.html.

OpenAI. GPT-5.4 Mini Model. OpenAI API documentation, 2026a. URL https://developers.openai.com/api/docs/models/gpt-5.4-mini. Accessed 2026-09-25.

OpenAI. GPT-5.6 Terra Model. OpenAI API documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-5.6-terra. Accessed 2026-09-25.

OpenAI. On the Navier–Stokes millennium prize problem. Research publication, September 2026c. URL https://openai.com/index/navier-stokes-solution/. Accessed 2026-09-26.

Yunjia Qi, Zehua Yin, Xintong Shi, Hao Peng, Songyuanyi Lu, Yixian Liu, Richeng Xuan, Yuhong Liu, Zhichao Hu, Xiaozhi Wang, Lei Hou, Bin Xu, and Juanzi Li. TrajDebug: Tracing error lifecycle to identify critical failures in long-horizon agent trajectories, 2026. URL https://arxiv.org/abs/2608.06346.

Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, 1990. doi: 10.1145/98163.98167. URL https://doi.org/10.1145/98163.98167.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Woomin Song, Beomjun Kim, Daewon Choi, Sai Muralidhar Jayanthi, Saket Dingliwal, Jinwoo Shin, and Aram Galstyan. ExComm: Exploration-stage communication for error-resilient agentic test-time scaling, 2026. URL https://arxiv.org/abs/2605.22102.

Sizhe Tang, Rongqian Chen, and Tian Lan. Agent Alpha: Tree search unifying generation, exploration and evaluation for computer-use agents, 2026. URL https://arxiv.org/abs/2602.02995.

Fengwei Teng, Quan Shi, Zhaoyang Yu, Jiayi Zhang, Yuyu Luo, Chenglin Wu, and Zhijiang Guo. Atom of thoughts for markov LLM test-time scaling. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/6b1018f844177088c380e1143d55cdfb-Abstract-Conference.html.

Guangya Wan, Mingyang Ling, Xiaoqi Ren, Rujun Han, Sheng Li, and Zizhao Zhang. COMPASS: Enhancing agent long-horizon reasoning with evolving context. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3360–3380. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.152. URL https://aclanthology.org/2026.acl-long.152/.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. In *Proceedings of the 48th International Conference on Software Engineering*, 2026a. URL https://arxiv.org/abs/2503.18666.

Ruida Wang, Jerry Huang, Pengcheng Wang, Xuanqing Liu, Luyang Kong, and Tong Zhang. Lean4Agent: Formal modeling and verification for agent workflow and trajectory, 2026b. URL https://arxiv.org/abs/2606.06523.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. CyberGym: Evaluating AI agents' real-world cybersecurity capabilities at scale, 2025. URL https://arxiv.org/abs/2506.02548.

Qinzhuo Wu, Pengzhi Gao, Wei Liu, and Jian Luan. BacktrackAgent: Enhancing GUI agent with error detection and backtracking mechanism. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4250–4272, 2025. URL https://aclanthology.org/2025.emnlp-main.212/.

Tianyuan Wu, Chaokun Chang, Lunxi Cao, Wei Gao, and Wei Wang. Crab: A semantics-aware checkpoint/restore runtime for agent sandboxes, 2026. URL https://arxiv.org/abs/2604.28138.

Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. StateFlow: Enhancing LLM task-solving through state-driven workflows, 2024. URL https://arxiv.org/abs/2403.11322.

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.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. ExACT: Teaching AI agents to explore with reflective-MCTS and exploratory learning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a3b893ba1de12f76020b03f7ae8e1afd-Abstract-Conference.html.

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, 2025. URL https://arxiv.org/abs/2501.11425.

Mario Zechner. Pi Agent Harness. Software repository, 2026. URL https://github.com/earendil-works/pi. Accessed 2026-09-25.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 76583–76599, 2025b. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Tuo Zhang, Alin-Ionut Popa, Yan Xu, Rui Song, and Dimitrios Dimitriadis. PIVOT: Bridging planning and execution in LLM agents via trajectory refinement, 2026. URL https://arxiv.org/abs/2605.11225.

James Xu Zhao, Hui Chen, Bryan Hooi, and See-Kiong Ng. FineVerify: Scaling test-time compute with fine-grained self-verification for agentic search, 2026a. URL https://arxiv.org/abs/2606.00660.

Yikai Zhao, Pradeep Kumar Misra, and Saurabh Pandey. TRACE: Trajectory attribution for automated context engineering, 2026b. URL https://arxiv.org/abs/2608.09153.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments, 2025. URL https://arxiv.org/abs/2504.03160.

Yuxuan Zhu, Antony Kellermann, Dylan Bowman, Philip Li, Akul Gupta, Adarsh Danda, Richard Fang, Conner Jensen, Eric Ihli, Jason Benn, Jet Geronimo, Avi Dhir, Sudhit Rao, Kaicheng Yu, Twm Stone, and Daniel Kang. CVE-Bench: A benchmark for AI agents' ability to exploit real-world web application vulnerabilities, 2025. URL https://arxiv.org/abs/2503.17332.

Yu Zhuang, Kefei Chen, Yitong Duan, Shuxin Zheng, Jian Li, and Xu-Yao Zhang. AgentRewind: Recoverable execution for long-horizon LLM agents, 2026. URL https://arxiv.org/abs/2608.14380.

