Minimum Sufficient Repair Chains for Stateful Tool Agents

Anonymous authors
Paper under double-blind review

Abstract

Tool agents often fail through coupled errors. Replacing one suspicious action can leave stale evidence, an incorrect calculation, or an invalid transaction in the execution history. We introduce Minimum Sufficient Repair Chains. The framework searches over rollback boundaries and typed repair operators under an explicit cost. A process verifier records the state of the evidence, calculation, payload, transaction, and final response after each replay. A learned predictor ranks candidate interventions. Shadow replay admits a branch only after executable verification. The method returns the least costly verified chain within its constructed candidate space. We evaluate it on 300 medical record agent trajectories from ten MedAgentBench task families. The repair corpus contains 153 failed trajectories, 1,003 intervention candidates, and repeated counterfactual outcomes for 459 candidates. When one task family is held out at a time, the method achieves an 83.7% stable repair rate and 82.8% exact agreement with the minimum verified repair in that space. Its top three candidates cover 92.8% of failed trajectories. These results establish verified minimal intervention search as an effective approach to recovery in structured tool environments with transactions.

1 Introduction

Large language models increasingly act through tools, application programming interfaces, and stateful environments (Yao et al., 2023; Schick et al., 2023). Their trajectories combine information retrieval, calculation, payload construction, and external writes. Interactive benchmarks now test these abilities in software, web, enterprise, and medical environments (Liu et al., 2024; Zhou et al., 2024; Qin et al., 2024; Jimenez et al., 2024; Yao et al., 2025; Jiang et al., 2025). This broader capability creates a clear failure mode. An early query can retrieve the wrong evidence. A later calculation can preserve the error. A final write can then commit a state that is internally consistent but factually wrong. A terminal task reward detects the failure but does not identify the state to restore or the actions to regenerate. ToolMaze confirms that implicit tool errors are especially difficult because agents often trust corrupted observations and continue along the wrong path (Zhu et al., 2026).

Existing correction methods critique an answer, revise one action, or restart an episode with additional feedback (Shinn et al., 2023; Gou et al., 2024; Chen et al., 2024; Yuan et al., 2025). These methods address local errors well. Stateful execution adds a second problem: the system must choose the scope of the intervention. It may need to replace a malformed call, refresh the evidence, recompute a value, or replay a complete transaction. A reliable repair can therefore require several coordinated interventions at different boundaries. Recent rollback and runtime recovery systems demonstrate the value of checkpoints, dependency analysis, and guided reexecution (Li et al., 2025; Zhuang et al., 2026; Zhao et al., 2026; Qi et al., 2026). They also expose a remaining question: which verified combination of interventions repairs the failure at the lowest cost?

We formulate recovery as a search for a minimum sufficient repair chain. Each intervention specifies a rollback boundary, a typed repair operator, and a replay policy. The system restores the selected boundary and applies the intervention on an isolated branch. A process verifier then summarizes the new execution state. Search retains branches that make verified

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

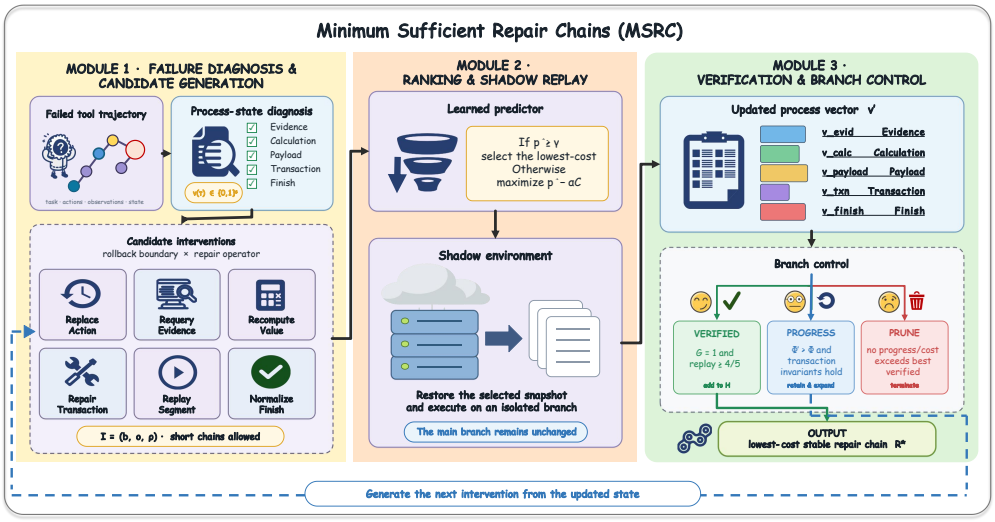

Figure 1: MSRC workflow. Failure diagnosis produces a process state and boundary–operator candidates. A learned ranker orders interventions for isolated shadow replay. The updated verifier state guides branch retention and pruning; the search returns the lowest-cost stable repair chain.

progress and accepts only branches that pass the executable task evaluator. Counterfactual replay provides direct evidence that the selected intervention changes the outcome (Shah, 2026; Luan et al., 2026; Li et al., 2026). The returned chain is sufficient because repeated replay restores task success. Its cost minimality is relative to the constructed candidate space and covers rollback scope, regenerated actions, token use, operator complexity, and chain length. This formulation targets structured tool environments with executable invariants and transaction boundaries.

Figure 1 gives the complete path from a failed trajectory to a committed repair. The framework separates three questions that are often collapsed into one prompt: where to roll back, what kind of repair to apply, and whether the repaired branch is safe to commit.

Our contributions are:

- We define agent recovery as a constrained minimum cost search problem over repair chains with multiple steps.
- We introduce a structured, multi-operator repair space and a process verifier that supports branch retention, pruning, and executable confirmation.
- We construct a MedAgentBench repair corpus grounded in replay. It provides stable counterfactual labels and folds separated by task family.
- We show that MSRC reaches 83.7% stable repair and 82.8% exact agreement with the minimum verified chain in the constructed pool on unseen task families. Its top three candidates cover 92.8% of failed trajectories.

