MINIMAL REPAIR SUBTRAJECTORIES: VERIFIER-CERTIFIED LOCAL RECOVERY FOR LONG-HORIZON AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon agents can fail because one incorrect tool action makes several later actions incompatible. Step repair may then be too narrow, whereas restarting discards a useful prefix. We introduce the *Minimal Repair Subtrajectory* (MRS): the shortest contiguous original span for which an admissible replacement, followed by executable replay, restores verifier-confirmed success. Minimality is explicit about its candidate space. Our training-free doctor–patient–verifier procedure first returns a bounded local-recovery certificate; exhaustive elimination of shorter admissible spans upgrades it to an exact MRS certificate. On τ^2, bounded search recovers 34 of 53 strict-replay-eligible failures, including five multi-step repairs. Across 960 controlled failures, MRS retains 100% verifier success while covering 1.98–3.16$\times$ fewer original positions than successful wider units, and its compact supervision outperforms full trajectories under matched budgets. An exhaustive audit certifies exact MRS boundaries for at least 32 of 41 successful public-environment recoveries. Across three such environments, a family-disjoint router raises recovery from 10.46% to 14.38% over always restart and reduces normalized cost by 15.8% while passing registered recovery non-inferiority and cost gates.

1 INTRODUCTION

Language agents couple LLMs with memory, actions, and feedback (Sumers et al., 2024; Yao et al., 2023) to operate across customer service (Yao et al., 2024), software and web environments (Jimenez et al., 2024; Zhou et al., 2024b; Drouin et al., 2024), and embodied or transactional tasks (Yao et al., 2022; Deng et al., 2023). Their success depends on extended chains of tool selection, argument construction, observation interpretation, and replanning (Schick et al., 2023; Qin et al., 2024; Li et al., 2023; Huang et al., 2024b). When an early action corrupts the state consumed downstream, repairing the visibly wrong step may be too narrow, while restarting the entire trajectory discards a potentially valuable prefix.

This creates a missing systems primitive: *what is the smallest verified part of a failed trajectory that must change, under a declared repair library?* Existing abstractions answer adjacent questions. Step-level attribution and process supervision identify suspicious actions (Lightman et al., 2024; Wang et al., 2024b; Zhang et al., 2025b); self-correction and retry generate another trajectory (Shinn et al., 2023; Madaan et al., 2023). A repair system instead needs a success-restoring interval with explicit boundaries, an executable replacement, and evidence that the environment accepts the result.

Replayable agent environments make this stronger objective attainable. A checkpoint preserves the successful prefix, replay recomputes the consequences of a candidate replacement, and a task verifier decides recovery. Together, these components support certificates based on execution rather than model self-judgment. We study this setting across deterministic controlled tasks and public environments with snapshots, executable state, or binary goal verification.

We call this unit the *Minimal Repair Subtrajectory*: the shortest original contiguous span whose admissible replacement succeeds under a specified candidate space. MRS is the trajectory-level rewrite dual of delta debugging (Zeller & Hildebrandt, 2002). A doctor proposes anchors and replacements, a patient supplies the failed execution, and an independent verifier certifies the replay.

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.

Bounded search yields certified local recovery; exhaustive elimination of shorter admissible spans yields candidate-space minimality. The method is post-hoc, training-free, and explicitly budgets verifier calls.

The empirical evidence makes repair granularity consequential. In the public τ^2 benchmark (Yao et al., 2024; Barres et al., 2025), bounded MRS-style search recovers 34 of 53 strict-replay-eligible failures, including five multi-step repairs. Controlled dependency tasks then make the structure measurable relative to a frozen replacement library: 72% of 81,920 recoverable parameterized DEPENDENTTOOLS failures have no passing one-position span in that library. On a 960-failure audit, MRS matches the 100% recovery of width-five windows, corrected suffixes, and full-trajectory replacement while covering 1.98–3.16$\times$ fewer original trajectory positions.

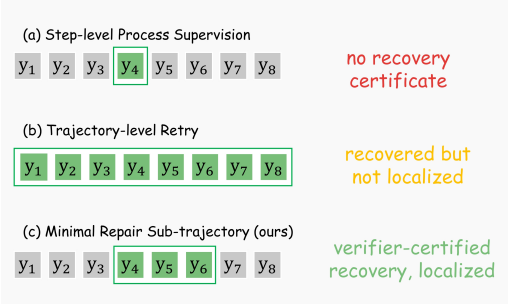

Figure 1: Three repair-unit granularities. (a) Step attribution localizes suspicion but does not establish recovery. (b) Full restart regenerates the run and discards the prefix. (c) MRS replaces the shortest contiguous original span that succeeds under the declared candidate space and executable verifier.

The benefit extends from repair to reuse. At a 1,400-token budget, MRS demonstrations outperform full trajectories in 10 of 14 configurations and recover 148 versus 131 of 160 paired failures. With matched fine-tuning examples, MRS raises recovery from 51.9% to 67.3%. Under matched token mass, it raises candidate-space exact localization from 42.1% to 58.9%. Public ToolSandbox, AppWorld, and ALFWorld experiments reveal and operationalize a prefix-validity axis: preserve a trustworthy prefix with MRS and restart a globally misdirected plan.