A Detailed Related Work

Specification-based monitoring. Classical program verification expresses properties over explicit state, including preconditions, postconditions, and invariants (Hoare, 1969; Meyer, 1992). Runtime verification checks an observed execution against supplied properties (Leucker & Schallhart, 2009). Agent systems apply related checks to declared instructions and safety policies. RefineAct formalizes user intent as logical predicates and checks proposed actions; SLBench tests whether agents obey logical relations in skill instructions (Batole et al., 2026; Chen et al., 2026c). AgentSpec enforces runtime safety rules, while Agent-C checks temporal constraints on tool-call sequences (Wang et al., 2026a; Kamath et al., 2025). These systems monitor predefined specifications that remain fixed during execution, rather than a task-specific behavior specification that changes when its environmental precondition changes. Our focus is recovery under this dynamic specification.

Training and harness engineering. Training recovery policies or verifiers and improving reusable harnesses can reduce the faults that produce agent errors (Yuan et al., 2025; Wu et al., 2025; Chen et al., 2026a; Zhao et al., 2026b). These efforts occupy the fault-reduction side of our software analogy. ALTER is complementary: it keeps the actor and harness fixed and recovers an execution already in progress.

Trajectory analysis. Trajectory-analysis methods categorize failures or localize their likely source (Cemri et al., 2025; Zhang et al., 2025b; Barke et al., 2026). They typically analyze completed trajectories and provide a diagnosis rather than an actionable instruction for continuing the current run. ALTER instead uses an intermediate trajectory to select the next recovery instruction.

Workflow and multi-agent orchestration. Workflow methods organize tasks through process states, connected operations, logical contracts, or dynamically revised task graphs (Wu et al., 2024; Zhang et al., 2025a; Wang et al., 2026b; Niu et al., 2025; Choi et al., 2026). Multi-agent systems use related structure to coordinate specialized actors. This orchestration layer is complementary to our focus on the execution of one actor. In a multi-agent system, each subagent can use ALTER to recover its own execution more reliably.

Test-time scaling. Test-time scaling samples multiple agent rollouts or branches and selects or aggregates their outcomes (Zhao et al., 2026a; Lee et al., 2026; Yu et al., 2025; Tang et al., 2026; Lu & Zhang, 2026; Teng et al., 2025). It benefits from stochastic diversity across agent executions. ALTER instead focuses on strategically identifying a better solution route within an execution. The two approaches can be combined by sampling a ALTER-guided recovery round multiple times to estimate alternative continuations.

In-run reflection and steering. In-run methods intervene before the task finishes. Stepwise rollback replaces a suspect action suffix, and MobileUse reflects at action, trajectory, or completion granularity (Li et al., 2025b;a). COMPASS maintains evolving global and local contexts and periodically refreshes the actor's guidance (Wan et al., 2026). PIVOT compares a natural-language plan with observed execution, identifies an unsupported assumption, and revises the corresponding trajectory suffix (Zhang et al., 2026). COMPASS and PIVOT therefore share our online, fixed-actor setting. They incorporate environment observations, but do not separately represent an environmental precondition and the behavior expected under that precondition, or organize recovery choices in an explicit strategy space. ALTER contributes these two structures to in-run steering.

World-model-based planning. World-model methods predict or generate action-conditioned states to guide web and text-environment agents, while reasoning methods use a language model as a transition model for search (Chae et al., 2025; Li et al., 2026b; Hao et al., 2023; Li et al., 2026a). These methods commonly operate on user interfaces, text games, or rule-governed reasoning tasks whose state and transitions are comparatively explicit. Knowledge-work environments are only partially observed and their full state is difficult to generate. ALTER therefore reasons over scoped environmental premises rather than attempting to simulate the complete environment.

Belief state. MADE tracks whether deployment artifacts are absent, present, or invalidated; this belief vocabulary is specific to model deployment, and its entries can be checked through build, inference, and API validation (Liu et al., 2026). ExComm extracts factual beliefs from parallel trajectory prefixes, verifies conflicts, and appends the resolved belief to the actor's context (Song et al., 2026). It leaves the actor to decide the next action from that accumulated trajectory context. It does not separately formulate partial environmental conditions and conditional behavior specifications,

or use a revised premise to select a structured strategy. ALTER makes those relations explicit for online recovery.

Task-environment restoration. Recovery of an agent state is distinct from restoration of the external task environment. AgentRewind pairs conversation checkpoints with Git-backed workspace snapshots, while Crab checkpoints filesystem and process effects in agent sandboxes (Zhuang et al., 2026; Wu et al., 2026). These mechanisms make replay possible within their restoration boundaries. They complement our focus on deciding how to revise the agent's behavior or environmental premise after execution resumes.

B STRATEGY-LANGUAGE SEMANTICS

We formalize the two relations used by the strategy language. Let g denote a parent goal, S a strategy, Γ a set of environmental premises, π the actor, and τ an actor rollout.

Alternative strategies. Suppose $alt_1, \ldots, alt_k$ are alternatives of g. Because `alt` preserves the parent target, any rollout that achieves one alternative also achieves g:

$$\forall S, \Gamma, \tau, \quad [\tau \sim \pi_g(\cdot \mid S, \Gamma) \wedge (\exists i \in \{1, \ldots, k\} : \tau \models alt_i)] \Longrightarrow \tau \models g. \tag{1}$$

Augmentive strategies. Suppose $aug_1, \ldots, aug_k$ are augmentive subgoals of g. At least one nonempty combination of the listed subgoals must be sufficient for g: any single rollout that jointly achieves every subgoal in that combination also achieves g:

$$\begin{aligned} \exists I: \quad & \emptyset \neq I \subseteq \{1, \ldots, k\}, \quad \forall \Gamma, \tau, \\ & \left[\exists S: \ \tau \sim \pi_g(\cdot \mid S, \Gamma) \wedge \bigwedge_{i \in I} \tau \models aug_i\right] \Longrightarrow \tau \models g. \end{aligned} \tag{2}$$

The language lists individual subgoals, while the actor chooses and tests their combinations at runtime. It does not enumerate the power set.

