FROM PLAUSIBLE REPAIRS TO POLICY CAPABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

A language model can propose a plausible correction for a failed attempt without knowing whether it will help the agent finish. Even when an intervention works, deciding what the policy should learn from it is a separate problem. We introduce RePair, a research harness that connects failure analysis to policy learning through same-state, paired environment tests. Only interventions that repeatedly improve terminal completion become positive training evidence. A dual-view representation pairs a deployable action with training-only supervision for subgoals, expected progress, and recovery; evaluation then removes external memory and intervention. An ALFWorld study with Qwen2.5-3B-Instruct exposes a gap between these stages. Five development rounds produce one stable beneficial intervention among 21 source-state experiments. A subsequent executable-strategy validation finds four among five selected states, but these successes are completed by the intervention programs themselves. Training on their eight dual-view examples leaves scaffold-free selection success unchanged at 3 of 355 tasks, with one gain and one regression. The result separates useful intervention from acquired capability: environment verification can establish what works at a state without establishing that a small policy has learned to reproduce it. RePair makes that distinction observable and identifies incomplete action coverage and persistent execution errors as concrete targets for further study.

1 INTRODUCTION

A failed attempt tells a language agent that something went wrong. It does not tell the agent what to learn. In a long task, failure may follow an unproductive search, a missed change of subgoal, or an earlier error from which the agent never recovered. A research model can inspect the trajectory and suggest a correction. Reflection-based self-training uses such feedback to improve generated samples (Dou et al., 2024). Yet a convincing suggestion is still a hypothesis about the policy's future behavior. Opening a cabinet instead of continuing to search might reveal the needed object, or it might spend the remaining budget and leave the task unsolved. The useful question is therefore concrete: what happens when *this policy* takes the proposed change from *that state*?

This question sits at a critical handoff in agent self-improvement. Memory and harness adaptation can preserve experience outside the model (Shinn et al., 2023; Yu et al., 2026); policy training can carry useful experience into the model parameters themselves (Chen et al., 2026; Yang et al., 2026a). The object we ultimately want to retain is a policy that performs better when these external aids are removed. A stronger intervention can improve the assisted trajectory while leaving that object unchanged. Aspire shows why completing the update loop is insufficient: agents can execute training while hidden-evaluation gains remain sparse or unstable (Wu et al., 2026b). We study the smaller, more testable step between these two ends—how evidence from a failed attempt becomes supervision that still helps after the research process is gone.

Prior work already supplies important pieces. CSO tests alternative actions through policy continuation and learns from verified preferences (Li et al., 2026). SIRI validates skill-guided rollouts and distills useful actions into an unassisted policy (He et al., 2026). OPID and SEED turn trajectory hindsight into skills whose effect on action probabilities guides policy learning (Yang et al., 2026b; Wu et al., 2026a). These advances make two questions especially salient. First, which failure-derived changes actually deserve to enter training? Second, once a change has earned that evidence, what information should the learner receive from it?

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.

The first question requires a baseline for the counterfactual. Restarting a failed attempt gives the policy another opportunity to succeed even without the proposed change. RePair therefore restores the source decision state and runs two matched continuations: the registered baseline action and the candidate intervention, with the same frozen policy available for subsequent decisions. A bounded intervention can itself complete the task; its success then measures the intervention's utility, not autonomous policy behavior. The terminal outcomes measure whether adding the intervention changes completion from that point. This same-state comparison turns a plausible correction into experimentally grounded evidence.

The second question begins after verification. An action target teaches the learner what to do at the source state. A compact strategy target can also teach the active subgoal, the progress signal to watch for, and the condition for changing course. In the cabinet example, opening the door is one action; recognizing what observation would complete the search, and how to recover if the object is absent, describes the decision structure around that action. The two targets define a controlled comparison on the same verified states and corrected actions. Here we instantiate the dual-view update and evaluate its descendant through the original action-only interface, with external failure memory and repair intervention disabled. An action-only training control remains necessary to isolate the auxiliary view's contribution.

We introduce RePair to make this handoff measurable end to end (Figure 1). Failure analysis and cross-round experience help propose testable changes; same-state continuations determine which ones become positive evidence; action and strategy targets specify what enters learning; scaffold-free evaluation measures what remains in the policy. The surrounding Analyzer, Planner, and memory machinery supports this sequence, while the empirical study tracks which of these handoffs actually succeeds.

Our ALFWorld study starts from Qwen2.5-3B-Instruct. Across five development rounds, most verified interventions leave both branches unsuccessful. Trace analysis locates a recurring failure: a local correction executes, but the continuation policy misses the next prerequisite or repeats an already completed action. A later validation makes more of the proposed strategy executable and obtains stable terminal benefits. Yet training on four such sources does not improve full-task selection success. These observations locate the unresolved handoff: a program can finish a task while the training examples teach only its initial decision. We report the trace and supervision evidence behind this gap and measure its consequence for the policy after external assistance is removed.

2 REPAIR

RePair treats a repair as an experiment before treating it as supervision. We follow that repair from its source state through testing, training, and evaluation. Figure 1 shows the surrounding research loop; Appendix D describes its analysis and memory mechanisms.

2.1 STARTING FROM A FAILED ATTEMPT

Let π_k be the task policy entering round k. Its input contains a goal, current observation, recent executed actions and feedback, and an admissible action menu. We collect trajectories on the training split and retain their actions, observations, budgets, and terminal outcomes. Completed task failures supply the repair candidates; infrastructure incidents have their own execution records.

Each candidate is tied to a recoverable decision state x and the context $c(x)$ visible to the policy. The source record fixes the checkpoint, baseline action, action menu, and remaining budget. Environment-restoration information stays outside $c(x)$. This distinction lets us restore the experiment while preserving the information available to the agent.

A useful source state precedes a decision the proposal can change. For example, continuing to search after the target has been found and searching before it has been found can require different corrections, even if both trajectories eventually fail. The source context anchors the proposal to which subgoal is still active and what the agent has already observed. This makes the unit of investigation a decision within a task.

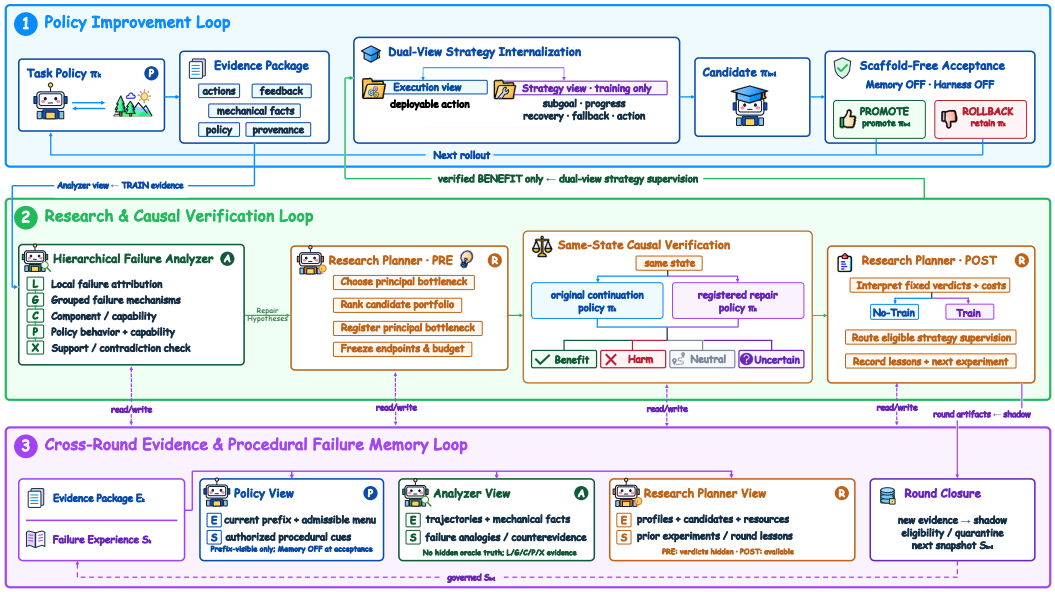

Figure 1: **RePair in context.** The paper follows the scientific path from failure-derived hypotheses through same-state verification, policy supervision, and scaffold-free acceptance. The research and memory loops support proposal selection and carry completed evidence across rounds; Section 2.5 defines action-only and dual-view targets from verified evidence.