We make four contributions:

- We introduce MRS, a candidate-space-relative repair unit that couples minimal contiguous coverage with verifier-backed replay.
- We develop a training-free doctor–patient–verifier search that returns certified recoveries with bounded verification cost.
- We isolate repair-unit granularity from replacement quality, showing full controlled recovery while covering substantially fewer original positions.
- We demonstrate that MRS converts failures into high-density supervision and operationalize prefix validity with a held-family router that improves recovery and cost over always restart.

2 RELATED WORK

Recent literature investigates failure attribution within long-horizon tasks. Step-level attribution formalizes which action caused a failure (Zhang et al., 2025b), while AgenTracer (Zhang et al., 2025a) and AgentDebug (Zhu et al., 2025) isolate root causes through counterfactual replay or taxonomic categorization. MRS advances from locating a suspect step to executing a multi-step contiguous span search, determining the boundaries of successful recovery, and certifying the resulting repair through a deterministic task verifier rather than model self-verification, whose limitations have been documented (Huang et al., 2024a; Stechly et al., 2025).

Process Reward Models (PRMs) score individual reasoning steps (Lightman et al., 2024; Wang et al., 2024b). AgentPRM reformulates action values as goal progress (Xi et al., 2026), while CSO verifies critical decisions for post-training (Li et al., 2026). MRS adds the complementary question of repair extent: anchor detection identifies where to intervene, whereas span search determines how much of the trajectory must change. Tree-search methods such as LATS (Zhou et al., 2024a) replan entire futures; MRS identifies the shortest sufficient contiguous span.

Self-correction mechanisms like Reflexion (Shinn et al., 2023), Self-Refine (Madaan et al., 2023), and CRITIC (Gou et al., 2024) rely on text feedback or full re-runs, leaving the repair unit implicit. ExpeL (Zhao et al., 2024), ETO (Song et al., 2024), AgentHER (Ding, 2026), and ECHO (Hu et al., 2025) relabel or reuse experience; workflow memory and skill libraries instead retain reusable procedures (Wang et al., 2025b; 2024a). Closer to our setup, STeP (Chen et al., 2025) constructs self-reflected trajectories, Agent-R (Yuan et al., 2025) splices corrected branches, and DoVer (Ma et al., 2025) verifies interventions. MRS preserves the original goal and makes contiguous minimality an explicit, verifier-checkable target.

The failure distributions analyzed in this work span established agent frameworks including MetaGPT (Hong et al., 2024) and AutoGen (Wu et al., 2024). Broader interactive evaluation covers real-world APIs, executable code, and multi-turn progress metrics (Qin et al., 2024; Yang et al., 2023; Ma et al., 2024; Liu et al., 2024; Mialon et al., 2024; Wang et al., 2024c); scalable trajectory synthesis provides complementary training data (Xu et al., 2025). Our formulation is the structural dual of delta debugging (Zeller & Hildebrandt, 2002): delta debugging isolates failure-inducing inputs, while MRS identifies success-restoring action spans. This positions MRS within the broader automated program repair lineage (Le Goues et al., 2012; Mechtaev et al., 2016), operating at the trajectory level rather than the source-code level.

3 Problem Setup and Certificate Scope

Let $\tau^- = (y_1, \ldots, y_T)$ be a failed trace, where y_t is an executed action and its resulting observation, and let s_{i-1} be the checkpoint before action i. For an interval $z = [i, j]$, its *coverage width* is $|z| = j - i + 1$. The frozen set $\mathcal{C}(z; s_{i-1})$ contains admissible replacement sequences $\hat{z}$; $|\hat{z}|$ may differ from $|z|$. Coverage width measures original trajectory positions; replacement actions, verifier calls, and runtime cost are separate quantities.

$\mathrm{Replay}(\tau^-, z, \hat{z})$ restores s_{i-1}, executes $\hat{z}$, and re-executes the recorded suffix while recomputing observations. With external verifier V, the candidate-space minimum is

$$h^*_{\mathcal{C}}(\tau^-) = \min_z \{|z| : \exists \hat{z} \in \mathcal{C}(z; s_{i-1}),\ V(\mathrm{Replay}(\tau^-, z, \hat{z})) = 1\}. \tag{1}$$

An MRS is any passing span with width $h^*_{\mathcal{C}}$. Thus, $h^*_{\mathcal{C}} > 1$ certifies that every one-position candidate in the declared space fails.

Two certificate levels. A *local-recovery certificate* records a checkpoint, an executed repair or regenerated suffix, and a passing verifier outcome. An *exact MRS certificate* additionally records the frozen candidate space and failures of every shorter admissible span under recorded-suffix replay. Accordingly, bounded search certifies recovery; exhaustive controlled search establishes distribution-wide candidate-space minimality, while an exhaustive audit of successful public recoveries tests the same property under benchmark-native execution.