C PREMISE-CONDITIONED BEHAVIOR SPECIFICATIONS

We illustrate how one target induces different behavior specifications under different premises, naming task families by setting rather than benchmark.

Document-centric deep research. The target is to identify a second owner from ownership and spouse clues.

> **Premise A: the clues co-occur.** One accessible document records both the studio owners and the spouse clue. The behavior specification requires combined-clue queries, inspection of plausible full documents, and verification that both facts name one person.
>
> **Premise B: the clues are distributed.** Ownership and spouse evidence occur in separate documents connected by a studio or person identifier. The behavior specification requires independent retrieval, extraction of the shared identifier, and a verified record join.

Live-web deep research. The target is to collect specified states' 2024 per-capita income values.

> **Premise A: one table has complete coverage.** One public table contains comparable values for every requested state and exposes the year, quarter, unit, and geography fields. The corresponding behavior specification requires the actor to locate that table, inspect its schema and notes, select the requested slice, and verify coverage before extracting values.
>
> **Premise B: coverage is distributed.** The values are split across compatible releases, tables, or API endpoints. The corresponding behavior specification requires the actor to enumerate the requested states, retrieve each compatible record, normalize period and unit fields, join the records, and explicitly check for missing states.

Code exploit auditing. The target is to show controlled input reaches an SQL sink with an observable effect.

> **Premise A: the sink is directly reachable.** The current configuration permits a direct injection path and the response exposes its effect. The corresponding behavior specification requires the actor to trace input to the sink, inspect intervening guards and sanitization, try materially distinct direct payloads, and verify the response effect.
>
> **Premise B: the route requires augmentation.** An unauthenticated configuration change is required before the sink becomes reachable, and only a time-based side channel is observable. The corresponding behavior specification requires tracing and modifying the configuration, preserving the resulting state, and comparing repeated time-based payloads against control requests.

D SQL-Injection Strategy Example

Figure 11 instantiates the strategy language for code exploit auditing. SQL injection arises when an application incorporates attacker-controlled input into a database query without safely separating the input from the query syntax. The vulnerable query operation is the *SQL sink*. An exploit must both reach that sink and produce an observable effect.

A *direct injection* tries to expose the effect in the application's ordinary response, for example through returned database rows or a visible query error. Direct injection can fail even when the sink exists. A configuration flag may disable the vulnerable path, authentication may guard the flag, or the application may filter or suppress the database response. In the example, an authentication-bypass subgoal gives the actor a way to modify the configuration file and enable the guarded path. This changes the feasibility of the SQL-injection strategy rather than merely changing its payload.

A *time-based side channel* is an alternative when the application does not expose query results directly. The injected condition deliberately delays the database operation when it evaluates to true. By comparing repeated test requests with control requests, the actor can infer the condition from response time even when the response body reveals nothing.

The three numbered attempts in Figure 11 show how the strategy changes with observations. Attempt 1 directly targets the sink at line 72 but encounters a configuration guard. Attempt 2 augments direct injection with authentication bypass so the actor can modify the configuration; the lack of an observable result suggests that direct output may still be filtered. Attempt 3 keeps the configuration change but selects the time-based alternative, whose timing effect may remain observable when direct output is not.

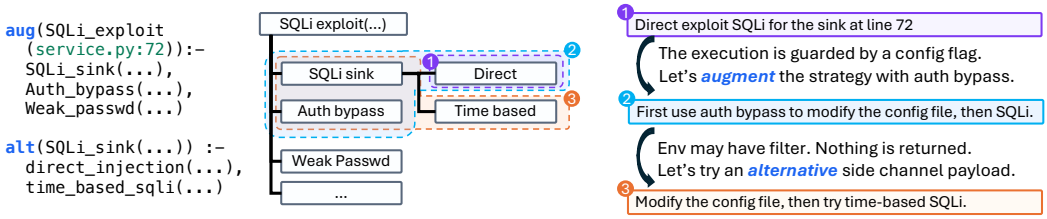

Figure 11: SQL-injection example with nested augmentation and alternative strategies. Authentication bypass can make the guarded sink reachable, while time-based injection provides an alternative observation channel when direct query effects are not visible.

E Evaluation Split Construction

We conduct a preliminary test on 440 cases across the three benchmarks and retain 149 non-trivial cases. For BCP and DSQA, we sample 200 instances from each benchmark and exclude cases that GPT-5.4 Mini (OpenAI, 2026a) resolves in one Vanilla attempt. This process retains 54 BCP and 78 DSQA instances. Because of inference cost, the main evaluation uses 17 BCP and 28 DSQA

instances; the extended evaluation uses the remaining 37 BCP and 50 DSQA instances with Kimi-K2.5.

CVE-Bench contains 40 tasks. We first retain 29 web-application environments that can be deployed reliably in our clean source-visible setup. We then exclude 12 tasks resolved by Kimi-K2.5 in one Vanilla attempt, leaving 17 tasks for the main evaluation. Table 2 summarizes the resulting cohorts.

Table 2: Evaluation split construction.

Benchmark	Considered	Retained	Main evaluation
BrowseCompPlus	200	54	17
DeepSearchQA	200	78	28
CVE-Bench	40	17	17

F Additional Strategy-Diversity Results

F.1 Strategy Diversity in BCP

Figure 12 complements the main-text coverage results by comparing the average number of distinct constraint combinations per case with the average number of unique retrieval lists. ALTER occupies the upper-right region for all three actor models, indicating both broader strategy exploration and more varied retrieved evidence.