2.2 Proposing a Repair That Can Be Tested

The research models inspect a failed trajectory, connect local errors with recurring failure patterns, and consult earlier repairs and counterexamples. A planner chooses a principal bottleneck and a small portfolio of tests. Each proposal specifies a source state, an executable repair r, and a compact strategy describing its subgoal, expected progress, and recovery conditions. The intervention is an action or a bounded program with conditions grounded in visible observations and explicit termination rules. Earlier rounds use one action or an option of at most four actions; the later validation uses ordered guarded phases within the same remaining environment budget. Its payload, strategy description, endpoint, and budget are fixed before the environment test.

The researcher receives trajectory observations and mechanical execution facts, including admissible actions and feedback. It can reason about search, subgoal transitions, prerequisites, or recovery, but the proposal must identify how to test that reasoning in the environment. Hidden restoration state remains with the verifier. This division lets the research model suggest a high-level strategy while the environment determines whether its executable realization helps.

This makes the proposal concrete. A suggestion to "search more carefully" becomes a particular action or option at a particular state. Related failures help the researcher distinguish an isolated invalid action from a recurring problem, such as continuing an exhausted search. Historical experience supplies possible next steps and the conditions under which they previously helped. These sources guide where to spend the test budget; current paired rollouts measure the effect on π_k.

Research within a round reads a fixed history snapshot. The proposal can use a lesson from an earlier round, but its action and strategy are settled before the current outcomes are revealed. Completed outcomes then become historical evidence for the next round (Appendix A.1). This allows research to accumulate experience while keeping each proposed intervention tied to a specific test.

2.3 Returning to the Same State

For a candidate (x, r), we restore x twice. Branch F_0 executes the registered baseline action a_0; branch F_1 executes the repair. If the task remains unfinished, the same frozen policy π_k supplies

subsequent decisions:

$$F_0 : x \xrightarrow{a_0} x'_0 \xrightarrow{\pi_k} Y^{(0)},$$
$$F_1 : x \xrightarrow{r} x'_1 \xrightarrow{\pi_k} Y^{(1)}. \tag{1}$$

Here $Y^{(b)} \in \{0, 1\}$ is terminal task success. We repeat the paired comparison five times. Each pair shares the starting state and visible context, checkpoint, base prompt interface, memory snapshot, remaining budget, and paired seed protocol. In the later strategy validation, r also includes a registered strategy cue on F1 continuation calls; the treatment is the program and cue together, not an isolated action change. The baseline is a fresh policy continuation and can succeed even though the source episode failed.

After the intervention the branches are free to diverge; that divergence is precisely what the repair is meant to cause. Every intervention action consumes the remaining interaction budget. For an option, the test covers the complete registered option, with admissibility checked at each reached state. Across repetitions, the joint F_0/F_1 outcomes describe the local, policy-conditioned effect of the proposed change. Appendix A.2 gives the restoration and execution checks.

Keeping the continuation policy fixed is central to this test. Replacing it with a stronger teacher asks whether the teacher can finish after the intervention. When continuation is needed, using π_k tests whether the learner can use the state reached by the intervention. If the program already reaches success, no policy continuation is required; we explicitly count this case. In the cabinet example, exposing an object is useful only to the extent that the remaining policy can turn that opportunity into task completion within the budget. Intermediate progress helps explain the branches; terminal success supplies the effect label.

2.4 Selecting Repairs for Learning

An independent verifier reads the paired outcomes. A candidate receives *Benefit* when at least four of the five valid pairs change failure to success and none changes success to failure. The symmetric rule gives *Harm*. Stable success in both branches and stable failure in both branches give two neutral outcomes; other complete results are uncertain. We use these repetitions to assess stability at the selected state; Appendix A.3 gives the exact counts and treatment of incomplete tests.

Let $\mathcal{I}_k$ contain completed interventions whose source and training records agree with the executed test. Positive supervision is the Benefit subset:

$$\mathcal{D}_k^+ = \{i \in \mathcal{I}_k : L_i = \mathsf{B}\}. \tag{2}$$

A change that repeatedly helps the current policy thus becomes a candidate lesson. The verifier selects on its executed effect; the associated strategy remains the description fixed before testing.

The remaining outcomes also inform research. Repeated success in both branches suggests that this correction is unnecessary under the tested conditions. Repeated failure leaves an unresolved difficulty, and a harmful intervention supplies a counterexample to the proposal. The planner uses these outcomes and their costs to choose TRAIN or NO-TRAIN and to guide later tests. This gives negative experiments a role without putting them in the positive-supervision set.

2.5 Learning an Action or Learning Its Strategy

A verified intervention tells us which decision helped. We now ask how to teach it. A historical data-path audit made this question concrete: an action-only materialization retained the chosen actions as targets while richer strategy information remained upstream (Appendix C.2). We therefore make that information an explicit auxiliary prediction target. The action-only formulation below defines the corresponding control; the completed update in this study uses dual views.

Action-only supervision uses the deployment context c_i and the target a_i, serialized as `{"action":"..."}`. Let c_i^s denote the same legal source context with a fixed training-only instruction. *Dual-view* supervision keeps the action examples and adds a strategy view:

$$\mathcal{T}_{\text{action}} = \{(c_i, a_i) : i \in \mathcal{D}_k^+\},$$
$$\mathcal{T}_{\text{dual}} = \mathcal{T}_{\text{action}} \cup \{(c_i^s, s_i) : i \in \mathcal{D}_k^+\}. \tag{3}$$

The strategy instruction requests a compact response s_i with the bottleneck, subgoal, expected event and state change, progress criterion, recovery trigger, fallback condition, and action. These fields relate a choice to the task: the subgoal says what it is meant to accomplish, the progress criterion says what to check next, and recovery conditions say when to change course. Table 1 illustrates the information added to an action target.

Table 1: **What each training view teaches.** Schematic example: the agent is looking for an object beside a closed cabinet, and `open cabinet 1` is admissible. The auxiliary view adds prospective task information to the same action. This illustrates target content.

View	Supervised content
Execution	`{"action":"open cabinet 1"}`
Strategy	Subgoal: inspect the cabinet. Expected progress: its contents become visible. Check: whether the target is present. Recovery/fallback: if it is absent, continue the search elsewhere. Action: `open cabinet 1`.

The response uses the strategy registered before verification. Expected progress is prospective: it states what to look for in subsequent feedback. Realized future outcomes, rewards, and effect labels are excluded from prediction targets. In the completed update, each source contributes its initial action and one strategy response. The later actions of a successful intervention are not expanded into additional execution-view examples. This distinction between verified program coverage and supervised action coverage matters for interpreting transfer.

The two views update the same policy weights through supervised assistant-token loss. Prompt tokens are masked; the auxiliary response makes the strategy itself a prediction target. The native-label check confirms that both action and substantive strategy tokens contribute to that loss. The fixed auxiliary instruction gives this training task its own output format while the original parser continues to accept exactly one action field. Evaluation uses the execution view directly, with no strategy-generation step or additional research-model call.

A controlled representation comparison would start both conditions from the same parent and evidence, and include an equal-compute action-only control. Those controls are not completed here. We report the executed dual-view labels, tokens, and optimizer steps in Appendix A.4; the observed parent–candidate change therefore measures this update as a whole.

2.6 Removing the Assistance

The final test belongs to the policy. We remove external failure memory and repair takeover and compare candidate π' with parent π under the original action interface. For task t and seed j, let $Y_{tj}(\pi)$ denote terminal success in this scaffold-free setting. We measure

$$\widehat{\mathrm{SR}}_{\mathcal{D}}(\pi) = \frac{1}{|\mathcal{D}|} \sum_{t \in \mathcal{D}} \frac{1}{J_{\mathrm{eval}}} \sum_{j=1}^{J_{\mathrm{eval}}} Y_{tj}(\pi),$$
$$\Delta^{\mathcal{D}}_{\mathrm{policy}}(\pi', \pi) = \widehat{\mathrm{SR}}_{\mathcal{D}}(\pi') - \widehat{\mathrm{SR}}_{\mathcal{D}}(\pi). \tag{4}$$

The goal, observations, ordinary within-task history, admissible menu, and strict action parser remain available to both policies. Evaluation starts at each task's initial state, so the learned model also has to reach useful decisions without being placed at a selected failure point. This extends the local intervention test to complete task behavior.

During the improvement loop, an independent predeclared acceptance rule compares candidate and parent on training-side selection tasks, after integrity checks. In the final validation, success count is primary; ties use the mean terminal fraction of satisfied goal conditions, evaluated outside the policy input. No clear improvement retains the parent. A NO-TRAIN decision retains the parent. The retained policy collects fresh failures for the next round, and completed research updates the next history snapshot. The final seen and unseen benchmark sets are evaluated after selection; their scores stay outside this adaptive loop. Appendix D specifies the full return from one round to the next.