2 Related Work

Language agents and interactive benchmarks. ReAct makes tool observations part of an agent's reasoning loop, and Toolformer studies how a language model learns to call external tools (Yao et al., 2023; Schick et al., 2023). These systems motivate evaluation that follows an entire sequence of decisions. AgentBench tests agents across eight interactive environments (Liu et al., 2024); WebArena supplies reproducible web tasks with executable success checks (Zhou et al., 2024); and ToolLLM pairs a large API collection with tool-use data and evaluation (Qin et al., 2024). SWE-bench evaluates repository changes with tests (Jimenez et al., 2024), while τ-bench adds user interaction, policy constraints, and database-state checks (Yao et al., 2025). MedAgentBench brings the same concern for executable outcomes

to FHIR-compatible medical records and physician-written tasks (Jiang et al., 2025). Together, these settings expose trajectories whose terminal outcome can be checked. Their evaluators establish whether a continuation succeeds, and their stored action histories reveal where an intervention can begin. MSRC connects those two forms of evidence: it uses the saved trajectory to choose a repair boundary, then applies the evaluator to each replayed continuation. This connection turns terminal success checks into a search signal over concrete recovery actions.

Reflection and correction. Reflexion carries verbal feedback into a later attempt (Shinn et al., 2023). CRITIC grounds revision in tool feedback (Gou et al., 2024), and Self-Debugging uses execution results to revise generated programs (Chen et al., 2024). Agent-R builds recovery data around erroneous steps through tree search (Yuan et al., 2025). These approaches establish the value of feedback after failure. In a stateful tool environment, that feedback must also determine which prior state to restore and which dependent operations to rerun. A revised response may still inherit a stale observation or an incorrect external write. Recovery therefore needs a unit of action that specifies the restored state, the replacement operation, and the continuation to evaluate. MSRC represents those choices as boundary–operator pairs, then composes verified pairs into a costed repair chain.

Process supervision and verification. Process supervision assigns credit to intermediate reasoning steps (Lightman et al., 2024). Math-Shepherd generates step-level labels for mathematical solutions (Wang et al., 2024), and outcome verifiers rank complete candidate answers (Cobbe et al., 2021). A tool agent also changes external state, so a useful intermediate signal must record evidence retrieval, calculated values, payload validity, transaction effects, and the final response. Our process vector makes these obligations explicit. Intermediate state and terminal outcome serve complementary purposes: the vector guides branch retention and identifies the next unresolved obligation, while the executable task evaluator determines final success. Repeated replay then checks whether that success survives stochastic continuation. This separation lets search preserve partial repairs without treating partial progress as a completed task.

Rollback, diagnosis, and counterfactual replay. GA-Rollback detects incorrect actions and triggers stepwise rollback (Li et al., 2025). AgentRewind aligns checkpoints of agent context and environment state (Zhuang et al., 2026). REVISE traces workflow dependencies to preserve unaffected work (Qi et al., 2026), and AgentTether uses a dependency graph to localize critical subtrajectories (Zhao et al., 2026). These methods motivate an explicit choice of rollback scope. Causal Agent Replay executes a downstream trajectory again to measure an intervention's effect (Shah, 2026); Repair or Resample examines the stability of that effect (Luan et al., 2026). Minimal Repair Family Recovery identifies jointly necessary events through graph-constrained replay (Li et al., 2026). The combined evidence suggests a decision sequence: localize the earliest state that must change, regenerate dependent actions, and verify the resulting trajectory. MSRC makes this sequence explicit by jointly selecting a boundary and typed intervention, preserving verified branches, and ranking complete chains by cost in its constructed candidate space.

3 Problem Formulation

Let an agent trajectory be

$$\tau = (s_0, a_0, o_0, s_1, a_1, o_1, \ldots, s_T), \tag{1}$$

where s_t is the environment state before action a_t and o_t is the resulting observation. A task evaluator $G(\tau) \in \{0, 1\}$ indicates final success. We consider a failed trajectory with $G(\tau) = 0$ and assume that environment snapshots can reproduce the state at selected boundaries.

An intervention is a tuple

$$I = (b, o, \rho), \qquad o \in \mathcal{O}, \tag{2}$$

where b is a rollback boundary, o is a repair operator, and ρ specifies the replay continuation. A repair chain $R = (I_1, \ldots, I_K)$ induces a shadow trajectory τ^R. The operator set is

$$\begin{aligned}\mathcal{O} = \{&\text{ReplaceAction}, \text{RequeryEvidence}, \text{RecomputeValue},\\ &\text{RepairTransaction}, \text{ReplaySegment}, \text{NormalizeFinish}\}.\end{aligned} \tag{3}$$

Because agent execution is stochastic, sufficiency is defined using repeated replay. If n_R replays produce m_R successes, the empirical success probability is $\hat{p}_R = m_R/n_R$. A chain is verified when $\hat{p}_R \geq \eta$ and n_R meets the required repeat count. We use $\eta = 0.8$ and five repeats for chain labels.

The repair cost is

$$C(R) = \lambda_r r(R) + \lambda_a a(R) + \lambda_t t(R) + \lambda_q q(R), \tag{4}$$

where r is normalized rollback scope, a is the burden from regenerated actions and interventions, t is normalized token use, and q is operator complexity. The target is

$$R^* = \arg\min_{R \in \mathcal{C}(\tau)} C(R) \quad \text{s.t.} \quad \hat{p}_R \geq \eta,\ G(\tau^R) = 1. \tag{5}$$

Here $\mathcal{C}(\tau)$ is the constructed, budgeted set of admissible repair chains. The objective evaluates an action as a repair target through the verified outcome of an intervention at its selected boundary.