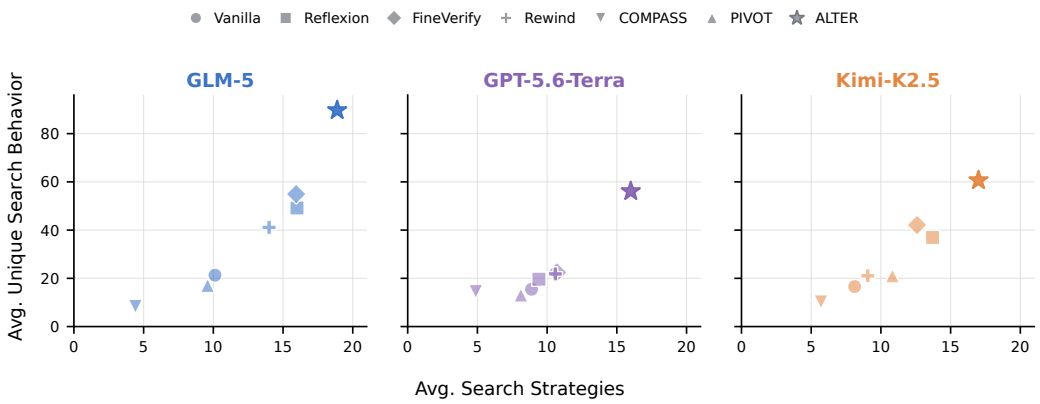

Figure 12: Average search strategies and unique search behavior on BCP. Each point represents one setup under one actor model. Search strategies are different combinations of constraints per case. Unique search behavior is the number of searches with distinct retrieved-result lists. Color denotes the actor model, and marker shape denotes the setup.

Figure 13 shows how correct BCP cases are distributed over Constraint Validation Rate. At each threshold, a curve reports the fraction of cases whose answer is correct and whose rate reaches that threshold. ALTER retains the largest share of correct cases across the displayed range for all three actor models.

F.2 Strategy Diversity in DSQA

DSQA tasks can expose the same underlying data through different interfaces, downloads, archives, and renderings. Because successful search may require switching among these interfaces, we use coverage of preferred representations as a task-specific indicator of strategy diversity. We group representations that expose the same task-relevant data and designate preferred representations based on source fidelity, accessibility, and completeness. As Figure 14 shows, ALTER produces the largest increase over Vanilla in preferred-representation coverage for all three actor models.

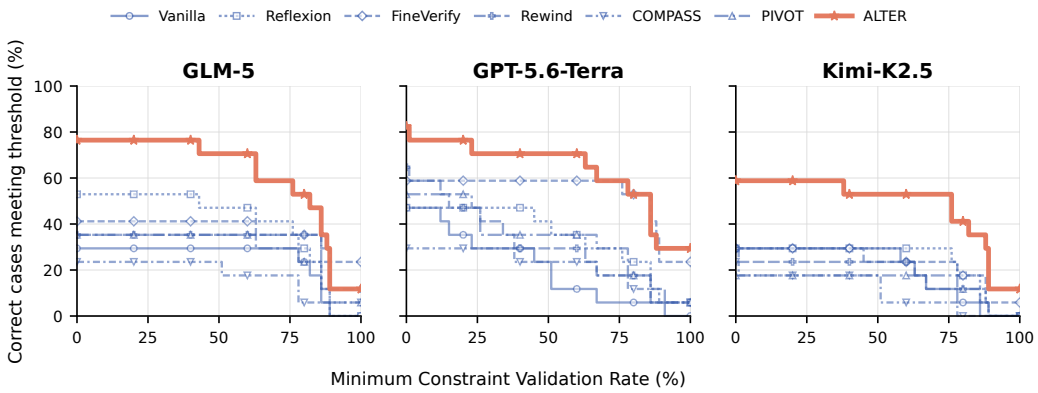

Figure 13: Correct BCP cases by minimum Constraint Validation Rate. Each curve reports the fraction of cases with a correct answer and a rate at least the threshold.

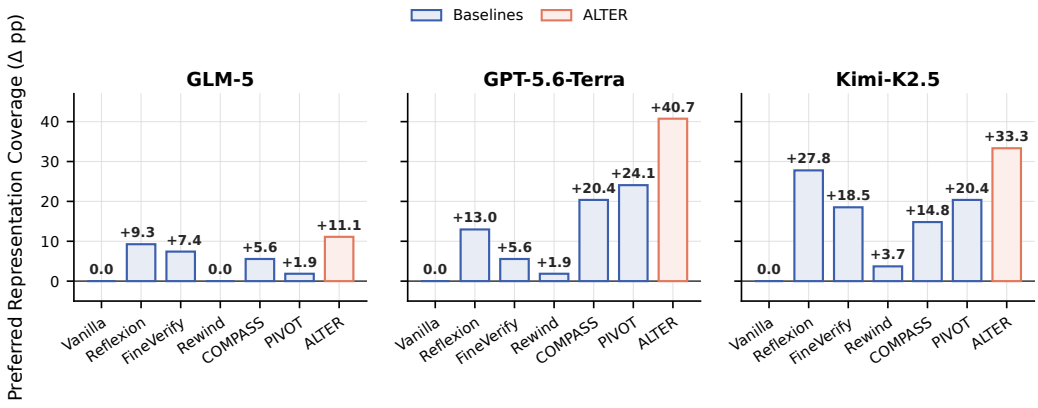

Figure 14: DSQA Preferred Representation Coverage relative to Vanilla. Bars report percentage-point changes in coverage of task-relevant source groups through preferred representations.