3 EXPERIMENTAL STUDY

We examine the path from a tested correction to an unassisted policy in a development campaign and a subsequent one-update validation. The campaign and validation are sequential diagnostics with implementation changes, not randomized comparisons of repair representations.

3.1 SETTING AND COMPARISONS

Tasks and policy. We use text-based ALFWorld (Shridhar et al., 2021) and the Qwen2.5-3B-Instruct lineage (Qwen Team, 2025), updated with LoRA (Hu et al., 2021). The 3,553 training games are partitioned into 2,843 TRAIN_UPDATE, 355 TRAIN_SELECT, and 355 TRAIN_AUDIT games. Only the first pool supplies repair and learning evidence. Selection uses the second pool. The final benchmark has 134 unseen and 140 seen games, kept outside adaptive research and checkpoint selection.

Execution and statistical units. The policy receives the goal, current observation, eight executed transitions, interface feedback, and the complete admissible menu. The strict interface accepts one JSON action. Episodes allow 30 environment steps and 60 policy attempts, with termination after three consecutive nonexecuted attempts. Repair verification uses five paired seeds (17, 31, 47, 73, 101); the final selection comparison uses one paired seed (17) on all 355 tasks. Repetitions at one source are not independent tasks. We report exact counts and descriptive differences for one trained candidate, without claims of significance or training-seed robustness. Appendix B.1 records the executed configuration.

Scope of the evidence. We report completed intervention tests, native training labels, and paired selection outcomes. Matched action-only, equal-compute, filter, hierarchy, and memory ablations have not been completed for this update. The released checkpoint benchmark supplies external context; it does not isolate RePair's components. The RePair candidate has no completed final seen/unseen benchmark result.

3.2 WHICH REPAIRS HELP THE CURRENT POLICY?

Table 2: **Completed same-state tests.** Counts are source-state experiments; each has five pairs and ten branches. N denotes stable neutral failure. Validations reuse R5 evidence and change the selected portfolio and executable representation. They are not matched ablations.

Run	States	Benefit	Harm	N	Uncertain
R1	11	0	0	11	0
R2	6	1	0	5	0
R3	2	0	0	2	0
R4	1	0	0	1	0
R5	1	0	0	1	0
Guarded-strategy validation	17	0	0	16	1
Executable-strategy validation	5	4	0	1	0

Table 2 separates the five development rounds from two later validations that reuse R5 trajectories and Analyzer outputs. The development rounds yield one stable Benefit among 21 tested states; the other 20 are neutral failures. The first guarded-strategy validation has no stable Benefit among 17 states, although two of its 85 pairs show an F1-only success at the same source. The later executable-strategy validation yields four stable Benefits among five selected states. All five repetitions succeed in F1 and fail in F0 at each beneficial state. The higher yield motivates examining what the successful programs do; the sequentially selected portfolios leave the effect of each design change unresolved.

The trajectories explain why many valid corrections remain neutral. In R3, reaching the fridge enables cooling an egg, but the continuation then cools it 25 times instead of placing it. In a CD task, navigation reveals the target, but the policy attempts placement three times without first acquiring it. In R4, using a lamp is followed by 28 examinations of the same side table. The intervention executes; the remaining dependency chain fails. Appendix C.1 records further cases.

The four beneficial programs in the later validation execute two, four, six, and four actions, respectively, covering cleaning, placement, heating-and-placement, and object-in-light tasks. All 20 successful F1 branches finish within the registered program, with zero subsequent policy calls. The successful behavior therefore comes from the programs at these selected states. Whether their verified actions become useful lessons for the policy is the next question.

The heated-cup case makes the distinction concrete. From the restored state, the program acquires the cup, reaches the microwave, heats it, opens the microwave, reaches the cabinet, and places the cup. Completion follows this six-action sequence without another policy decision. In the remaining neutral case, the program locates and acquires a credit card, but the three subsequent policy calls fail to execute and placement remains unfinished. A useful intermediate state becomes a completed task only when the remaining prerequisites are also resolved. These cases locate the handoff that a trained policy would need to learn.

3.3 What Enters Training?

The four Benefit states contribute four execution views and four strategy views. The native-label audit counts 51 action-view and 2,546 strategy-view loss-bearing tokens per pass. Four passes produce eight optimizer updates and 10,388 supervised token exposures. Prompt and padding tokens are masked. The current adapter teaches the initial action and associated strategy at each source, leaving the remaining successful actions without stepwise execution targets.

Table 3: **Verified program coverage and supervised action coverage.** Each row is one Benefit source, with five successful F1 repetitions and zero policy continuation calls. Action counts are per repetition; native training counts are per source, not multiplied by the five seeds.

Repair task	Program actions	Action targets	Strategy targets
Clean pan	2	1	1
Place potato	4	1	1
Heat and place cup	6	1	1
Examine newspaper with lamp	4	1	1
Total	16	4	4

For example, the heated-cup training pair supplies an action target for acquiring the cup and a strategy description of the intended plan. The five later action positions are exercised by the verifier but are not separate action-prediction examples. The auxiliary view may communicate the plan, while execution from the intervening observations remains a generalization demand. Table 3 makes this coverage gap visible without treating each repetition as a new training source.

Thus verification coverage and training coverage differ: a program can complete several prerequisites while the action target teaches only its first decision. Strategy text supplies most supervised tokens; its gradient influence also depends on token losses and accumulation-group composition (Appendix A.4). The comparison below tests this eight-example update as a whole. Isolating the strategy objective would require an action-only update from the same four sources.

3.4 What Remains After the Research Harness Is Removed?

On TRAIN_SELECT, both parent and candidate solve 3 of 355 tasks (0.85%). Two successes are shared; one task improves and one regresses. Mean terminal goal-condition completion decreases from 10.845% to 10.704%, so the registered success-first, progress-on-ties rule retains the parent. The comparison comprises 710 completed episodes with memory and intervention disabled. It is a selection result, not a final benchmark.

Looking at task identities prevents the equal totals from hiding redistribution: 351 tasks fail under both policies, two succeed under both, and only two switch outcome. The secondary goal measure improves on five tasks, worsens on six, and is unchanged on 344. We therefore find neither a net completion gain nor a compensating aggregate progress gain under this acceptance rule. The two discordant success outcomes support an exact descriptive comparison, rather than a claim about training-seed robustness.

Table 4: **Scaffold-free selection after the dual-view update.** TRAIN_SELECT only: 355 paired tasks, seed 17, one trained candidate. Counts concern episodes except where marked as calls. Goal completion is the mean terminal fraction of satisfied environment goal conditions.

Measure	Parent	Candidate
Task successes	3 / 355	3 / 355
Mean goal-condition completion (%)	10.845	10.704
Consecutive-nonexecution terminations	261	252
Environment-budget terminations	91	100
Out-of-menu actions (calls)	1,556	1,612
Format failures (calls)	135	119

Most failures concern execution rather than unavailable observations. The parent and candidate terminate 261 and 252 tasks, respectively, after consecutive nonexecuted attempts. Their traces contain 1,556 and 1,612 out-of-menu actions despite visible admissible menus. Fewer early stops coexist with more budget-exhausted episodes (91 versus 100), leaving terminal success unchanged. In a task requiring an alarm clock to be examined with a desk lamp, both policies emit an unavailable examine command even though the observation contains the clock and the menu offers a take action. The missing step is therefore visible in the interface; the policy fails to use it. This example links the aggregate execution errors to a prerequisite failure of the same kind observed after local repairs.

The existing reference-checkpoint benchmark is retained as task context in Appendix B.2; all policy-update conclusions here use the paired TRAIN_SELECT comparison.

3.5 Research Cost and the Cross-Round Record

The five development rounds consume 210 completed verification branches over 21 tested states. The guarded and executable-strategy validations consume 170 and 50 branches, respectively. They reuse R5 collection and analysis, so they are neither fresh formal rounds nor independent samples of discovery yield. The last portfolio costs 50 branches for four Benefit states, or 12.5 branches per Benefit, excluding upstream research and earlier unsuccessful portfolios.

The campaign stops after five valid rounds under its three-round no-promotion patience rule. One infrastructure-invalid attempt, repeated engineering recoveries, and one user-authorized promotion exception remain part of the record. The later validation completes training and selection but encounters an ownership check during history closeout. We therefore report its verified selection decision separately from campaign closure, and make no claim of uninterrupted autonomous execution. Full API cost, matched discovery efficiency, and further policy improvement remain unmeasured in this study.