4 Empirical Trajectory Analysis

4.1 Multi-step repair on τ^2

On τ^2 (Yao et al., 2024; Barres et al., 2025), 149 failures are normalized from 442 source trajectories. The strict-replay protocol admits 53 failures, and bounded search recovers 34 (64.2%). Five returned repairs span multiple steps, providing public-benchmark evidence of successful multi-step local recovery. These bounded-search outcomes carry local-recovery certificates; candidate-space minimality is measured by the exhaustive controlled experiments below.

4.2 Controlled validation on DependentTools

To measure repair extent under controlled dependencies, we construct DependentTools with seven service-ticket, diagnosis, and patch templates. Each injects a corruption that propagates through mandatory downstream tool calls.

We generate 81,920 recoverable parameterized failures over 15 seeds. The candidate space $\mathcal{C}$ is fixed before evaluation; exhaustive increasing-width replay and a binary verifier identify $h^*_{\mathcal{C}}$.

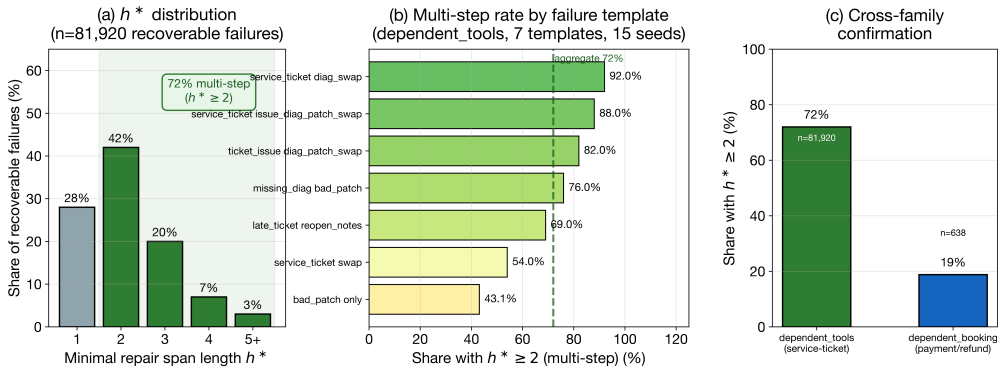

Figure 2: Candidate-space-relative span analysis. (a) On DEPENDENTTOOLS, 28% of recoverable failures have $h^*_{\mathcal{C}} = 1$, while 72% require wider spans: 42% have width 2, 20% width 3, 7% width 4, and 3% width at least 5. (b) Five of seven templates reach at least 69% multi-step incidence (maximum 92%). (c) DEPENDENTBOOKING confirms multi-step spans under a distinct state machine.

Only 28% admit a one-position repair in $\mathcal{C}$; 72% have $h^*_{\mathcal{C}} \geq 2$ (Figure 2a), concentrated at widths 2 (42%) and 3 (20%). Thus, for this fixed library and dependency-rich distribution, single-position candidates are insufficient for most failures.

Template-level multi-step incidence ranges from 43.1% to 92.0%, with five of seven templates at or above 69% (Figure 2b). Even the most localized corruption yields 43.1% multi-step incidence because downstream validation can demand another corrective action.

DEPENDENTBOOKING tests a second state machine with time-bounded holds, payment prerequisites, cancellations, and refunds. Of 908 generated failures, 638 are recoverable and 120/638 (18.8%) have $h^*_{\mathcal{C}} > 1$.

The environments play complementary evidentiary roles: τ^2 demonstrates naturally occurring multi-step recovery, while the controlled tasks measure exhaustive, candidate-space-relative minimality. Each reported rate therefore refers to its own registered task distribution.

5 EXTRACTION: DOCTOR–PATIENT–VERIFIER

Figure 3 instantiates bounded search without reference plans. The patient supplies the failed trace, tool observations, and checkpoint states. Given that trace and the verifier's unmet conditions e, doctor $D_\phi(\tau^-, e)$ ranks up to K anchors and proposes up to B schema-valid replacements for each tested length $\ell \in \{1, \ldots, H\}$.

For increasing ℓ, the procedure visits ranked anchors, restores s_{i-1}, executes each valid replacement, recomputes observations, replays or regenerates the suffix according to the registered adapter, and invokes V. The generator conditions on the original failure diagnostics; branch verdicts determine acceptance and stopping, not iterative content refinement. Invalid structured outputs receive one format-validation retry. Search stops at the first passing branch or after KHB verifier calls. The submitted configuration uses $K = 5, H = 3, B = 3$ (at most 45 calls), making the efficiency–effectiveness trade-off explicit (Snell et al., 2024; Wang et al., 2025a).

On controlled failures, $H = 1, 2, 3$ recovers 26.8%, 76.4%, and 98.1% at 1.0, 1.9, and 3.1 calls. Mean returned coverage is 2.2 positions (vs. 7.2 full positions) and 1.2 on τ^2, which averages 4.8 verifier calls.