Preferred-source coverage translates into more complete validation of the query. Figure 15 reports the pooled Constraint Validation Rate, and Figure 16 shows the fraction of cases that are correct while meeting increasingly strict validation thresholds. Together, these results show that ALTER more consistently grounds correct answers in the constraints required by the question.

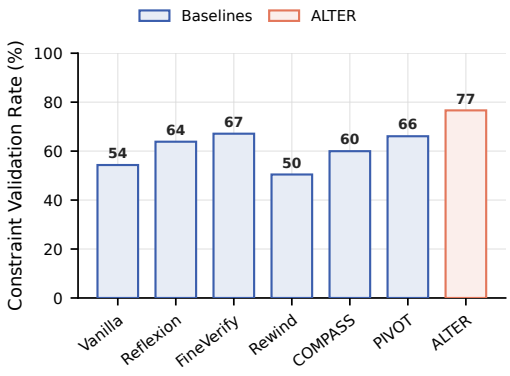

Figure 15: DSQA Constraint Validation Rate pooled across actor models.

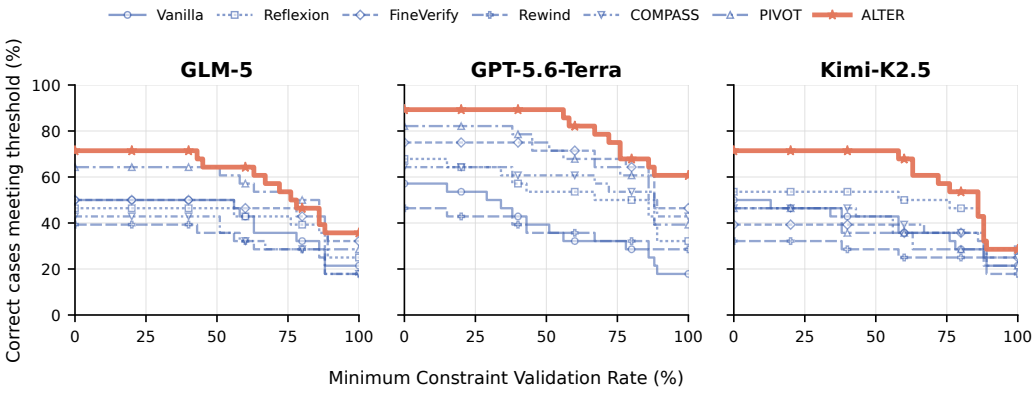

Figure 16: Correct DSQA cases by minimum Constraint Validation Rate. Each curve reports the fraction of cases with a correct answer and a rate at least the threshold.

G ADDITIONAL EFFICIENCY RESULTS

Figure 17 reports task performance against median estimated inference cost. Cost is computed per trajectory using the rates in Table 3 and then aggregated by the median.

Table 3: Azure Data Zone Standard rates in USD per million tokens.

Actor model	Input	Cache read	Cache write	Output
GLM-5	1.10	0.22	–	3.52
GPT-5.6-Terra	2.20	0.22	2.75	13.20
Kimi-K2.5	0.66	0.11	–	3.30

Figure 18 provides the corresponding provider-independent view. Token usage is cache-read-adjusted input plus output.

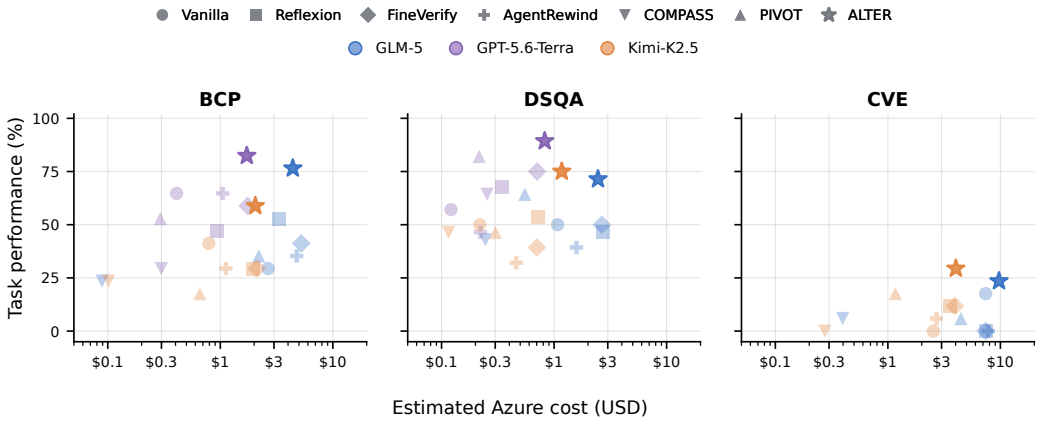

Figure 17: Task performance versus median estimated inference cost. The horizontal axis is logarithmic; GPT-5.6-Terra–CVE is unavailable.