4 Related Work

From failure to a tested correction. Reflexion uses linguistic feedback in later attempts (Shinn et al., 2023), and Re-ReST uses reflection and external feedback to improve self-training data (Dou et al., 2024). AgentRx and TrajDebug localize failures from execution evidence (Barke et al., 2026; Qi et al., 2026); SAGE considers multiple failure hypotheses for research recovery (Ma et al., 2026). CSO goes from diagnosis to intervention, testing alternative actions through policy continuation and learning verified preferences (Li et al., 2026). RePair also tests corrections in the environment, then traces which parts of the verified intervention enter training and whether the resulting policy improves. This follows the correction beyond its local effect to the behavior retained after training.

From guidance to policy learning. GiGPO addresses credit assignment in agent RL (Feng et al., 2025), while Agent-G^2 adapts the expert-prefix guidance used to start training rollouts (Wang et al., 2026). SkillRL develops a hierarchical skill library (Xia et al., 2026); SIRI tests skills and distills useful skill-guided actions into an unassisted policy (He et al., 2026). OPID and SEED use the probability shifts induced by hindsight skills as a dense learning signal (Yang et al., 2026b; Wu et al., 2026a). Rationale supervision can also complement ordinary task targets (Hsieh et al., 2023). Our

formulation focuses on state-bound failure repairs: the same tested actions supply either ordinary action targets or those targets plus a training-only strategy view. The distinction is in the supervision given to a shared learner while its deployment interface stays fixed.

What carries across improvement rounds. Self-Harness and Recuris retain changes in external execution and memory mechanisms (Zhang et al., 2026; Yu et al., 2026). EvoTrainer couples training-harness adaptation with policy learning (Chen et al., 2026), and Frontis-MA1 learns program-evolution operators from executable feedback (Yang et al., 2026a). HELIX organizes matched harness outcomes into verified model-update records (Fan & Huang, 2026). Aspire and PAST-Bench examine whether updates and retained experience produce supported gains (Wu et al., 2026b; Xue et al., 2026). RePair studies the passage into the task policy: proposals are tested, their supervision is audited, and the updated model is evaluated after the research assistance is removed. Its research history can guide later experiments, while OFF/OFF evaluation measures what each update leaves in the learner.

5 Discussion and Limitations

The study locates two distinct obstacles. Early local corrections often leave the policy facing an unresolved prerequisite or phase transition. Making a larger part of the strategy executable can complete a task, but that success may belong entirely to the intervention program. The later dual-view update does not transfer this observed utility into higher scaffold-free selection success.

The learner receives initial-action targets from just four source states, whereas evaluation requires it to reach and complete these decisions from task reset. This mismatch in both coverage and starting conditions offers a plausible explanation for weak transfer. A controlled training comparison is needed to establish its causal role. The trajectories also reveal persistent menu and prerequisite errors. A useful next test would match source states and training budget while varying coverage of the verified continuation, before scaling the number of rounds. In particular, the current result suggests measuring what the learner does after each verified intermediate state, as well as from task reset. This would distinguish learning the successful continuation from learning to reach its starting point. Such a comparison follows directly from the observed gap; its outcome remains an empirical question.

The revisions and portfolios were chosen sequentially, so their outcome counts are descriptive. The 355-task comparison uses one seed and one candidate; it measures neither training variability nor final benchmark generalization. Matched action-only, filtering, equal-compute, and research-memory comparisons, together with a final RePair benchmark evaluation, remain open tests. The heterogeneous released baselines provide task context. Our present evidence concerns the local-to-policy transfer gap in this development study.

Five paired repetitions establish the registered stability criterion, not statistical certainty, and failure-selected states provide limited opportunities to measure harm. Human engineering recovery and a disclosed promotion exception prevent interpreting the development campaign as fully unattended. RePair's contribution here is the explicit separation and measurement of these handoffs, including the observed absence of a retained policy gain.

6 Conclusion

RePair connects failure analysis, same-state verification, dual-view supervision, and scaffold-free policy evaluation. Its ALFWorld study finds that executable corrections can produce stable terminal benefits while a small update from those corrections leaves independent task success unchanged. The trace and supervision audits show where the handoff remains incomplete: the intervention can perform a sequence that the action training data does not teach in full. This establishes a concrete experimental target for improving internalization, while keeping useful research assistance distinct from capability retained by the policy.

AI USE STATEMENT

Generative AI was used in the research method and in its development. External GPT API models supplied failure analyses, repair proposals, PRE/POST planning, and strategy supervision; the recorded provider model for the recent validation is `gpt-5.6-sol`. Qwen2.5-3B-Instruct and its trained descendants generated task actions. ChatGPT and Codex assisted with conceptual and experimental design, implementation and debugging, data extraction and analysis, literature search, mathematical exposition, English revision, figures, and LaTeX production. Task outcomes and effect labels were computed from environment execution rather than accepted from model judgments. Checks included source/result identity checks, tests, native training-label audits, paired episode recounts, and document compilation and layout inspection. The authors remain responsible for the final claims, interpretation, and artifacts.

ETHICS STATEMENT

The study concerns simulated household tasks and provides no evidence for safe physical deployment. External research models process simulated trajectory evidence. The accompanying numerical supplement excludes credentials, private server locations, and identifying operational logs. Third-party environments and checkpoints retain their original licenses and access conditions. Regressions, unsuccessful tests, implementation changes, and human intervention are disclosed to avoid conflating system execution with autonomous capability improvement.

REPRODUCIBILITY STATEMENT

Section 2 defines the procedure; Appendix A specifies interventions and training; Appendix B gives the evaluation protocol and checkpoint context; and Appendix D documents the campaign. The accompanying anonymous numerical supplement contains the 355 paired selection records, 50 validation branch summaries, training configuration, and a script that recomputes the reported counts. It supports direct numerical verification of the tables. Independently rerunning the GPU/API campaign would additionally require model weights, environment replay assets, and API access, which are outside this numerical supplement.

REFERENCES

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. AgentRx: Diagnosing AI Agent Failures from Execution Trajectories. *arXiv preprint arXiv:2602.02475*, 2026. URL https://arxiv.org/abs/2602.02475.

Guhong Chen, Yingcheng Shi, Yongbin Li, Binhua Li, Xander Xu, Hu Wei, Shiwen Ni, Min Yang, and Jieping Ye. EvoTrainer: Co-Evolving LLM Policies and Training Harnesses for Autonomous Agentic Reinforcement Learning. *arXiv preprint arXiv:2606.03108*, 2026. URL https://arxiv.org/abs/2606.03108.

Zi-Yi Dou, Cheng-Fu Yang, Xueqing Wu, Kai-Wei Chang, and Nanyun Peng. Re-ReST: Reflection-Reinforced Self-Training for Language Agents. *arXiv preprint arXiv:2406.01495*, 2024. URL https://arxiv.org/abs/2406.01495.

Tianyu Fan and Chao Huang. HELIX: Model-Harness Co-evolution for Recursive Self-Improvement. *arXiv preprint arXiv:2608.13951*, 2026. URL https://arxiv.org/abs/2608.13951.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-Group Policy Optimization for LLM Agent Training. *arXiv preprint arXiv:2505.10978*, 2025. URL https://arxiv.org/abs/2505.10978.

Zhongyu He, Yuanfan Li, Fei Huang, Tianyu Chen, Siyuan Chen, Xingyang Li, Meng Hsuan Yu, Xiangrong Liu, Leyi Wei, Lu Pan, Ke Zeng, and Xunliang Cai. SIRI: Self-Internalizing Reinforcement Learning with Intrinsic Skills for LLM Agent Training. *arXiv preprint arXiv:2606.02355*, 2026. URL https://arxiv.org/abs/2606.02355.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.507. URL https://aclanthology.org/2023.findings-acl.507/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

Mukai Li, Qingcheng Zeng, Tianqing Fang, Zhenwen Liang, Linfeng Song, Qi Liu, Haitao Mi, and Dong Yu. Verified Critical Step Optimization for LLM Agents. *arXiv preprint arXiv:2602.03412*, 2026. URL https://arxiv.org/abs/2602.03412.

Jie Ma, Binfei Chu, Jie Gao, Jinlu Zhang, Yiwei Ma, Yi Tan, Jiayi Ji, Xiaoshuai Sun, and Rongrong Ji. One Reflection Is Not Enough: Self-Correcting Autonomous Research via Multi-Hypothesis Failure Attribution. *arXiv preprint arXiv:2606.31478*, 2026. URL https://arxiv.org/abs/2606.31478.