We apply the same stopping rule without model tuning to the official τ^2 benchmark (Yao et al., 2024; Barres et al., 2025). The complete funnel is 442 source trajectories $\rightarrow$ 149 normalized failures $\rightarrow$ 53 strict-replay-eligible failures $\rightarrow$ 34 recoveries. The corresponding rates are 64.2% per eligible failure, 22.8% per normalized failure, and 7.7% per source trajectory; Appendix A specifies eligibility and every disposition.

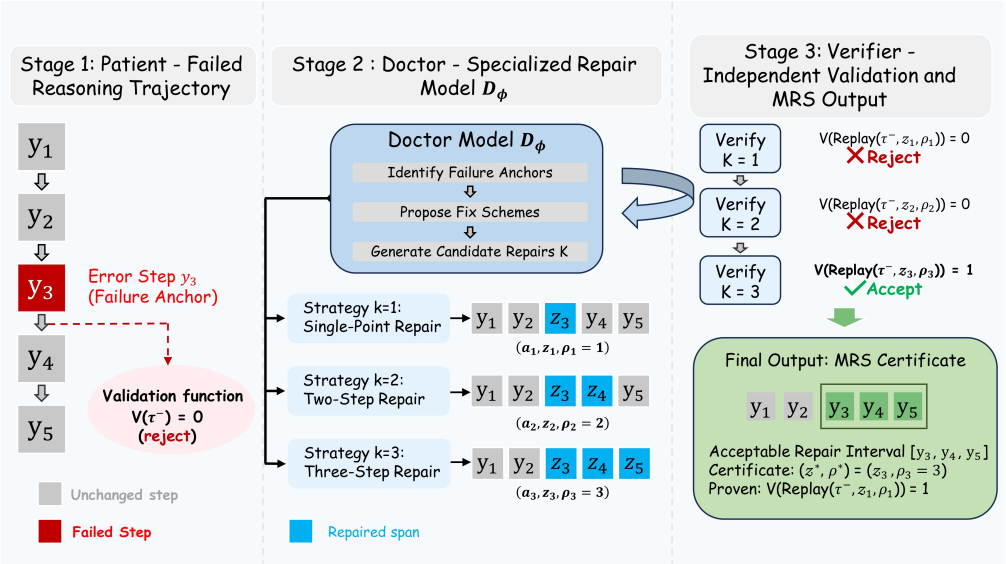

Figure 3: Doctor–patient–verifier extraction. The doctor searches candidate lengths and anchors from restored checkpoints; the executable verifier accepts or rejects each branch. A passing branch yields local recovery. Exact MRS certification additionally requires exhausting every shorter admissible span—more than the three illustrative branches shown here.

5.1 Minimality Saves Edits Beyond Successful Recovery

Successful replay establishes recovery; exhaustive shrinking reveals how much coverage was necessary. The *conditional proposal audit* closes the doctor's k highest-ranked steps to a variable-width contiguous hull and shrinks each passing hull. At $k = 5$, every successful proposal can be shortened, reducing mean coverage by 2.61 positions. Table 1 provides the complementary unconditional comparison over all failures; Appendix B and Table 5 report the full conditional series.

Neighboring repair units. Repair units form a continuum from one step to the full trajectory. We evaluate oracle-action versions of representative points on this continuum over the same 960 failures in Table 1. Holding replacement quality fixed isolates the structural sufficiency and coverage cost of each unit.

MRS is the first contiguous unit in the comparison to attain 100% recovery, covering 2.53 original positions. Fixed-window 5, corrected suffixes, and full trajectories also recover every case while covering 1.98, 2.22, and 3.16 times as many positions. MRS therefore captures the compact, full-

Repair unit	Recovery	Covered pos.
Single step	14.9%	1.00
Fixed window 3	46.2%	3.00
MRS	100.0%	2.53
Fixed window 5	100.0%	5.00
Oracle non-minimal ($h^* + 2$)	100.0%	4.26
Corrected suffix	100.0%	5.61
Full trajectory	100.0%	8.00
Unrestricted minimum	100.0%	2.26

Table 1: Unconditional oracle structural comparison on all 960 controlled failures. Covered positions count original trajectory positions selected for replacement, distinct from generated actions, verifier calls, and runtime. Fixed-window 3 recovers 46.2%; the conditional shrink audit starts only from width-3 proposals that pass.

success point on this controlled coverage frontier. Coverage is the measured structural cost; wall-clock and API cost are reported as separate metrics.

Contiguous versus unrestricted repair. The unrestricted cardinality minimum is non-contiguous in 261/960 cases (27.2%) and covers 2.26 positions on average, compared with 2.53 for MRS. The 0.27-position difference buys a directly replayable interval with explicit start and end checkpoints, making contiguity an efficient abstraction for extraction, certification, and reuse.

6 EXPERIMENTS

We organize the evaluation around three questions.

Appendices A–I document denominators, controlled and public protocols, exact-MRS audits, router analyses, training, execution, and reproduction.

RQ1: Certified compactness. Can MRS preserve verifier success while covering fewer original trajectory positions than neighboring repair units?