4 Minimum Sufficient Repair Chains

4.1 Structured process state

After each replay, the verifier maps the branch to

$$v(\tau) = \left[v_{\text{evid}}, v_{\text{calc}}, v_{\text{payload}}, v_{\text{txn}}, v_{\text{finish}}\right] \in \{0,1\}^5. \tag{6}$$

The components indicate whether the branch used adequate evidence, derived the requested value correctly, built a valid payload, completed the required transaction, and returned the required terminal format. The progress potential is

$$\Phi(\tau) = \frac{1}{5} \sum_j v_j(\tau). \tag{7}$$

This compact state allows a branch to be retained when it repairs a local defect even before final task success. Domain facts are stored separately as typed fields such as numerical value, unit, timestamp, record identifier, resource type, and expected write operation.

4.2 Repair operators

ReplaceAction. Replace a malformed, unsupported, or incorrectly parameterized tool call while keeping valid prefix state. **RequeryEvidence.** Issue a corrected retrieval request and rebuild the evidence set when the original branch selected an incorrect code, record, date range, or patient identifier. **RecomputeValue.** Apply a deterministic calculator to typed evidence; numerical aggregation, unit conversion, and temporal ordering are executed outside the language model and returned as structured facts. **RepairTransaction.** Reconstruct and validate the complete payload for a state change, including references and required companion requests, before committing it on the shadow branch. **ReplaySegment.** Regenerate a bounded continuation from a retained rollback boundary when no more specific deterministic operator applies. **NormalizeFinish.** Preserve verified evidence and transactions while rewriting only the terminal response in the format required by the evaluator.

4.3 Candidate prediction and search

For each failed trajectory, we systematically enumerate admissible single-intervention boundary–operator pairs and use bounded search to construct short operator chains. A predictor estimates candidate success,

$$\hat{p}_\theta(I \mid x) = \sigma\big(f_\theta(x, I)\big), \tag{8}$$

from the task instruction, local trajectory context, original action, boundary, operator sequence, intervention cost, and the process vector after the first replayed intervention. The predictor combines TF–IDF text features with numeric features standardized within each training fold. It uses balanced logistic regression. The first replayed branch supplies the

process state for ranking continuation candidates. Candidates below the success probability gate are prioritized by

$$U(I \mid x) = \hat{p}_\theta(I \mid x) - \alpha C(I). \tag{9}$$

Search restores a snapshot at the selected boundary and executes the candidate on an isolated branch. Successful branches enter the verified set. Other branches are retained only when their process state improves, $\Phi(\tau^{R \oplus I}) > \Phi(\tau^R)$, or when they resolve a failing verifier dimension without invalidating a completed transaction. The system generates the next candidate from this updated state. It prunes a branch when its accumulated cost exceeds the best verified cost. Search stops when the frontier is empty or the budget is reached. It returns the least costly verified chain. The system does not commit repaired state before shadow verification.

Algorithm 1 Predictor guided minimum sufficient repair search

Require: Failed trajectory τ, snapshots S, verifier v, task evaluator G, depth d, replay budget B

Initialize frontier F with the unmodified failed branch; verified set $H \leftarrow \emptyset$
while $F \neq \emptyset$ and fewer than B candidates have been replayed do
    If any candidate has $\hat{p}_\theta \geq \gamma$, pop the least costly such candidate; otherwise pop the candidate with highest $\hat{p}_\theta - \alpha C$
    Restore its boundary in a shadow environment and execute the intervention
    Compute the new process vector v', potential Φ', and task outcome G
    if $G = 1$ and repeated success is at least η then
        Add the chain to H and update the best verified cost
    else if a failing verifier dimension is resolved or $\Phi' > \Phi$, and completed transaction invariants remain valid then
        Generate successor interventions and insert admissible successors into F
    end if
    Prune branches whose accumulated cost exceeds the best verified cost
end while
return the lexicographically minimum chain in H by cost, depth, boundary sum, and identifier

Minimum cost property. Assume a finite intervention space, nonnegative additive costs, complete successor generation, and no budget truncation. Algorithm 1 then returns the least costly verified chain in the enumerated space. Suppose a cheaper verified chain existed. Every prefix would cost less than the current best chain, so cost pruning could not remove it. Complete generation and finite search would eventually replay the full chain and place it in H. This contradicts selection of a more expensive result. Our implemented adaptive chain search is bounded, so these completeness assumptions do not establish global minimality. With the finite replay budget, minimum labels refer only to verified candidates in $\mathcal{C}(\tau)$.

4.4 Training labels

A replay candidate is positive only when it reaches the executable task objective in at least 80% of repeated runs. For each trajectory, we choose the minimum chain label among positive candidates. The lexicographic order is weighted cost, chain depth, the sum of rollback boundaries, and a deterministic identifier. This procedure produces success labels for candidates and minimum chain labels for trajectories. It does not treat an unverified critique as causal evidence.

5 Experimental Setup

5.1 Environment and corpus

We use MedAgentBench because it combines long tool trajectories with executable FHIR queries and writes (Jiang et al., 2025). The benchmark contains 300 tasks about specific patients across ten categories written by physicians. We execute all 300 tasks and retain

153 failed trajectories for repair. Candidate generation yields 1,003 rows: 832 single interventions and 171 chains with several interventions. Repeated counterfactual replay verifies 239 candidates. Among the 153 failed trajectories, 151 have at least one verified solution.

To measure generalization across task families, we hold out one task family at a time for every learned component. No task family contributes repair labels to its own test fold. The strict main comparison uses 459 candidates with five replay trials and exactly three repeated candidates per failed trajectory. Stable repair requires at least four successful trials out of five.

Candidate-space construction and audit. All 287 executed action boundaries from 154 failures were crossed with four single-intervention operators (1,148 planned cells); 832 compliant singles remained after applicability checks and one benchmark-incompatible exclusion. Score improvement and repair outcome were not admission filters. For 57 failures without a stable single repair, predictor-guided search retained three first-stage branches and expanded them to depth three, yielding 171 chains. The strict pool is shared by every method, but acquisition was not neutral: 153 cells came from MSRC rank and 150 from minimum cost. A fixed-seed, outcome-blind audit exhaustively repeated the 76 admitted singles on 14 stratified trajectories. It required 152 fresh calls, added two stable candidate labels, and left strict-pool coverage at 149/153. The main table remains on the original 459-row pool; Appendix H gives the filter rules, budget ceilings, failure decomposition, and post-audit refit sensitivity.