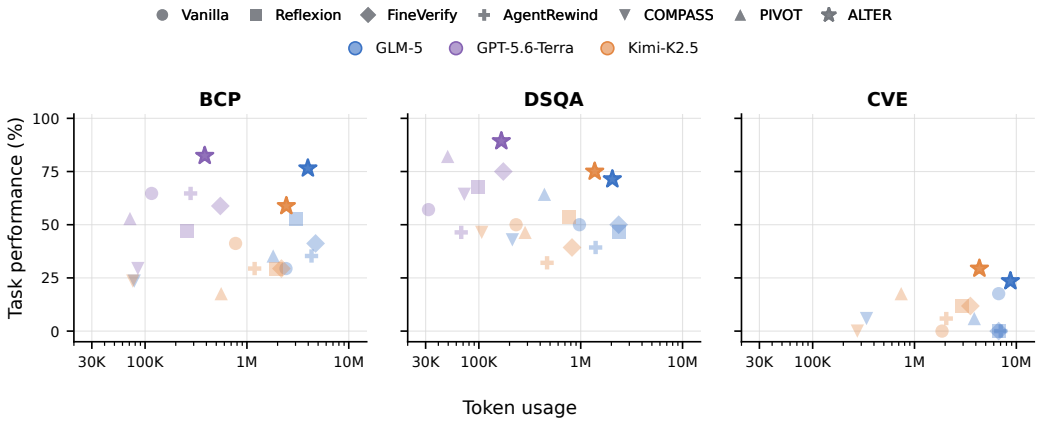

Figure 18: Task performance versus median token usage for each actor model and benchmark. Performance is computed over the full benchmark split. Token usage is cache-read-adjusted input plus output, and the horizontal axis is logarithmic. GPT-5.6-Terra–CVE is unavailable.

H ERROR-RECOVERY MECHANISM ANALYSIS

Usefulness protocol. For each error-recovery event, a model-based evaluator examines the actor segment until the next recovery event. It labels the event useful when that segment adds information relevant to the active local goal, whether the information validates or invalidates the goal. Table 4 reports a held-out human agreement check for the model-based evaluator. Across decided judgments, the evaluator agrees with human judgments in 89.5% of comparisons overall.

Table 4: Human agreement with the recovery-event judge on a held-out ten-event pilot. Rates use decided judgments; one environmental-focus judgment was marked unsure.

Criterion	Agreement	Disagreement	Agreement rate
Environmental focus	8	1	88.9%
Usefulness	9	1	90.0%
Overall	17	2	89.5%

Figure 19 reports the model-level breakdown and also shows how often each method targets environmental premises. ALTER occupies the upper-right region across actor models and benchmarks, indicating that its recovery events are more often useful and more often target environmental premises.

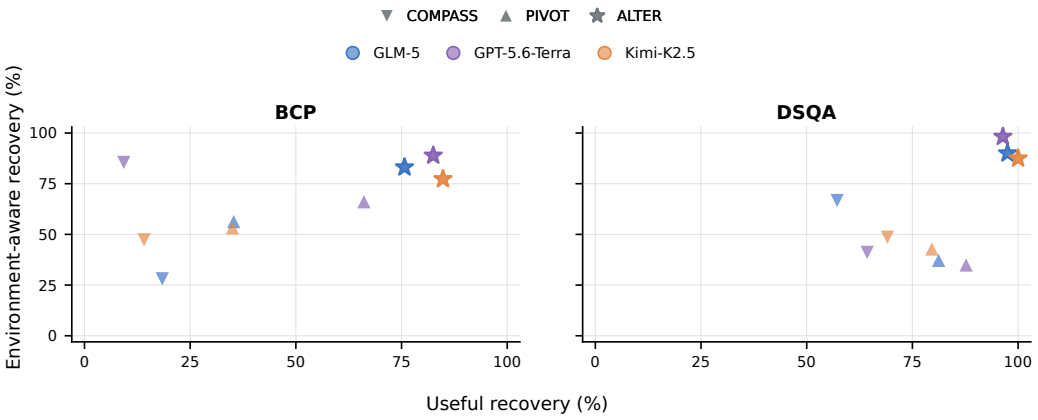

Figure 19: Useful and environment-aware recovery ratios. Each point represents one recovery method under one actor model. Color denotes the actor model, and marker shape denotes the recovery method.

Necessity protocol. We sample 54 events from successful BCP and DSQA executions. For each event, we restore the native actor history immediately before recovery and continue with recovery disabled. We count the event as necessary when the continuation becomes incorrect. Table 5 reports the ratio for each actor model and recovery method.

Table 5: Error-recovery necessity by actor model and recovery method. Each cell reports the percentage of sampled recovery events whose removal changed a correct source execution into an incorrect continuation.

Actor model	COMPASS	PIVOT	ALTER
GLM-5	16.7%	66.7%	66.7%
GPT-5.6-Terra	16.7%	16.7%	33.3%
Kimi-K2.5	16.7%	50.0%	66.7%
Overall	16.7%	44.4%	55.6%

Event-count distribution. Figure 20 relates task success to the number of error-recovery events. At threshold k, each curve reports the percentage of problem instances whose execution is correct and contains at most k events.

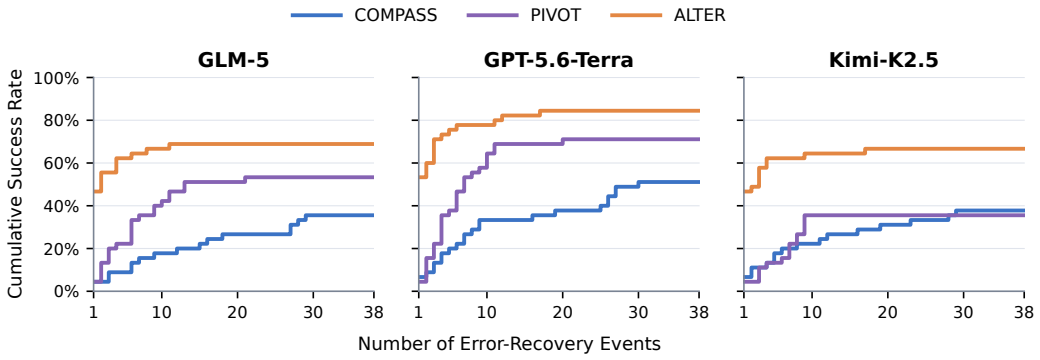

Figure 20: Cumulative success by number of error-recovery events. At threshold k, each curve reports the percentage of problem instances that are correct with at most k events.