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. *arXiv preprint arXiv:2608.06346*, 2026. URL https://arxiv.org/abs/2608.06346.

Qwen Team. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2025. URL https://arxiv.org/abs/2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *arXiv preprint arXiv:2305.18290*, 2023. URL https://arxiv.org/abs/2305.18290.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Zixuan Wang, Yanrui Miao, Zhengxi Lu, Teng Pan, Yiwen Qiu, Hongxing Li, Peng Qiu, Ruiqing Zhang, and Yongliang Shen. Agent-G^2: Gaussian guidance for agentic reinforcement learning. *arXiv preprint arXiv:2608.23318*, 2026. URL https://arxiv.org/abs/2608.23318.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SEED: Self-Evolving On-Policy Distillation for Agentic Reinforcement Learning. *arXiv preprint arXiv:2607.14777*, 2026a. URL https://arxiv.org/abs/2607.14777.

Yuhao Wu, Jingyuan Zhang, Jiajun Shi, Yuxuan Zhang, Xinping Lei, Junting Zhou, Zexuan Wang, Yuchen Wu, Huan Zhou, Duo Wang, Yinzhu Piao, Yongchang Peng, Yunfeng Shi, Jin Chen, Zuo Wang, Jinkai Liu, Jiaheng Liu, Wenxuan Zhang, Shen Yan, Wenhao Huang, and Ge Zhang. Aspire: Can Models Self-Evolve from Vague Goals? *arXiv preprint arXiv:2608.31111*, 2026b. URL https://arxiv.org/abs/2608.31111.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. SkillRL: Evolving Agents via Recursive Skill-Augmented Reinforcement Learning. *arXiv preprint arXiv:2602.08234*, 2026. URL https://arxiv.org/abs/2602.08234.

Shuhan Xue, Zixin Ding, Yichen Shen, Yinjie Wang, Zhenfei Yin, Yingcheng Wu, Yuxin Chen, Mengdi Wang, and Ling Yang. PAST-Bench: Benchmarking the Foundations of Recursive Self-Improvement in Personal Agents. *arXiv preprint arXiv:2608.04003*, 2026. URL https://arxiv.org/abs/2608.04003.

Junlin Yang, Che Jiang, Yu Fu, Tianwei Luo, Can Ren, Weizhi Wang, Kaikai Zhao, Hongyi Liu, Yuxin Zuo, Yuru Wang, Yuchen Fan, Kai Tian, Zhenzhao Yuan, Xiaojian Lin, Li Sheng, Rushi Qiang, Guoli Jia, Xingtai Lv, Ermo Hua, Dianqiao Lei, Youbang Sun, Ning Ding, Bowen Zhou, and Kaiyan Zhang. Frontis-MA1: Training an AI4AI Model towards Recursive Self-Improvement in Machine Learning Engineering. *arXiv preprint arXiv:2607.28568*, 2026a. URL https://arxiv.org/abs/2607.28568.

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

Zhaochen Yu, Yingcheng Wu, Zhenfei Yin, Kaiyuan Chen, Zhe Zhao, Mengdi Wang, Shuicheng Yan, and Ling Yang. Recursive Experiential-Working Memory Evolution for Long-Horizon Agent Harnesses. *arXiv preprint arXiv:2608.24876*, 2026. URL https://arxiv.org/abs/2608.24876.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses That Improve Themselves. *arXiv preprint arXiv:2606.09498*, 2026. URL https://arxiv.org/abs/2606.09498.

APPENDIX

A REPAIR TESTS AND TRAINING DETAILS

This appendix specifies the state restoration, evidence handling, and learning procedure used in Section 2.

A.1 NOTATION AND INFORMATION AVAILABLE TO EACH ROLE

Table 5: Notation for a repair, an improvement round, and an evaluation task.

Symbol	Meaning
π_k	Retained task policy entering round k.
$\widetilde{\pi}_{k+1}$	Trained candidate awaiting the selection test.
$x,\ c(x)$	Recoverable environment state and policy-visible context.
$r,\ a_0$	Registered repair and current-policy baseline action.
$E_k,\ S_k$	Current trajectory evidence and round-start experience snapshot.
$\mathcal{U}$	Common predeclared set of failure units.
$\mathcal{I}_k$	Completed interventions whose source and training records match the executed test.
J_{eval}	Repetitions per evaluation task; one in the primary benchmark.
$\mathcal{D}_k^+$	Benefit subset used for positive supervision.
A_k	Scaffold-free acceptance result for candidate policy $k+1$.

Table 6: Research inputs and responsibilities. Current experiment outcomes enter POST directly; historical retrieval reads the round-start snapshot.

Role	Available evidence	Responsibility
Task policy	Goal, visible prefix, menu; permitted procedural cues when memory is enabled	Select the next action. OFF/OFF evaluation removes the extra cues.
Analyzer	Trajectories, mechanical facts, failure analogies, counterexamples	Propose source-grounded failure hypotheses and repairs.
Planner PRE	Failure profiles, candidates, resources, earlier experiments	Select and register the intervention portfolio before current verdicts.
Planner POST	Registered PRE, fixed current verdicts and costs, earlier lessons	Interpret experiments and recommend TRAIN or NO-TRAIN.
Verifier	Executed paired traces and the fixed effect rule	Assign Benefit, Harm, neutral, or uncertain labels.
Acceptance rule	Candidate and parent results on training-side selection tasks	Apply the predeclared promotion and regression criteria.

A historical record retains starting conditions, the action/feedback sequence, observed state changes, a wrong branch and its consequence, the repair, applicability conditions, counterexamples, and effect provenance. Retrieval identifies whether a record is an exact past observation or an analogy. Its usefulness for the current policy is determined by a new experiment. Within round k, historical reads use the immutable snapshot S_k. New records accumulate in a shadow writeback store and become available as history through S_{k+1} after round closure. POST receives current verdicts through its explicit experiment-result input. Hidden restoration state and future outcomes are excluded from task-policy inputs.

A.2 RESTORING THE SOURCE AND EXECUTING THE REPAIR

The source record identifies the task/gamefile, source prefix and state, policy checkpoint, prompt, initial menu, memory snapshot, candidate payload, and remaining budget. Restoration uses an environment-supported snapshot or a checked replay of the recorded prefix. The experiment records

the backend, replay checks, paired seed protocol, and restored-context equality; the current validation uses checked public-prefix replay. Its five continuation seed slots are 17, 31, 47, 73, and 101. All 50 planned branches of the executable-strategy validation are scientifically complete. The policy receives only $c(x)$, even when restoration requires additional simulator state.

A one-action repair replaces the registered baseline decision once. Earlier options execute one to four registered actions. In the later validation, ordered guarded rules select admissible actions using the current public observation, carried-object evidence, and menu. Each rule has a finite use count and the program obeys the remaining episode budget. The executor checks each action against the newly observed menu, debits actual calls and steps, and records completion or invalidity. If the task is unfinished after the program, π_k continues from the resulting state and remaining budget. The later validation appends the registered strategy to F1 continuation prompts. F0 retains the original continuation input. This cue is part of the treatment; it is not added to the scaffold-free selection prompts. The intervention remains fixed throughout the paired repetitions. This tests the whole option, including its extra actions and observations. The completed training run uses one initial-action example per verified source, not an expansion of every executed program step. A future action-only comparison must match this action coverage explicitly.

Missing restoration, unsupported options, parser violations, and incomplete branches have explicit execution outcomes. The discovery export retains attempted, supported, completed, and labeled counts over the registered universe. Only complete protocol-valid tests enter the effect-label rule. This preserves the distinction between an experiment that found no benefit and one that did not produce an interpretable comparison.

A.3 Paired Outcomes and the Stability Rule

For five valid paired repetitions, define

$$n_{ab} = \sum_{j=1}^{5} \mathbf{1}\{Y_j^{(0)} = a, Y_j^{(1)} = b\}, \qquad a, b \in \{0, 1\}. \tag{5}$$

The effect label is

$$L(x, r) = \begin{cases} \mathsf{B}\ \text{(Benefit)}, & n_{01} \geq 4,\ n_{10} = 0, \\ \mathsf{H}\ \text{(Harm)}, & n_{10} \geq 4,\ n_{01} = 0, \\ \mathsf{NS}\ \text{(neutral-success)}, & n_{11} \geq 4, \\ \mathsf{NF}\ \text{(neutral-failure)}, & n_{00} \geq 4, \\ \mathsf{U}\ \text{(Uncertain)}, & \text{otherwise.} \end{cases} \tag{6}$$

Insufficient valid pairs retain their incomplete-test status. The endpoint is terminal task success, fixed when PRE registers the experiment. Intermediate progress, loop reduction, and environment steps support the behavioral analysis without replacing that endpoint.