5.2 Implementation details

For a single intervention, the normalized cost is

$$C(I) = 0.35r + 0.25\min(A/8, 1) + 0.30\min(T/20000, 1) + 0.10q, \tag{10}$$

where r is rollback fraction, A is the number of generated actions, T is token use, and q is operator complexity. A chain starts with the cost of its first intervention. We add total chain tokens divided by 10^6 and a penalty of 0.05 for every later intervention. Search uses maximum depth three and beam width three. The candidate predictor applies balanced logistic regression to TF–IDF features from the task and trajectory. It also uses boundary, depth, cost, operator, and the available process vector from the first branch. Numeric features are standardized using statistics from the training fold only. At inference, candidates with predicted success of at least 0.5 are ordered by the lexicographic minimum cost objective. If none reaches the threshold, Equation 9 orders the candidates with $\alpha = 0.15$. Stored shadow replays provide every reported outcome label.

Appendix C tests all four cost groups with 2,000 independent $\pm 20\%$ perturbations. All 151 verified minimum chain labels remain unchanged. The central 95% ranges are $[0.830, 0.850]$ for stable repair and $[0.821, 0.841]$ for exact minimum chain agreement.

5.3 Baselines

We compare no repair, the least costly single intervention in the shared pool, a fixed operator schedule, a rule based adaptive policy, and an oracle that selects from true replay outcomes. All methods except the oracle receive the same failed trajectory, candidate pool, and selection budget. Cost and exact minimum chain agreement are computed using Equation 4 and the verified labels.

5.4 Metrics

The primary metric is stable repair rate, measured over all 153 failed trajectories. We also report mean intervention count, normalized repair cost, and exact agreement with the minimum verified chain, measured over the 151 trajectories with a verified solution in the constructed pool. For the candidate predictor, we report the area under the receiver operating characteristic curve, average precision, Brier score, expected calibration error, verified success at rank one, and verified coverage within the top three. Confidence intervals for stable repair use resampling at the trajectory level.

6 Results

6.1 Main results

Figure 2 reports results with strict separation by task family; Appendix Table 1 lists the exact values. MSRC repairs 83.7% of failed trajectories and selects the exact minimum verified chain in the constructed pool for 82.8%. It improves stable repair by 26.1 percentage points over the cheapest single intervention. Exact agreement with the minimum chain improves by 24.5 points. The mean selected chain uses 1.35 interventions at a normalized cost of 0.444. The oracle selects from verified outcomes in the shared candidate pool and reaches 97.4% stable repair.

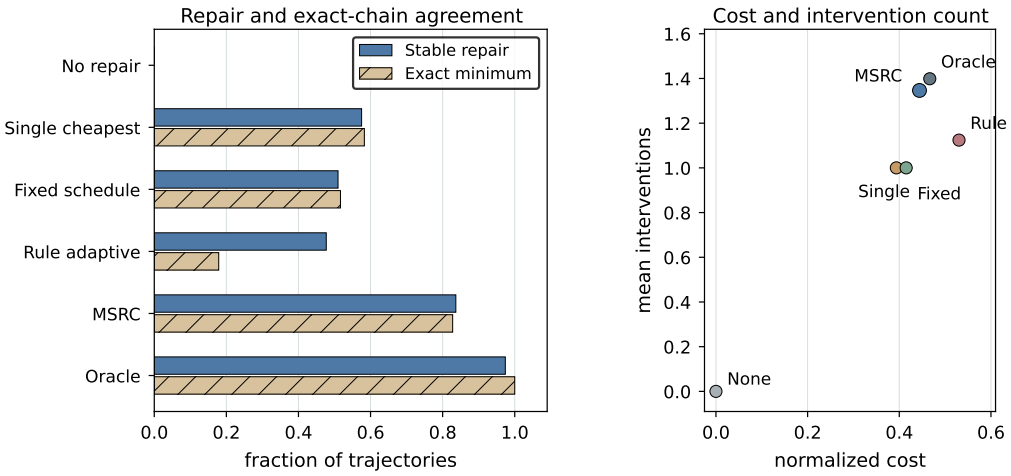

Figure 2: Main results with task-family holdout. Blue bars show stable repair over 153 failures; hatched ochre bars show exact minimum-chain agreement among 151 trajectories with verified solutions. The right panel plots normalized cost against mean interventions; the oracle uses stored outcomes.

The candidate predictor obtains an area under the receiver operating characteristic curve of 0.775, average precision of 0.461, a Brier score of 0.192, and expected calibration error of 0.176. Selection at rank one finds a verified repair for 68.0% of trajectories. The top three candidates raise coverage to 92.8%. This gain supports a search frontier with three candidates.

The trajectory level analysis confirms these gains. Relative to the cheapest single intervention, MSRC gains 26.1 points. The paired bootstrap 95% interval is $[19.0, 34.0]$ points. The exact McNemar test gives $p = 1.13 \times 10^{-10}$, with 42 successes unique to MSRC and two unique to the baseline. The gains over the fixed schedule and the rule based policy are 32.7 points ($[24.8, 40.5]$, $p = 1.65 \times 10^{-13}$) and 35.9 points ($[24.8, 46.4]$, $p = 7.72 \times 10^{-9}$).

6.2 Search budget

Figure 3 measures how candidate budget changes verified coverage; Appendix Table 2 lists the plotted values. The experiment reuses task-family-fold predictions and stored replay labels.

The first three candidates cover 92.8% of trajectories. Search stops at the first verified repair and inspects only 1.48 candidates on average. Raising the cap from three to ten adds 3.3 coverage points and only 0.08 average inspections. Useful repairs concentrate near the front of the learned ordering.