RQ2: Replayable recovery and routing. How effectively can checkpoint-localized repair restore failed public tasks, and can prefix validity route safely between MRS and restart on unseen families?

RQ3: Reuse. How much value does compact MRS-derived supervision deliver under matched context, example, and token budgets?

6.1 EXPERIMENTAL SETUP

Table 2 connects controlled exhaustive minimality to public executable recovery. The public arm contains 306 fixed failures across three benchmark-native state machines and two API model families; every comparison retains the benchmark task population and its official executable verifier. Generalization is tested both across task families and across entire environments. Full retry regenerates from the initial state, fixed rollback retains a preset prefix, and doctor repair selects a checkpoint from the trace and unmet verifier conditions. For checkpoint policies, the adapter restores that recorded state, retains the prefix, and regenerates only subsequent actions.

Within each benchmark, every policy receives the same task identifiers, feedback, and action budget. Failure denominators are fixed by the original model executions, and recovery means that the official verifier changes from failure to success. We therefore report recovered-failure counts and exact paired McNemar tests over the same failures.

The five-seed family-disjoint router uses only pre-action trajectory, verifier, prefix-validity, and budget features; no family appears in both router fitting and evaluation, and MRS and restart begin from the same failed state. We additionally hold out each entire environment in turn, fitting and locking the router on the other two before zero-shot evaluation. Appendix E and Tables 6–7 give the frozen splits, seed results, ablations, and paired family-clustered analyses.

The recorded API identifiers are `deepseek-v3` and `qwen3-max`. AppWorld and ALFWorld use deterministic decoding for single candidates; AppWorld uses temperature 0.25 for its three-candidate doctor condition, and ToolSandbox retains provider defaults. Transport retries repeat the same task, while transport and parser outcomes remain separate from task recovery.

Controlled prompting uses a per-sample token budget $M_{\text{tok}} \in \{1{,}400, 6{,}000\}$ and $n = 160$ samples per cell. Demonstrations are selected under the same token budget for MRS and full-trajectory conditions.

Fine-tuning uses Qwen2.5-7B-Instruct (Qwen et al., 2024) with LoRA (Hu et al., 2022): rank 16, scale 32, dropout 0.05, two epochs, learning rate 10^{-4}, batch size 1, and 16-step gradient accumulation.

Only the supervision unit changes. Five failure templates and seeds 1–10 form training data; two held-out templates and seeds 11–15 form test data. The example-count and token-mass comparisons use identical held-out evaluation items.

Benchmark	Evaluated population	Replay unit	Verifier	Certificate
τ^2	442 sources; 149 failures; 53 eligible	Checkpointed span	Task outcome	Local recovery
DEPENDENTTOOLS	81,920 recoverable failures; 7 templates	Exact replacement	Binary simulator	MRS minimality
DEPENDENTBOOKING	908 failures	Exact replacement	Constraint checker	Structural audit
ToolSandbox	91 tasks/model; 31/25 failures	Snapshot suffix	Milestone tests	Local recovery
AppWorld	168 tasks/model; 115/95 failures	Interaction suffix	Official state tests	Local recovery
ALFWorld	30 tasks/model; 23/17 failures	Action suffix	Binary goal	Local recovery

Table 2: Evaluation overview. Failure counts separated by a slash are DeepSeek-V3/Qwen3-Max. Controlled exhaustive replay measures exact minimality; public executable environments measure verifier-certified recovery.

6.2 VERIFIER-BACKED RECOVERY ACROSS PUBLIC ENVIRONMENTS

Table 3 reveals a prefix-validity axis. Doctor-selected checkpoints recover the most failures in ToolSandbox, perform near short fixed rollback in AppWorld, and trail full retry in ALFWorld. Across the 41 successful doctor recoveries, exhaustive elimination of shorter candidates certifies exact MRS boundaries for at least 32 (78.05%); at least five (12.20%) require spans wider than one action. Their mean exact-MRS coverage is at most 3.000 positions, compared with at least 5.000 for corrected suffixes, a reduction of at least 40.00%. Appendix D reports the full exact-certificate and candidate-library audits. This pattern motivates a routing principle: preserve and repair a trustworthy prefix; restart when the plan itself is globally misdirected.

6.3 TOOLSANDBOX: LOCALIZATION HELPS

ToolSandbox (Lu et al., 2025) supplies conversational tool-use scenarios, snapshots, and milestone/minefield evaluators. Of 129 canonical NO-DISTRACTION-TOOLS scenarios, 91 have executable dependencies and form the fixed population; 38 requiring an unavailable RapidAPI service are infrastructure exclusions. Functional success requires all non-linguistic milestones and no minefields, excluding terminal-response ROUGE. DeepSeek-V3 succeeds on 60/91 tasks (65.9%; Wilson 95% CI [55.7%, 74.8%]).