The four-of-five rule describes finite-repeat stability, not a confidence level. Its effect is conditional on the source state and continuation policy. Because the source episodes were failures, the sampled states may provide few successful baseline continuations that a repair could damage. Harm counts are consequently reported for this selected population. Verification establishes the utility of the executed intervention; claims about a particular explanatory field require their own interventions.

A.4 Training Views and Native Supervision

The execution view retains the exact I1 policy prompt and one string-valued `action` field. The auxiliary view uses the same policy-visible context followed by a fixed training-only instruction requesting a compact strategy response. The target describes the principal bottleneck, current subgoal, expected next event and state change, progress criterion, recovery trigger, fallback condition, and action. A strategy identifier and evidence references link this response to the tested repair. The strategy identifier is computed from the pre-verification payload. Verification provenance links that same payload to the effect label, which controls training eligibility. Expected progress is prospective information; the target excludes realized future outcomes, rewards, Benefit tags, promotion decisions, and sealed benchmark feedback. The native census contains four execution views and four auxiliary

views: 51 action and 2,546 strategy loss-bearing tokens per pass, with 4,096 prompt tokens masked across the eight examples. No prompt truncation, retokenization, or repeated chat templating is applied.

Both views share one real source state. A common source-state identifier links them to that state, while a distinct example identifier distinguishes each view. Native labels mask prompt and padding tokens with -100 and supervise the assistant suffix. The label census checks each verified pair for nonzero action supervision and nonzero substantive strategy supervision. It reports action, strategy, and metadata tokens separately, including the renderer's treatment of strategy identifiers and evidence references. Every admitted pair is Benefit-verified; the census rejects a purported strategy view that contains only an action target. These checks concern explicit supervision. Prompt content can still affect the conditional distribution learned from assistant targets.

For view $v \in \{e, s\}$, assistant-token mask m_{ivt}, and minibatch $\mathcal{B}$, the common objective is

$$\mathcal{L}_{\text{policy}}(\theta; \mathcal{B}) = -\frac{1}{Z_{\mathcal{B}}} \sum_{(i,v)\in\mathcal{B}} \sum_{t} w_{ivt} m_{ivt} \log \pi_\theta(y_{ivt} \mid p_i^v, y_{iv,<t}). \tag{7}$$

Action-only training samples execution views; dual-view training samples both views. In the executed run, $w_{ivt} = 1$ on supervised assistant tokens and $Z_{\mathcal{B}}$ is the number of shifted, unmasked targets across the gradient-accumulation group. Each microbatch has one example; gradients from four microbatches accumulate using that common token denominator. There is no separate equal weighting of the two views. Every one of the eight rows appears once per pass in the registered deterministic order; four passes produce eight optimizer updates on one GPU. Thus the longer strategy responses contribute many more supervised token terms, although token counts alone do not determine their gradient magnitudes.

The executed update continues the retained parent's trainable LoRA adapter with a fresh optimizer and scheduler. Its four Benefit sources are drawn from R5 evidence; the candidate is not initialized from a new untrained base model. An action-only or equal-compute descendant was not trained in this validation. Consequently, the result does not estimate the causal contribution of strategy supervision.

A.5 Additional Training Conditions

The historical design also identifies T5 for Benefit–Harm preference learning and T6 for audited RL. They are optional extensions beyond the primary supervised comparison. A Benefit pair prefers the repair to its baseline; a Harm pair reverses that preference. Neutral outcomes alone supply no such ordering. An executed T5 condition would specify its reference policy, preference serialization, and objective parameters, including β for DPO (Rafailov et al., 2023). T6 requires its own executed loss and rollout protocol. Neither extension contributes a result to this study.

A.6 Recording Autonomy and Recovery

Routine scientific decisions include choosing a research question and test portfolio, recommending training, and accepting a candidate under the fixed rules. Campaign records distinguish these decisions from initial configuration, file transfers, restarts, code edits, and automated retries. Each has its own count and timestamp; absence of a record remains unknown. The intended steady-state operation has zero routine human scientific decisions after initialization.

Valid attempts, no-promotion patience, abstention, nonproductive portfolios, integrity checks, and total resource limits determine stopping. Infrastructure-invalid attempts follow bounded recovery rules and retain their cost even when excluded from the scientific-attempt counter. The observed campaign stops after five valid rounds and three successive rounds without promotion. The invalid attempt remains separately counted; later reused-evidence validations do not increment the formal round count. Local Analyzer/Planner distillation is a separate research-role experiment, with shadow qualification preceding any takeover. The main method can use fixed strong research models throughout.

B Evaluation Protocol and Comparisons

B.1 Tasks, Interface, and Execution Settings

The training design partitions 3,553 gamefiles into 2,843 TRAIN_UPDATE, 355 TRAIN_SELECT, and 355 TRAIN_AUDIT tasks. The first pool supplies rollouts, analysis, repair experiments, and training; the second supplies candidate-selection results; the third supports research-role qualification. All states, seeds, and branches belonging to one gamefile inherit its pool. Final seen and unseen evaluations remain outside RePair's adaptive research and selection loop.

Table 7: Executed dual-view validation and evaluation settings. Repair verification and policy selection use different seed protocols.

Field	Setting
Base model	Qwen2.5-3B-Instruct; retained parent adapter continued, base weights frozen.
Research models	External GPT API; recent validation records `gpt-5.6-sol`; no teacher takeover during branch execution.
Training-side pools	UPDATE 2,843; SELECT 355; AUDIT 355, disjoint by gamefile.
Final benchmark	Unseen 134 and seen 140, kept separate from selection.
Interface and budget	Strict single-action JSON; full menu; eight executed transitions; 30 environment steps; 60 policy attempts; three consecutive nonexecutions stop.
Selection / repair seeds	Selection: 17; repair: 17, 31, 47, 73, 101.
LoRA	Rank 16; alpha 32; dropout 0.05; no bias; attention projections and gate/up/down MLP projections.
Optimizer	AdamW; learning rate 2×10^{-5}; betas $(0.9, 0.999)$; epsilon 10^{-8}; weight decay 0.01; gradient norm cap 1.0.
Schedule and batches	Linear schedule; one warmup step; microbatch 1; accumulation 4; four passes; eight optimizer steps.
Data and randomness	Eight examples; max length 1,248 without truncation; training and data seed 17; one trained run.
Loss and precision	Assistant-only token mean per accumulation group; BF16; one GPU; no packing or distributed reduction.
Checkpoint	Final update only; no intermediate evaluation or checkpoint selection.
Statistics	Exact task-paired counts; no significance or multi-training-seed estimate.

Shared raw prompt. The seven-line prompt begins with a fixed format identifier, followed by the task goal, current observation, executed-transition history, complete admissible-action menu, interface feedback, and output requirement. The context fields are JSON-serialized, and the menu retains its original order. The output requirement is `{"action":"<command>"}`. The parser accepts exactly one top-level JSON object with one string member named `action`. It removes outer transport whitespace and ASCII spaces at action edges, then tests exact case-sensitive menu membership. Duplicate members, trailing data, nonstandard constants, multiline or control characters, and actions longer than 256 codepoints fail parsing. No action-repair or canonicalization layer is applied.

Budget accounting. A format failure or parsed out-of-menu action consumes one policy attempt and zero environment steps. An admissible action consumes one of each; successful environment execution resets the consecutive-nonexecution count. After a nonexecution, the consecutive-three limit takes precedence over policy-attempt exhaustion. After execution, termination priority is infrastructure failure, environment terminal status, the 30-step limit, then the 60-attempt limit. Each episode retains its termination reason.

Frozen evaluation assets. The shared evaluation protocol and the two task lists are fixed before evaluation and kept unchanged across methods. The lists contain 140 seen games and 134 unseen games, respectively. Version records and file checksums identify these assets, and execution records link each run to the assets it used. Full task lists and run records are retained with the experiment artifacts; the accompanying numerical supplement preserves the table inputs and result provenance

without private server paths. The shared external interface and action budget coexist with model-specific inference profiles. In particular, high-reasoning API inference and local 128-token generation have different computation costs. The shared action budget is not a matched inference-compute budget; we do not claim equal token cost or latency across these methods.