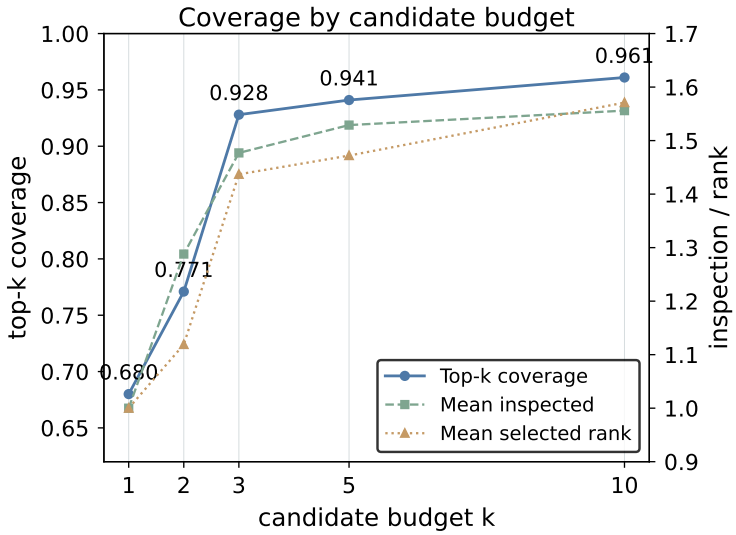

Figure 3: Search-budget curves. Solid: top-k verified coverage (left axis); dashed: mean candidates inspected; dotted: mean rank of selected success (right axis). Search stops at the first verified repair.

6.3 Ablations

Figure 4 isolates the execution mechanisms on 57 hard trajectories that no single intervention repairs stably; Appendix Table 3 gives the counts. The complete adaptive chain repairs 55 cases. The process vector and branch retention each raise stable repairs from one to 55. The deterministic executor adds seven stable repairs, while transaction validation adds nineteen. These gains show the value of preserving verified progress, deterministic computation, and consistent side effects.

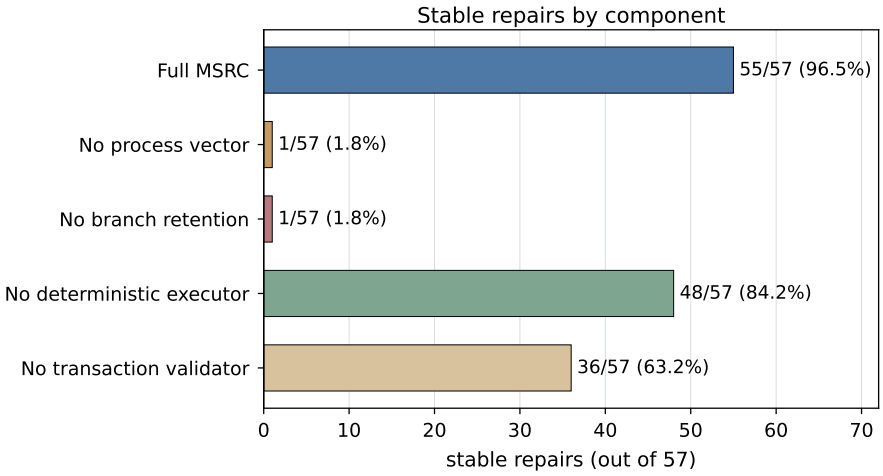

Figure 4: Mechanism ablation on 57 hard trajectories. Bars count stable repairs across five replays; at least four successes define stability. Full MSRC appears first, followed by component removals.

The successful searches include 3 chains with one step, 41 with two steps, and 13 with three steps. Most hard failures therefore require coordinated interventions across several boundaries. Appendix G reports the detailed results for each task family.

Appendix Figure 5 traces one repaired HbA1C task through evidence refresh and transaction repair.

7 Discussion

The experiments support structured search for agent repair. Candidate scoring narrows the search space, and replay supplies causal confirmation. The process vector shows which obligations have been restored. Typed operators separate evidence retrieval, arithmetic, transaction construction, and final formatting. Selection of the minimum chain gives recovery a clear optimization target and an interpretable cost. Search budget controls the number of replayed candidates, while the chain objective controls intervention burden. These two efficiency axes are visible in the budget and main-result figures.

Interaction among recovery mechanisms. The ablations clarify why the search components are coupled. The process vector determines which obligations remain unresolved, and branch retention allows a partially repaired trajectory to survive until those obligations can be addressed. Removing either mechanism leaves one stable repair among the 57 hard cases. Deterministic execution and transaction validation act later in the chain: one fixes calculations and reproducible transformations, while the other checks state-changing writes. Their ablation losses measure different failure modes and should be interpreted as component contributions within the complete pipeline. The results support coordinated repair across several boundaries, consistent with the predominance of two-step and three-step successful chains.

Interpreting the search frontier. The top-three coverage result measures the presence of a verified repair near the front of the learned ordering. The 83.7% stable-repair result measures the chain actually selected under the decision rule and repeat protocol. Their difference identifies room for improving candidate choice within the existing pool. The oracle result characterizes the verified candidate pool itself. Together, these measures separate coverage, selection, and attainable pool performance, which is important when comparing recovery policies that share stored candidates.

Deployment without native rollback. MSRC supports guarded validation when production interfaces do not expose snapshots. Read only calls and schema checks first reject invalid candidates. A calibrated simulator can then predict state transitions and verifier outcomes. Its uncertainty determines whether a branch may advance. A candidate with high confidence reaches one guarded production commit. Idempotency keys, compensating transactions, authorization limits, and human approval can protect this commit. This design limits production interaction to the smallest verified candidate while preserving the same search objective.

8 Limitations

The executable evaluation covers one medical agent environment and seven task families with repair samples. Single interventions were systematically enumerated, but adaptive chains were bounded and the shared strict pool was partly acquired by the MSRC ranker. The 14-trajectory outcome-blind audit checks admitted singles, not the complete chain space. Thus minimality and the 97.4% oracle ceiling apply only to the constructed pool. Cost calibration must reflect the operational risk of each environment. Production use without snapshots also requires calibrated simulators, uncertainty thresholds, and safeguards at commit time.