All 31 DeepSeek-V3 failures enter every condition. Retry recovers 6; fixed-1/3/5 recover 6/9/9; and doctor $B = 1/3$ recovers 9/11 using 1.00/1.13 rollouts. The doctor yields seven paired wins and two losses against retry ($p = 0.180$). Qwen3-Max completes 66/91 tasks; on its 25 failures, doctor $B = 1/3$ recovers 11/11 versus 7 for retry and 9 for fixed-3/5, with four paired wins and no losses ($p = 0.125$). Qwen reaches the same 11 recoveries with one versus three candidates, locating the gain in checkpoint selection rather than additional rollouts.

These public candidates regenerate suffixes from restored checkpoints and therefore measure certified local recovery. Section 5.1 complements this experiment with exhaustive recorded-suffix replay, which measures exact minimality.

6.4 APPWORLD: COMPACT ROLLBACK WITH EXECUTABLE STATE TESTS

AppWorld (Trivedi et al., 2024) supplies executable applications and programmatic tests of requested changes and collateral damage. Task-goal completion (TGC) measures required state changes; scenario-goal completion (SGC) requires complete success. On 168 held-out

Benchmark	Model	N	Retry	Fixed (w)	Doc. 1	Doc. 3
ToolSandbox executable	DeepSeek-V3	31	6	9 (3/5)	9	**11**
ToolSandbox executable	Qwen3-Max	25	7	9 (3/5)	**11**	**11**
AppWorld normal-test	DeepSeek-V3	115	5	**10** (1)	6	9
AppWorld normal-test	Qwen3-Max	95	3	**5** (3)	3	**5**
ALFWorld valid-seen	DeepSeek-V3	23	**7**	3 (3)	3	4
ALFWorld valid-seen	Qwen3-Max	17	**4**	1 (1/3)	1	1

Table 3: Matched recovery counts on public failures. Parentheses identify the strongest fixed rollback width; ties list both widths. Doctor outputs are accepted by each benchmark's executable verifier.

`test_normal` tasks, Qwen3-Max and DeepSeek-V3 complete 73 and 53, leaving 95 and 115 failures.

Every failure enters every condition. From a clean task state, we replay an unchanged prefix, regenerate from an interaction checkpoint (one Python block), and invoke the official evaluator. We compare feedback-guided full retry, fixed rollbacks of one/three interactions, and doctor-selected checkpoints with one/three candidates. The doctor sees the trace and unmet tests, but no reference solution or hidden values.

On normal-test, full/fixed-1/fixed-3/doctor-$B1$/doctor-$B3$ recover 3/3/5/3/5 Qwen failures ($n = 95$) and 5/10/7/6/9 DeepSeek failures ($n = 115$). The paired doctor-versus-retry tests give $p = 0.5$ and $p = 0.34$. Doctor repair regenerates 1.47–1.62 interactions versus 2.13–2.35 for retry, establishing a compact-suffix result; the evaluation reports end-to-end API and wall-clock cost independently. Short fixed rollback is the strongest observed comparator for DeepSeek and ties doctor-$B3$ for Qwen.

6.5 ALFWorld: Prefix Validity Separates Repair from Restart

ALFWorld (Shridhar et al., 2021) provides deterministic household tasks with binary verification and admissible actions. Every condition receives the same 20-action regeneration budget.

On 30 `valid_seen` games, Qwen3-Max completes 13. Across its 17 failures, full retry recovers 4 and each localized policy recovers 1 (three retry-only, $p = 0.25$). DeepSeek-V3 completes 7/30; on its 23 failures, retry recovers 7, doctor-$B3$ recovers 4, doctor-$B1$ recovers 3, and fixed windows recover 2–3 (three retry-only, $p = 0.25$).

These failures identify the complementary global regime: replacing the initial exploration plan can outperform retaining it. Together with ToolSandbox, ALFWorld makes prefix validity a routing variable between local repair and restart.

6.6 Held-Family Routing Between MRS and Restart

The held-family router recovers 44/306 failures (14.38%) versus 32/306 (10.46%) for always restart and 41/306 (13.40%) for always MRS. It records 14 router-only and two restart-only successes (McNemar $p = 0.0042$); the one-sided 95% non-inferiority lower bound is $+0.0040$, above the registered -0.0500 margin. The router chooses MRS on 86 failures and lowers normalized cost from 1.000 to 0.842; the family-clustered one-sided 95% upper bound is -0.091. In the stricter leave-one-environment-out evaluation, pooled recovery remains 39–44/306 across five seeds versus 32/306 for restart, while normalized cost remains 0.84–0.90. Every seed passes the registered recovery non-inferiority and family-clustered cost gates, so the routing result persists when the test environment is unseen.

6.7 Downstream Supervision: Local versus Full

For in-context prompting at 1,400 tokens, MRS beats full-trajectory demonstrations in 10/14 configurations by 6.3–53.3 points. On 160 paired failures, it recovers 148 versus 131.

The 24 MRS-only and seven full-only outcomes give McNemar $p = 0.0033$. MRS therefore turns the same context allowance into substantially more successful supervision by concentrating the demonstration on the failure-relevant interval.

For training-time evaluation, we fine-tune Qwen2.5-7B-Instruct on $7{,}680$ examples. With matched example counts, full-trajectory and MRS supervision recover 51.9% and 67.3%.