B.2 Reference Checkpoints on the Final Benchmark Splits

The benchmark columns are Pick, Clean, Cool, Look, Heat, Pick2, and All. They correspond to pick-and-place, clean-and-place, cool-and-place, look-at-object-in-light, heat-and-place, and pick-two-object tasks. Table 8 reports unseen-134; Table 9 uses the same method roster and columns for seen-140. These reference measurements give task context across heterogeneous model scales and inference settings. The RePair candidate has not been evaluated on either final split; the tables therefore contain reference models only.

Table 8: **Unseen ALFWorld reference results.** Success (%) on 134 games using seed 17. All is task-weighted. These seven reference models use the shared task and environment protocol with model-specific inference settings. They provide task context, not a matched comparison with RePair; its final benchmark evaluation remains incomplete.

Method	Pick	Clean	Cool	Look	Heat	Pick2	All
GPT 5.6 sol high	83.3	80.6	90.5	83.3	78.3	70.6	81.3
Qwen2.5-3B-Instruct (base model)	0.0	0.0	0.0	0.0	4.3	0.0	0.7
SkillRL	37.5	41.9	61.9	44.4	43.5	11.8	41.0
GiGPO	87.5	87.1	95.2	88.9	95.7	94.1	91.0
SEED	91.7	83.9	71.4	11.1	95.7	35.3	69.4
Agent_G2	79.2	87.1	85.7	50.0	78.3	94.1	79.9
OPID	54.2	22.6	19.0	11.1	52.2	5.9	29.1

Table 9: **Seen ALFWorld reference results.** Success (%) on 140 games using seed 17. All is task-weighted. These seven reference models use the shared task and environment protocol with model-specific inference settings. They provide task context, not a matched comparison with RePair; its final benchmark evaluation remains incomplete.

Method	Pick	Clean	Cool	Look	Heat	Pick2	All
GPT 5.6 sol high	88.6	59.3	72.0	84.6	93.8	58.3	75.0
Qwen2.5-3B-Instruct (base model)	0.0	0.0	0.0	0.0	0.0	4.2	0.7
SkillRL	60.0	51.9	68.0	46.2	50.0	20.8	50.7
GiGPO	100.0	96.3	88.0	92.3	100.0	95.8	95.7
SEED	100.0	100.0	64.0	46.2	81.2	54.2	78.6
Agent_G2	97.1	96.3	84.0	76.9	62.5	91.7	87.9
OPID	42.9	51.9	20.0	15.4	43.8	29.2	35.7

For disjoint task-family subsets $\mathcal{D}_c$ within one split,

$$\widehat{\mathrm{SR}}_c = \frac{1}{N_c} \sum_{t \in \mathcal{D}_c} \bar{Y}_t, \qquad \widehat{\mathrm{SR}}_{\mathrm{All}} = \frac{\sum_c N_c \widehat{\mathrm{SR}}_c}{\sum_c N_c}, \qquad \bar{Y}_t = \frac{1}{J_{\mathrm{eval}}} \sum_j Y_{tj}, \quad N_c = |\mathcal{D}_c|. \tag{8}$$

Thus All weights families by their task counts. Tables report $100\widehat{\mathrm{SR}}$ as percentages. The primary benchmark uses $J_{\mathrm{eval}} = 1$. Raw successes, family counts, and their mapping to ALFWorld task IDs are exported with each table. Seen and unseen have separate denominators throughout.

B.3 Scope of Internal Comparisons

The framework supports simple versus hierarchical analysis, historical memory access, alternative Planner rankings, and action-only versus dual-view supervision. These define useful controlled

experiments, but the present development sequence changes multiple components and does not instantiate those matched comparisons. In particular, there is no completed weaker-filter training control or equal-compute action-only control for the reported candidate. We do not report a component effect from the difference between sequential portfolios.

B.4 Repair Yield, Cost, and Statistical Units

A registered-unit discovery-yield comparison would require a common failure universe, proposal budget, and verifier. The current report instead uses the directly observed number of tested states and branches in each portfolio. These denominators exclude unselected candidates and therefore must not be interpreted as a matched discovery-yield comparison. Cost per useful unit is

$$C_{\text{repair}}(a) = \frac{N_{\text{branches}}(a)}{N_{\text{Benefit units}}(a)}. \tag{9}$$

A zero-Benefit run reports its total cost and an undefined ratio. The reported branch counts include complete unsuccessful tests. They do not constitute total campaign cost: upstream API calls, retries, engineering recovery, and training compute have not been reconciled into a comparable end-to-end cost estimate.

Among candidates with complete formal verdicts, the Benefit share is

$$p_B = \frac{N_B}{N_{\text{tested}}}, \qquad N_{\text{tested}} = N_B + N_H + N_{NS} + N_{NF} + N_U. \tag{10}$$

This candidate-level denominator differs from a discovery yield measured over a common registered failure universe. The outcome breakdown retains neutral and uncertain cases alongside Harm, while unbound, unsupported, invalid, and incomplete proposals have separate counts.

The 355 paired selection tasks are the statistical units for the final validation. The paired outcome table has 2 shared successes, 1 candidate-only success, 1 parent-only success, and 351 shared failures. We provide exact counts rather than a significance claim based on these two discordant tasks. Five continuation seeds at a repair source are repetitions nested within that source, not five independent repairs.

B.5 The Verified-Action Control

A matched action-only control would share the parent, source evidence, action coverage, and evaluation protocol while omitting auxiliary targets. CSO uses verified alternatives with a different critical-step procedure and DPO objective (Li et al., 2026); such an internal RePair control would not constitute a CSO reproduction. This control has not been run here.

B.6 Released Checkpoints in the Reference Benchmark

The external comparison uses the five author-released checkpoints in Table 10. Their selection precedes shared benchmark evaluation. Each is loaded at the previously frozen repository revision; the experimental release records that revision, conversion steps, weight hashes, tokenizer, and episode outputs. The model names and nominal sizes identify a heterogeneous checkpoint comparison.

Table 10: Author-released ALFWorld checkpoints evaluated under the common RAW_WITH_MENU interface. Repository names identify the selected releases; immutable revisions, conversion receipts, and completion records are retained with the experimental artifacts.

Method	Scale	Checkpoint repository
SkillRL	7B	`Jianwen/Alfworld-7B-RL`
GiGPO	7B	`langfeng01/GiGPO-Qwen2.5-7B-Instruct-ALFWorld`
SEED	3B	`Jinyang23/Seed-AlfWorld-3B`
Agent_G2	7B	`xiamoent/Agent-G2-alfworld-7b`
OPID	1.7B	`Jinyang23/OPID-ALFWorld-1.7B`

All five use the shared ALFWorld task/environment contract and primary seed slot, while method-faithful adapters preserve inference behavior required by the released checkpoint. Project RePair

Memory and Harness are disabled. A frozen native component is retained when it is part of the released method's inference definition—for example, SkillRL uses its frozen official SkillBank without benchmark-time writeback or dynamic updating. The evaluation performs no checkpoint adaptation from benchmark feedback. Model scales, checkpoint identities, decode profiles, and adapter choices are recorded with the run. The reported parent–candidate selection comparison uses one lineage, but it does not isolate individual algorithmic components.

SkillRL's sharded actor checkpoint requires its recorded conversion to a loadable model; no optimizer state is used for benchmark-time learning. GiGPO, SEED, Agent_G2, and OPID likewise retain the chosen release identity through loading and evaluation. The final export accounts for all 274 primary cells per checkpoint, with 134 unseen and 140 seen tasks, and records raw responses, parser outcomes, actions, steps, termination, tokens, and latency. Aggregate scores are derived after completion. The completed task-level exports for all five released baselines are the source of Tables 8 and 9.

C Behavioral Analysis and Additional Experiments

C.1 Task Outcomes and Trace-Linked Cases

The audit covers all 710 selected-seed episodes, with action-trace checksums matched to the recorded attempts, and all 50 branches in the executable-strategy validation. The public task goal, current observation, admissible menu, recent executed history, and interface feedback are present in policy prompts. The policy is not being asked to act without environment feedback. Full private restoration state is intentionally excluded, and the research models receive public trajectory evidence rather than an unrestricted live simulator.

Table 11: Observed continuation failures and a beneficial program. These cases illustrate mechanisms; they are not estimates of their frequency across all ALFWorld tasks.