Future directions. Future work will test the boundary–operator–verifier interface in non-medical stateful environments and expand the bounded offline candidate pool toward online branch generation.

9 Conclusion

Minimum Sufficient Repair Chains searches rollback boundaries and typed interventions in stateful tool environments. A process verifier retains useful partial repairs, and repeated shadow replay supplies stable success labels. On MedAgentBench, MSRC repairs 83.7% of failed trajectories under task-family holdout and selects the minimum verified chain in its constructed candidate pool for 82.8% of feasible cases. The framework makes recovery a costed search over executable, verified changes while preserving valid prior work.

AI Use Statement

In this work, generative AI tools were used to generate agent rollouts and candidate repair actions as experimental data, assist with code implementation, and support language editing. All reported replay outcomes and numerical results were produced by executable evaluation pipelines and checked against stored experiment artifacts. Citations and factual claims were checked against their source materials. The authors reviewed all work supported by AI and take responsibility for the final content, including text, claims, code, and artifacts produced with the aid of generative AI.

Ethics Statement

This work studies benchmark agents in an isolated electronic health record environment. Deployment in patient care would require clinical governance, authorization controls, audit logs, privacy review, and human confirmation for operations that change state. The reported results are research measurements and provide no clinical guidance or safety certification for a live system.

Reproducibility Statement

The supplementary material includes task family folds, candidate definitions, minimum chain labels, repeated replay outcomes, trained predictor artifacts, and scripts for reproducing the main tables. Section 4 specifies the objective, operators, verifier state, pruning rule, and label threshold. The appendices record data counts, ranker diagnostics, cost sensitivity, and the candidate-space construction audit.

References

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to Self-Debug. In International Conference on Learning Representations, 2024. doi:10.48550/arXiv.2304.05128.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021. doi:10.48550/arXiv.2110.14168.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In International Conference on Learning Representations, 2024. doi:10.48550/arXiv.2305.11738.

Yixing Jiang, Kameron C. Black, Gloria Geng, Danny Park, James Zou, Andrew Y. Ng, and Jonathan H. Chen. MedAgentBench: A realistic virtual EHR environment to benchmark medical LLM agents. arXiv preprint arXiv:2501.14654, 2025. doi:10.48550/arXiv.2501.14654.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In International Conference on Learning Representations, 2024. doi:10.48550/arXiv.2310.06770.

Bingjie Li, Yumeng Song, Zhongming Yao, and Tianyi Li. Localizing emergent failures in agentic AI: Recovering minimal repair families via counterfactual replay. arXiv preprint arXiv:2608.29228, 2026. doi:10.48550/arXiv.2608.29228.

Xingzuo Li, Kehai Chen, Yunfei Long, Xuefeng Bai, Yong Xu, and Min Zhang. Generator-Assistant stepwise rollback framework for large language model agent. arXiv preprint arXiv:2503.02519, 2025. doi:10.48550/arXiv.2503.02519.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In International Conference on Learning Representations, 2024. doi:10.48550/arXiv.2305.20050.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In International Conference on Learning Representations, 2024. doi:10.48550/arXiv.2308.03688.

Zhongwen Luan, Xiaoyu Zhang, Ming Hu, Yue Yang, Jiongchi Yu, and Xiaohong Chen. Repair or resample? rethinking failure debugging in LLM multi-agent systems. arXiv preprint arXiv:2608.25920, 2026. doi:10.48550/arXiv.2608.25920.

Ruoling Qi, Xuaner Wu, Penghang Liu, Jian Chen, and Yirui Liu. REVISE: Validity-guided recovery for online revisions in agent workflows. arXiv preprint arXiv:2609.00643, 2026. doi:10.48550/arXiv.2609.00643.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In International Conference on Learning Representations, 2024. doi:10.48550/arXiv.2307.16789.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In Advances in Neural Information Processing Systems, volume 36, 2023. doi:10.48550/arXiv.2302.04761.

Jaineet Shah. Causal agent replay: Counterfactual attribution for LLM-agent failures. arXiv preprint arXiv:2606.08275, 2026. doi:10.48550/arXiv.2606.08275.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In Advances in Neural Information Processing Systems, volume 36, 2023. doi:10.48550/arXiv.2303.11366.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, pp. 9426–9439, 2024. doi:10.18653/v1/2024.acl-long.510.

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. doi:10.48550/arXiv.2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In International Conference on Learning Representations, 2025. doi:10.48550/arXiv.2406.12045.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-R: Training language model agents to reflect via iterative self-training. arXiv preprint arXiv:2501.11425, 2025. doi:10.48550/arXiv.2501.11425.

Chenyu Zhao, Shenglin Zhang, Wenwei Gu, Yongqian Sun, Dan Pei, Chetan Bansal, Saravan Rajmohan, and Minghua Ma. AgentTether: Graph-guided diagnosis and runtime intervention for reliable LLM agent operation. arXiv preprint arXiv:2607.06273, 2026. doi:10.48550/arXiv.2607.06273.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In International Conference on Learning Representations, 2024. doi:10.48550/arXiv.2307.13854.

Dongsheng Zhu, Xuchen Ma, Yucheng Shen, Xiang Li, Yukun Zhao, Shuaiqiang Wang, Lingyong Yan, and Dawei Yin. When tools fail: Benchmarking dynamic replanning and anomaly recovery in LLM agents. arXiv preprint arXiv:2606.05806, 2026. doi:10.48550/arXiv.2606.05806.

Yu Zhuang, Kefei Chen, Yitong Duan, Shuxin Zheng, Jian Li, and Xu-Yao Zhang. AgentRewind: Recoverable execution for long-horizon LLM agents. arXiv preprint arXiv:2608.14380, 2026. doi:10.48550/arXiv.2608.14380.