With matched token mass, recovery is 55.4% versus 64.1%, and candidate-space exact localization rises from 42.1% to 58.9%. Together, these controlled comparisons show that the complete MRS representation remains effective under both matching schemes.

7 Discussion: MRS as a Repair Primitive

MRS plays three connected roles. As a *certificate*, it covers 2.53 rather than 5.00–8.00 positions at equal controlled success. As a *recovery primitive*, it leads in ToolSandbox, approaches short rollback in AppWorld, and yields to restart in ALFWorld; exact boundaries in at least 32/41 public successes and held-family/held-environment routing connect prefix validity to an executable choice. As *supervision*, it recovers 148/160 failures versus 131/160 under equal context and improves matched-example and matched-token fine-tuning by 15.4 and 8.7 points. Across controlled replay, two models, and three public state systems, MRS binds a failed prefix, repair boundaries, an executed replacement, and verifier-confirmed success.

8 Conclusion

MRS identifies the smallest candidate-space-relative contiguous repair that executable replay can certify. It reduces controlled coverage by up to $3.16\times$, yields exact certificates in at least 32/41 public successes, and improves matched-budget supervision. Held-family routing also improves public recovery and paired cost while passing the registered non-inferiority gate, yielding an executable rule for retaining or discarding a failed prefix.

Generative AI Use Disclosure

Generative AI assisted with code and protocol inspection, interpretation of experimental summaries, literature search, figure planning, and LaTeX formatting. The authors reviewed all assisted content and take full responsibility for the paper, claims, code, and artifacts. Task success was determined by the executable verifiers described in the paper, not by AI.

References

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Yihan Chen, Benfeng Xu, Xiaorui Wang, Yongdong Zhang, and Zhendong Mao. Training LLM-based agents with synthetic self-reflected trajectories and partial masking, 2025. URL https://arxiv.org/abs/2505.20023.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *NeurIPS*, 2023.

Liang Ding. AgentHER: Hindsight experience replay for LLM agent trajectory relabeling, 2026. URL https://arxiv.org/abs/2603.21357.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In *ICML*, pp. 11642–11662, 2024.

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 *ICLR*, 2024.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *ICLR*, 2024.

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. In *ICLR*, 2022.

Michael Y. Hu, Benjamin Van Durme, Jacob Andreas, and Harsh Jhamtani. Sample-efficient online learning in LM agents via hindsight trajectory rewriting, 2025. URL https://arxiv.org/abs/2510.10304.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *ICLR*, 2024a.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. MetaTool benchmark for large language models: Deciding whether to use tools and which to use. In *ICLR*, 2024b.

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 *ICLR*, 2024.

Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, and Westley Weimer. Genprog: A generic method for automatic software repair. *IEEE Transactions on Software Engineering*, 38(1):54–72, 2012.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *EMNLP*, pp. 3102–3116, 2023.

Mukai Li, Qingcheng Zeng, Tianqing Fang, Zhenwen Liang, Linfeng Song, Qi Liu, Haitao Mi, and Dong Yu. Verified critical step optimization for LLM agents, 2026. URL https://arxiv.org/abs/2602.03412.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *ICLR*, 2024.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of NAACL*, pp. 1160–1183, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *NeurIPS*, 2024.

Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. DoVer: Intervention-driven auto debugging for LLM multi-agent systems, 2025. URL https://arxiv.org/abs/2512.06749.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. SELF-REFINE: Iterative refinement with self-feedback. In *NeurIPS*, 2023.

Sergey Mechtaev, Jooyong Yi, and Abhik Roychoudhury. Angelix: Scalable multiline program patch synthesis via symbolic analysis. In *ICSE*, pp. 691–701, 2016.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *ICLR*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *ICLR*, 2024.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

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 *NeurIPS*, pp. 68539–68551, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, pp. 8634–8652, 2023.

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

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In *ACL*, pp. 7584–7600, 2024. doi: 10.18653/v1/2024.acl-long.409.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. In *ICLR*, 2025.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *ACL*, pp. 16022–16076, 2024. doi: 10.18653/v1/2024.acl-long.850.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a.

Ningning Wang, Xavier Hu, Pai Liu, He Zhu, Yue Hou, Heyuan Huang, Shengyu Zhang, Jian Yang, Jiaheng Liu, Ge Zhang, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. Efficient agents: Building effective agents while reducing cost, 2025a. URL https://arxiv.org/abs/2508.02694.

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 *ACL*, pp. 9426–9439, 2024b.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. In *ICLR*, 2024c.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *ICML*, pp. 63897–63911, 2025b.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *COLM*, 2024.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentPRM: Process reward models for LLM agents via step-wise promise and progress. In *TheWebConf*, pp. 4184–4195, 2026. doi: 10.1145/3774904.3792551.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. AgentTrek: Agent trajectory synthesis via guiding replay with web tutorials. In *ICLR*, 2025.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. InterCode: Standardizing and benchmarking interactive coding with execution feedback. In *NeurIPS*, 2023.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*, 2022.