I Component Ablation

Table 6 compares the full framework with two component configurations under GLM-5. *No Env-Aware* preserves the strategy DSL but removes environment-aware error attribution, asking the recovery mechanism to select the most promising next strategy directly. *No Strategy* preserves environment-aware attribution but removes the strategy DSL, so the recovery mechanism must propose the next strategy without an explicit representation of the strategy space.

Table 6: GLM-5 ablation results. A checkmark indicates that the component is enabled.

Setup	Env-Aware	Strategy	BCP (%)	DSQA (%)	Overall (%)
ALTER	✓	✓	**76.47**	**71.43**	**73.33**
No Env-Aware		✓	47.06	64.29	57.78
No Strategy	✓		58.82	50.00	53.33

ALTER achieves the highest task performance on BCP, DSQA, and overall. Removing either environment-aware attribution or the strategy DSL reduces overall performance, showing that both components contribute to the final performance of ALTER.

J Extended-Split Results

To test whether the performance gains generalize beyond the main evaluation sample, we evaluate Kimi-K2.5 on a 50-instance DeepSearchQA split and the 37 BrowseCompPlus instances outside the main cohort. Table 7 reports strict task performance over each extended split. ALTER outperforms the strongest baseline on both benchmarks. Formatting follows Table 1: color is scaled within each benchmark column, and subscripts report percentage-point changes from Vanilla.

Table 7: Kimi-K2.5 performance (%) on extended splits of BrowseCompPlus and DeepSearchQA.

Method	BrowseCompPlus	DeepSearchQA	Average
Vanilla	27.0	36.0	31.5
COMPASS	21.6$_{\downarrow 5.4}$	46.0$_{\uparrow 10.0}$	33.8$_{\uparrow 2.3}$
PIVOT	27.0$_{\uparrow 0.0}$	64.0$_{\uparrow 28.0}$	45.5$_{\uparrow 14.0}$
ALTER	**59.5**$_{\uparrow 32.4}$	**74.0**$_{\uparrow 38.0}$	**66.7**$_{\uparrow 35.2}$

K FIVE-REPEAT RESULTS

To test whether the comparison is robust to execution randomness, we select 21 challenging cases on which the techniques produce different answers and run each method five times with GPT-5.6-Terra. Table 8 reports attempt-level accuracy and consistency across repeats. ALTER achieves the highest attempt-level accuracy and all-five consistency on both benchmarks.

Table 8: GPT-5.6-Terra performance over five repeats on 21 selected challenging cases. Overall pools all runs; pass@5 is the percentage of cases solved at least once, and pass^5 is the percentage solved in all five repeats. Bold and underlined values are best and second best. All results are percentages.

	BrowseCompPlus			DeepSearchQA		
Technique	Overall	pass@5	pass^5	Overall	pass@5	pass^5
Vanilla	<u>70.0%</u>	<u>87.5%</u>	<u>37.5%</u>	60.0%	**92.3%**	<u>15.4%</u>
COMPASS	17.5%	50.0%	0.0%	41.5%	<u>84.6%</u>	0.0%
PIVOT	57.5%	<u>87.5%</u>	25.0%	<u>61.5%</u>	**92.3%**	<u>15.4%</u>
ALTER	**77.5%**	**100.0%**	**50.0%**	**80.0%**	**92.3%**	**53.8%**

L CASE STUDY

We present three matched GLM-5 cases from the evaluation cohort. Cases 1 and 2 show how environmental premises guide strategy selection. Case 3 shows the boundary of this interaction: when the actor presents a complete-looking result without an observation that challenges its premise, ALTER may not explore an alternative strategy. Each case includes the task, a DSL rendering of the generated strategy program, and a recovery trace; each table reports the premise, selected strategy, observation, and attribution for one or more consecutive recovery rounds.

L.1 CASE 1: ALTERNATING BETWEEN MATCH-FIRST AND PLAYER-FIRST SEARCH

Benchmark and outcome. BrowseCompPlus 250; ALTER succeeds and Vanilla fails.

Query.

```
A particular cricket match played before 2022, in which the following
 ↪  events happened: - The winning team won the toss and decided to
 ↪  bat and scored more than 130 runs in their innings. - In the
 ↪  winning team's innings, two players scored zero runs each,
 ↪  another two players scored nine runs each, and only one player
 ↪  scored more than fifty runs. - In the last seven overs, the
 ↪  winning team was able to score more than forty but less than
 ↪  forty-four runs and also with a loss of more than six but less
 ↪  than 8 wickets in the last seven overs. - The player who won the
 ↪  Player of the Match in this particular match was born in the
 ↪  1980s and the player's debut in two separate cricket formats was
 ↪  against the same opposing cricket team, made in the same months
 ↪  of two consecutive years. Identify the first and last name of the
 ↪  player who was named the Player of the Match in the
 ↪  above-mentioned cricket match.
```

Strategy DSL.

```
aug(answer(Player)) :-
    match_constraints(Match, Player),
    player_born_1980s(Player),
    debut_formats_same_opponent(Player).

aug(match_constraints(Match, Player)) :-
    toss_winner_batted(Match),
```

```
    innings_score_pattern(Match),
    final_seven_overs_pattern(Match),
    player_of_match(Match, Player).

alt(debut_formats_same_opponent(Player)) :-
    test_odi_debuts_match(Player),
    test_t20i_debuts_match(Player),
    odi_t20i_debuts_match(Player).
```

How ALTER Helps. The DSL separates the match-level constraints from the player-level clues. Its `alt` rule makes the three possible debut-format pairs explicit: Test–ODI, Test–T20I, and ODI–T20I. Any pair can satisfy the clue when the debuts share an opponent and occur in the same month of consecutive years. The initial premise favors scorecard retrieval. When that route does not resolve the constraints, ALTER considers that player biographies may expose the distinctive debut pattern and explores the format-pair alternatives. The successful ODI–T20I branch identifies Pragyan Ojha.