A Data and Evaluation Details

The corpus contains 300 raw trajectories. Repair search is applied to 153 failed trajectories from seven task families. Candidate generation produces 1,003 rows, of which 239 are verified positives. There are 151 minimum chain labels and two trajectories without a verified solution. The strict comparison uses 459 candidates with five repeated trials. Learned methods are trained seven times, each time holding out one complete task family. A chain is stable when it succeeds in at least four of five trials. Exact minimality compares the selected chain identifier with the verified lexicographic minimum defined by cost, depth, the sum of rollback boundaries, and identifier.

B Detailed Result Tables

Table 1: Main results on 153 failed trajectories with task-family holdout. Stable repair requires at least four successes in five replays. ↑ is better; ↓ is better.

Method	Stable repair ↑	Interventions ↓	Cost ↓	Exact min. ↑
No repair	0.000	0.000	0.000	–
Single cheapest	0.575	1.000	0.394	0.583
Fixed schedule	0.510	1.000	0.415	0.517
Rule adaptive	0.477	1.124	0.530	0.179
MSRC	0.837	1.346	0.444	0.828
Oracle	0.974	1.399	0.467	1.000

Table 2: Verified repair coverage as the maximum number of inspected candidates increases.

Budget k	Coverage ↑	Mean inspected ↓	Mean success rank ↓
1	0.680	1.000	1.000
2	0.771	1.288	1.119
3	0.928	1.477	1.437
5	0.941	1.529	1.472
10	0.961	1.556	1.571

Table 3: Mechanism ablations on 57 hard trajectories under five repeated trials.

Variant	Stable repairs ↑	Rate ↑
Full MSRC	55/57	0.965
Without process vector	1/57	0.018
Without branch retention	1/57	0.018
Without deterministic executor	48/57	0.842
Without transaction validator	36/57	0.632

C Repair Chain Case Study

Figure 5 illustrates a repaired HbA1C task. The original branch does not obtain the required observation and ends without the required order. ReplaceAction corrects the initial call but still leaves evidence, payload, transaction, and finish checks unresolved. RequeryEvidence executes the canonical A1C query and establishes that no measurement is present, moving process progress from 0.2 to 0.6. RepairTransaction then creates the required FHIR ServiceRequest for LOINC 4548-4 and returns the expected terminal value. The final verifier vector changes from $[0, 1, 0, 0, 0]$ to $[1, 1, 1, 1, 1]$, and the chain succeeds in all five repeated trials.

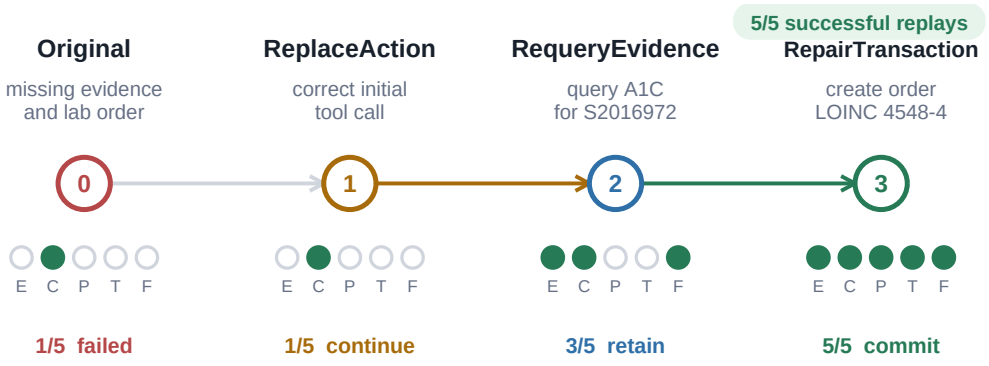

Figure 5: Repair chain for trajectory task10_16. The process vector improves from the failed branch to a fully verified branch after three typed interventions. All five repeated replays succeed.

D Candidate Ranker Diagnostics

Table 4 reports diagnostic variants of the candidate ranker under the same task family protocol as Table 1. Selection based only on probability increases repair coverage at higher intervention cost. Process state features produce the largest gain in exact minimum chain agreement.

Table 4: Candidate ranker diagnostics on 153 failed trajectories.

Variant	Stable repair ↑	Cost ↓	Exact min. ↑
Full MSRC	0.837	0.444	0.828
No boundary prediction	0.824	0.434	0.834
No explicit operator feature	0.843	0.436	0.854
Probability only selection	0.902	0.496	0.821
No process vector	0.824	0.451	0.623
Structured features only	0.843	0.443	0.662
Text features only	0.784	0.449	0.675

E Cost Function Sensitivity

We perturb four additive cost groups: rollback, the burden from regenerated actions and interventions, token use, and operator complexity. Each group receives an independent multiplier sampled uniformly from $[0.8, 1.2]$. For each of 2,000 draws, we recompute the minimum verified chain labels over all 1,003 candidates. We then rerun cost aware selection over the 459 candidates with five replays. Success probabilities from the held out task family remain fixed, which isolates the effect of the cost definition. Table 5 reports the resulting distribution.

Table 5: Sensitivity to independent $\pm 20\%$ perturbations of all cost groups. Intervals are the 2.5th and 97.5th percentiles across 2,000 draws.

Metric	Nominal	Mean	Std.	95% range
Minimum label retention	1.000	1.000	0.000	$[1.000, 1.000]$
Stable repair	0.837	0.842	0.007	$[0.830, 0.850]$
Exact minimum agreement	0.828	0.833	0.007	$[0.821, 0.841]$
Selected perturbed cost	0.444	0.443	0.029	$[0.389, 0.498]$

Every nominal minimum chain label is retained across all draws. Stable repair and exact minimum agreement each remain within 1.4 percentage points of their nominal values across the central 95% interval. Endpoint tests for one group at a time also retain every label. Perturbations to action burden and operator complexity leave both task metrics unchanged. Perturbations to rollback and token use shift selection by at most two trajectories. Minimum chain supervision and the resulting search behavior are therefore locally robust to the specified cost weights.