Case	Executed intervention	Observed continuation
R3, cool egg	Navigate to fridge while carrying egg	Cool the egg 25 times; no placement; budget exhausted.
R3, place CD	Navigate to desk; CD becomes visible	Attempt placement three times without acquisition; commands absent from menu.
R4, examine CD	Use desk lamp	Examine the side table 28 times; target not acquired.
R5, place remote	Put a newspaper on sofa	The requested object is a remote control; the tested action does not complete its prerequisite chain.
Validation, heat cup	Acquire cup, reach microwave, heat, open, reach cabinet, place	Program completes the task in six actions; no policy continuation call.

In the selection task requiring an alarm clock to be examined with a desk lamp, the public observation contains an alarm clock and the menu offers a take action. Both policies nevertheless emit an out-of-menu examine command. Across all tasks, out-of-menu calls total 1,556 for the parent and 1,612 for the candidate; format failures total 135 and 119. Most format failures are multiline action strings (130 and 113); the remainder are invalid JSON envelopes (5 and 6).

The candidate executes slightly more environment actions on average (13.479 versus 13.228) and makes more policy calls (18.355 versus 17.992). Adjacent repeated executed actions total 1,902 versus 1,767. Repetition is a trace statistic, not automatically proof that an action is useless. Together with the unchanged success count and additional budget-exhausted episodes, these results do not show overall behavioral improvement.

C.2 What Was Verified and What Was Supervised

The four successful programs contain 2, 4, 6, and 4 executed actions per repetition. All 20 F1 successes occur before any policy continuation call. The remaining selected state acquires a credit card but does not complete placement; it remains a neutral failure. The training materialization

contains four initial-action targets and four strategy targets. It does not supervise all 16 action positions of the four successful suffixes.

An earlier native-label audit recorded 12 action-only rows and 151 loss-bearing tokens, with richer strategy descriptions remaining upstream. The present native-label census establishes that substantive auxiliary targets now enter learning. Neither audit alone proves that a particular representation causes better policy behavior. The numerical supplement reports the current census and executed training configuration.

C.3 Process Effects and Capability

Intermediate changes remain separate from terminal effect labels. In the earlier guarded validation, 85 pairs include 11 positive process effects, 1 negative, 57 unchanged, and 16 unassessable. Ten positive process cases are terminal-neutral. These descriptive process records do not become Benefit labels and are not admitted to the current Benefit-only training set. A future progress-supervision condition would require its own admission rule and comparison.

D Research Procedure and Cross-Round Evidence

The main text follows one repair into the policy. The full procedure in Figure 1 chooses those repairs, allocates tests, and carries completed evidence into later rounds.

D.1 Hierarchical Failure Analysis

The evidence package E_k contains policy actions, observations, feedback, mechanically computed progress events, budgets, termination, and checkpoint provenance. Local analysis L identifies error lifecycles and recovery opportunities. Deterministic grouping supplies related trajectories to G for recurring-mechanism analysis. Stage C aggregates component and capability hypotheses. Stage P assembles the global policy-behavior profile deterministically, and X checks its support, contradictions, and unresolved alternatives. Here P denotes the profile stage; the task policy and the planner's bottleneck choice have separate roles.

A0 uses simple local analysis, A1 structured local hypotheses, A2 the cross-trajectory hierarchy, and A3 the hierarchy plus historical experience. A2–A1 and A3–A2 define hierarchy and history comparisons, respectively. No matched effect estimate for these contrasts is reported in this study. The reused R5 analysis includes 60 local slots, 44 completed G pairs, 88 C terminals, and 87 X terminals. These are analysis records, not independent successful repair experiments.

D.2 Procedural Experience across Rounds

The round-start snapshot S_k stores starting conditions, actions and feedback, state changes, a wrong branch and its consequence, repair applicability, counterexamples, and effect provenance. Policy, Analyzer, and Planner views expose the parts appropriate to each role. A past repair can enter a new proposal as an analogy or counterexample; current paired execution determines the new effect.

Writes produced within the round enter a shadow store. Historical reads throughout round k use the fixed snapshot S_k. Only after round closure do admitted records become part of S_{k+1}, which is first readable in round $k+1$. R3 PRE receives closed R1/R2 lessons, and R4 PRE receives R1–R3 lessons; repeated local proposals therefore cannot be explained solely by absent history access. Matched history and ranking comparisons remain unexecuted. The read/write arrows in Figure 1 depict these mediated interfaces. Current verdicts reach POST directly as experiment outcomes. This timing keeps current results distinguishable from earlier experience.

D.3 Research Planning before and after a Test

PRE receives the failure profile, candidate proposals, history, and available resources. It selects a principal bottleneck, ranks a bounded portfolio, and fixes the intervention, endpoints, stopping conditions, and budget. A principal change can be evaluated at several registered states. The main training portfolio and the discovery-comparison portfolios retain separate experiment identities.

POST receives the fixed PRE decision and independent verification outcomes and costs. It considers alternative explanations, recommends TRAIN or NO-TRAIN, selects eligible supervision, and records the next research question. Effect labels remain the verifier's output; candidate acceptance uses the predeclared rule below.

D.4 Policy Selection and the Next Round

Let $A_k \in \{0, 1\}$ be the predeclared scaffold-free acceptance result on $\mathcal{D}_{\text{select}}$. The retained policy for the next round is

$$\pi_{k+1} = \begin{cases} \widetilde{\pi}_{k+1}, & A_k = 1, \\ \pi_k, & A_k = 0. \end{cases} \tag{11}$$

For the reported final validation, integrity must pass, and the candidate is accepted if its selection success count increases, or if success ties and mean terminal goal-condition completion increases. Otherwise the parent is retained. Goal-condition evaluation is separate from policy input and does not alter causal Benefit labels. A NO-TRAIN decision also retains π_k. Algorithm 1 connects this selection to collection and research.

Algorithm 1 RePair research and policy-update procedure

Require: Initial policy π_0, history S_0, train-side pools, budgets and acceptance rule
$k \leftarrow 0$; initialize attempt and cost records
while the stopping rule permits another attempt **do**
  Fix π_k, S_k, the interface, and the attempt budget
  Collect fresh rollouts and construct evidence E_k
  $\mathcal{C}_k \leftarrow$ failure analysis using the permitted history views
  $\mathcal{P}_k \leftarrow$ PRE: rank and register a repair portfolio
  Execute paired tests; obtain independent effect labels
  if the attempt is protocol-invalid **then**
    Record the incident and cost; apply the registered recovery or stop
    continue
  end if
  $d_k \leftarrow$ POST: interpret outcomes and recommend TRAIN or NO-TRAIN
  Construct $\mathcal{D}_k^+$ under Eq. (2)
  if $d_k = \text{TRAIN}$ and $\mathcal{D}_k^+ \neq \varnothing$ **then**
    Check native labels; stop/recover on a failed label check
    Train $\widetilde{\pi}_{k+1}$ on the registered dual views
    Retain π_{k+1} by Eq. (11)
  else
    Record NO-TRAIN; $\pi_{k+1} \leftarrow \pi_k$
  end if
  Close the round and release eligible history into S_{k+1}
  Update attempt, patience, and cost records; $k \leftarrow k + 1$
end while
return retained policy and experiment records

The campaign permits at most ten scientifically valid attempts and stops after three consecutive valid attempts without promotion, subject to its additional integrity and resource stops. Invalid attempts follow the registered recovery procedure; their costs and human recovery actions remain in the campaign record. The observed stop is the three-valid-round no-promotion patience limit. The five-round record includes one earlier user-authorized promotion exception; it is not represented as an automatic acceptance decision. Strong research models can run this sequence after initial configuration; its unattended execution is measured separately from policy improvement.

D.5 Cross-Round Outcomes

The development campaign records five valid rounds and one invalid attempt. R1, R3, R4, and R5 yield no training update; R2 contains the beneficial intervention and the user-authorized promotion exception. Consequently, the sequence is not evidence of five successful updates or five rounds of

uninterrupted automatic improvement. Engineering recoveries are documented separately and are not erased by resuming.

The later validations reuse R5 trajectories and Analyzer outputs because the retained parent identity is unchanged. The guarded validation ends without a training update. The executable-strategy validation trains one candidate and completes selection with a rollback decision. A subsequent closed-history ownership check stops its bookkeeping closeout; this does not change the already completed episode outcomes or turn the run into a closed new formal round.

The selection entry initially scheduled five seed slots through a legacy path. An explicit correction retains the prescribed seed 17, after episode publication, and stops the additional jobs. Both arms have all 355 seed-17 tasks. The supplementary record discloses this amendment; extra-seed work is not counted as additional evidence or represented as having never occurred. Goal fractions are recomputed by replaying the recorded actions with no new policy calls.