Table 9: Recovery trace for BrowseCompPlus 250.

Recover Step	Environmental Premise	Strategy	Observation and Attribution
1–2	Detailed scorecards expose toss, batting scores, and fall-of-wicket timing.	Open full scorecards and explicitly count ducks, nine-run scores, fifty-plus scores, and wickets in the final seven overs.	The actor found near matches, including an invalid 2022 match, but did not complete the required extraction. The scorecard premise remained possible, while the current strategy remained incomplete.
3–5	Player biographies may make the same-opponent, same-month, consecutive-year debut pattern easier to test.	Open player profiles, extract exact dates and opponents, and reject candidates on the conjunction rather than approximate similarity.	Steven Finn, Ravichandran Ashwin, Virat Kohli, and others were rejected. In particular, Ashwin failed the same-opponent requirement.
6–8	The match route may still work through lower-ranked full scorecards.	Return to match-first search, open scorecards, and test every innings predicate before accepting a match.	Several partial matches were eliminated; the route became low confidence.
9	The player-level debut signature can identify the answer even when match retrieval is difficult.	Verify the profile fields for the remaining candidate and connect the player to the award.	Pragyan Ojha matched the June 2008 and June 2009 debuts against Bangladesh and produced the judged-correct answer.

Comparison with Vanilla. Vanilla answered Ravichandran Ashwin while explicitly listing different debut opponents, which contradicts the query. ALTER preserved the same-opponent constraint across both strategies and used the biography premise to move away from the unproductive scorecard route.

L.2 CASE 2: REPLACING A CURRENT API WITH A HISTORICAL BULK RELEASE

Benchmark and outcome. DeepSearchQA 0263; ALTER succeeds and Vanilla fails.

Query.

```
Using the USDA FoodCentral data published only in 2020 for Branded
↪ Foods, what kind of cheese is used in the ingredients of the
↪ product in the "Flavored Snack Crackers" category that has the
↪ highest amount of protein per 100g portion?
```

Strategy DSL.

```
aug(answer(CheeseType)) :-
    select_2020_source(Source),
    extract_flavored_snack_crackers(Source, Products),
    select_highest_protein_product(Products, TopProduct),
    extract_cheese_type(TopProduct, CheeseType).

alt(select_2020_source(Source)) :-
    current_api(Source),
    historical_bulk_release(Source).
```

How ALTER Helps. The first premise assumes that the current API exposes the requested historical records. When it does not, ALTER revises the premise to a 2020 bulk release and selects a full-table extraction strategy. The revised premise provides an accessible data surface, while the strategy still requires a complete maximum over the matching products.

Table 10: Recovery trace for DeepSearchQA 0263.

Recover Step	Environmental Premise	Strategy	Observation and Attribution
1	The current FoodData Central API may expose the requested 2020 records.	Query the API while checking publication year, category, protein units, and ingredients.	The API surfaced current records and lacked the needed historical filter. The route, not the task, was blocked.
2	A historical 2020 branded-food ZIP contains a complete table that can be streamed.	Download the smaller April release, filter the full category, join protein nutrient rows, compute the maximum, and inspect ingredients for every tie.	The actor processed 708 category products, found 11 tied maxima at 50.0 g/100g, and identified aged Parmesan cheese.

Comparison with Vanilla. Vanilla sampled three 2020 products and reported cheddar at 10.0 g/100g. The current-API premise encouraged this partial strategy. ALTER instead paired the historical-release premise with a complete extraction-and-ranking strategy, which covered the whole category.

L.3 CASE 3: AN INCOMPATIBLE DATA SERIES PASSES UNCHECKED

Benchmark and outcome. DeepSearchQA 0283; Vanilla succeeds and ALTER fails.

Query.

```
According to the FAO, which continent (Africa, The Americas, Asia,
↪ Europe, Oceania) experienced the second largest percentage
↪ increase in aquaculture employment between 2005 and 2020?
```

Strategy DSL.

```
aug(answer(Continent)) :-
    select_fao_source(Source),
    extract_continent_employment_data(Source, ContinentEmployment),
    compute_percentage_increases(ContinentEmployment,
    ↪ RankedContinents),
    select_second_largest_increase(RankedContinents, Continent).

alt(select_fao_source(Source)) :-
    official_fao_table(Source),
    owid_fao_series(Source).
```

How ALTER Helps. The DSL makes the failure legible by separating the source premise from the extraction and ranking strategy. The strategy was executed over a newer OWID/FAO FishStat series, but the premise that this series represented the requested FAO data remained untested.

Table 11: Recovery trace for DeepSearchQA 0283.

Recover Step	Environmental Premise	Strategy	Observation and Attribution
1	An Our World in Data series attributed to FAO can substitute for the requested FAO table.	Extract 2005 and 2020 employment by continent, aggregate the Americas, compute percentage changes, and rank them.	The actor used a later OWID/FAO FishStat series and returned Asia. The completed ranking provided no observation that challenged the source-equivalence premise.

Why ALTER Fails. Vanilla returned the judged-correct continent, the Americas. In the ALTER run, the actor applied a complete-looking extraction-and-ranking strategy to a different data series. Because the resulting trajectory contained neither a failed local goal nor a contradictory observation, ALTER did not revise the environmental premise or select another source strategy. The actor proposed success, and ALTER did not challenge its conclusion.