F Search Procedure

The search frontier first applies the predicted success gate described in Section 4.3. It orders eligible candidates by the lexicographic minimum cost objective. Equation 9 orders the fallback frontier when no candidate reaches the gate. Each node stores an environment snapshot, accumulated repair chain, process vector, and cost. Expanding a node applies one pair of a boundary and an operator in a shadow environment. A successful node enters the verified set and is replayed to estimate stability. A node that has not terminated remains on the frontier only after measurable process improvement. Once a verified chain exists, nodes with greater accumulated cost are pruned. The final output is the verified chain that minimizes Equation 5.

G Generalization by Task Family

Table 6 reports stable repair for each held out task family. MSRC reaches at least 85.7% on five of seven families. The largest gains occur on task10 and task9. These tasks often require

an evidence refresh followed by a calculation or transaction repair. On simpler families, the policy retains inexpensive single interventions when deeper repair is unnecessary.

Table 6: Stable repair by held out task family. The final row reports the number of trajectories repaired by MSRC.

	task1	task10	task3	task5	task6	task7	task9
N	28	20	29	14	3	30	29
Single cheapest	.679	.300	1.00	.714	1.00	.567	.138
MSRC	.679	.950	1.00	1.00	1.00	.567	.931
MSRC successes	19	19	29	14	3	17	27

H Candidate-Space Construction Audit

All minimality statements in this work are scoped to the minimum verified repair within the constructed candidate space. We separate candidate construction into three layers so that the scope of this claim is explicit.

Exhaustive single-intervention matrix. Before excluding benchmark-incompatible cases, the 154 failed trajectories contain 287 executed action boundaries. We cross every boundary with the four single-intervention operators (ReplaceAction, RequeryEvidence, RecomputeValue, and ReplaySegment), producing all $287 \times 4 = 1{,}148$ planned cells with no missing cells and no execution errors. Of these cells, 928 are executable and 844 satisfy the operator-specific compliance predicate. The 220 inapplicable cells are all RecomputeValue calls for which no structured numeric or temporal evidence exists before the boundary. After excluding one trajectory with a documented tool-schema/evaluator conflict, 832 compliant single-intervention candidates remain. Candidate admission uses only benchmark compatibility, operator applicability, executability, and syntactic compliance; neither score improvement nor replay success is used as an admission filter.

Bounded adaptive chain search. The chain layer is deliberately not exhaustive. It is invoked for the 57 benchmark-compatible trajectories without a stable single-intervention repair. For each trajectory, the search retains the three highest-scoring first-stage branches and expands each branch to depth at most three. After each failed stage, a deterministic five-dimensional process vector selects the next repair operator. This procedure yields 171 stored chain candidates. Consequently, the single-intervention layer is an exhaustive boundary–operator matrix, whereas the multi-intervention layer is a bounded, predictor-guided search.

Shared strict evaluation pool. The resulting dataset contains 1,003 candidates: 832 single interventions and 171 chains. The strict main comparison uses only the 459 candidates with five stored replay trials, exactly three candidates for each of 153 trajectories. Every selector and ablation receives this same pool and chooses at most one repair plan per trajectory; the Oracle differs only in its access to the stored replay labels. The acquisition policy for this replay pool is not completely selector-neutral: 153 newly replayed candidates were selected by the MSRC rank, 150 by minimum cost, while 99 single candidates and 57 chain candidates were available from earlier experiments. We therefore claim a shared candidate and selection budget, but do not claim that acquisition of the strict replay pool is unbiased toward every selector.

Oracle ceiling as a function of candidate budget. To expose dependence on pool size, we order the three strict candidates for each trajectory by weighted cost, chain depth, rollback-boundary sum, and a deterministic identifier. The fraction of trajectories for which the first k candidates contain at least one verified-success repair is 58.2% for $k = 1$, 66.0% for $k = 2$, and 97.4% for $k = 3$. These values are oracle ceilings for the corresponding stored candidate budgets and should not be interpreted as performance of an implementable selector.

Coverage and selection failures. Every evaluated trajectory has at least one admitted candidate, so there are no admissible-candidate generation failures. In the full stored pool, 2 of 153 trajectories have no verified-success candidate. In the strict five-replay pool, 4

of 153 trajectories have no verified-success candidate. MSRC succeeds on 128 trajectories. Its 25 failures therefore decompose into four strict-pool coverage gaps and 21 selection errors among the 149 trajectories for which the strict pool contains a verified repair (14.1% conditional selection-error rate).

Completed outcome-blind exhaustive audit. Before inspecting outcomes, we selected two trajectories from each of seven task families by the smallest SHA-256 hashes of msrc-candidate-audit-v1:<trajectory-id>. The 14 selected trajectories contain all 76 admitted single-intervention cells in their constructed space. Thirty-eight cells already had five stored replays; the other 38 were brought to five trials by retaining the original run and executing four new trials. This required 152 fresh calls, four more than the initial estimate of 148 because two partially repeated cells were re-run under the standardized protocol. All 152 calls completed without infrastructure errors.

The audit newly verifies two RequeryEvidence cells, each at 5/5 success. Both were already one-shot positives. The strict five-replay table grows from 459 to 497 rows, but strict-pool coverage remains 149/153 and full-pool coverage remains 151/153. The original main table is kept fixed. A separate post-audit leave-one-task-family-out refit improves stable repair from 128/153 (83.7%) to 130/153 (85.0%), while exact minimum-label agreement falls from 82.8% to 64.9%. None of the 151 trajectory-level oracle minimum labels changes. The refit changes 31 selections: it gains two verified minima in task1 but chooses an equally successful, more expensive candidate on 29 task3 trajectories. These are sensitivity results, not a revised main-table estimate.

Under the enlarged strict pool, MSRC's 23 failures consist of four pool noncoverage cases and 19 selection errors among the 149 covered trajectories (12.8% conditional selection-error rate). The audit is exhaustive only over admitted single interventions in the sampled trajectories. It neither exhausts the adaptive chain space nor eliminates the non-neutral acquisition of the original shared strict pool.