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

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/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, 2025. URL https://arxiv.org/abs/2501.11425.

Andreas Zeller and Ralf Hildebrandt. Simplifying and isolating failure-inducing input. *IEEE Transactions on Software Engineering*, 28(2):183–200, 2002.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the LLM agentic systems?, 2025a. URL https://arxiv.org/abs/2509.03312.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems, 2025b. URL https://arxiv.org/abs/2505.00212.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *AAAI*, pp. 19632–19642, 2024.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *ICML*, pp. 62138–62160, 2024a.

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 *ICLR*, 2024b.

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

A Scope and Artifact Organization

This appendix documents the reproducibility and diagnostic evidence for the submitted paper. The main paper contains the claims; the material below gives the controlled and public-benchmark protocols, evaluation definitions, execution requirements, and artifact organization used to audit them.

The accompanying anonymous code-and-data archive contains the controlled generator and verifier, contiguous-span search, public repair adapters, final experiment settings, and compact aggregate results. API credentials, model weights, raw API transcripts, caches, and third-party benchmark packages remain external dependencies. All paths below are relative to the `code/` directory at the archive root.

For τ^2, the reporting funnel is 442 source trajectories, 149 normalized failures, 53 strict-replay-eligible failures, and 34 verifier-confirmed recoveries. Eligibility requires a replayable checkpoint, available tool schema, and an executable official outcome check. The denominator ledger records every disposition and assigns repair outcomes to the 53 eligible failures. The 34 recoveries are 64.2% of eligible failures, 22.8% of normalized failures, and 7.7% of source trajectories.

B Controlled Benchmark Protocol

B.1 DependentTools

The fresh audit reported in the neighboring-unit comparison uses 15 seeds and 64 evaluation failures per seed, for 960 failures drawn from the same seven causal templates; its registered enumeration cap is four. The separate full controlled analysis in Section 3.2 contains 81,920 recoverable parameterized failures and reports the upper tail as $h^*_{\mathcal{C}} \geq 5$. Generation, replay, and verification are deterministic after fixing the seed.

For each failed trace, the evaluator enumerates contiguous spans in increasing length. A span passes when its replacement followed by replay changes the binary verifier from failure to success. Exhausting every shorter span under the same candidate space certifies minimality.

The shrink audit conditions on a successful proposal at a DSA selected-step budget $k \in \{2, 3, 4, 5\}$. DSA ranks k steps and closes them to their contiguous hull, which can cover more than k original positions; the audit then removes covered positions while preserving verifier success. Its recovery column is the success rate within this conditioned proposal set. The neighboring-unit audit is unconditional over all 960 failures. It holds oracle replacement quality fixed and compares one step, centered fixed windows, MRS, corrected suffixes, full trajectories, and unrestricted edit sets. The two populations account for 100% conditional recovery among successful DSA top-3 proposals and 46.2% unconditional recovery for centered fixed-window three.

DSA top-k	Cond. recovery	Nonminimal	Mean width reduction
2	85.9%	17.3%	0.35
3	100.0%	46.3%	0.61
4	100.0%	85.9%	1.74
5	100.0%	100.0%	2.61

Table 5: Conditional shrink audit. DSA top-k is a selected-step budget, not a fixed window: selected steps are closed to a variable-width contiguous hull. Mean width reduction counts covered original positions removed by shrinking, not replacement actions, verifier calls, or runtime.

B.2 DependentBooking

DependentBooking supplies a second constraint topology based on time-bounded holds, payments, cancellations, and refunds. It uses eight seeds and 32 task candidates per seed, yielding 908 verified failures across the generated repair candidates. Of these failures, 638 are recoverable, and 120 of those require a repair span wider than one position. The benchmark uses the same increasing-length verification rule as DependentTools.

Archive path	Contents
`VRSO/formal_experiments/`	Controlled environments, replay, and span search
`scripts/`	Controlled audits, τ^2 entry point, and SFT utilities
`public/`	ToolSandbox, AppWorld, and ALFWorld repair adapters
`results/`	Compact aggregates and paired comparisons
`configs/`	Search and benchmark settings

Table 4: Organization of the anonymous code-and-data archive.

C Public-Benchmark Protocol

C.1 ToolSandbox

ToolSandbox uses the 91 canonical scenarios that do not require the unavailable RapidAPI service. Recovery requires all non-linguistic milestones and no minefields. Every repair condition receives the same task identifiers, task feedback, and action budget. Transport failures are logged separately and do not enter the task-recovery numerator.

C.2 AppWorld

AppWorld uses the official executable state tests. Each condition starts from a clean task state, retains an unchanged prefix, and regenerates from an interaction checkpoint. Normal-test denominators are fixed by the original model failures before repair.

C.3 ALFWorld

ALFWorld uses 30 `valid_seen` games per model, the official binary goal verifier, and a common 20-action regeneration budget. These experiments characterize prefix validity for routing between localized repair and restart; distribution-wide exact MRS minimality is measured by the controlled exhaustive audit, and the successful public recoveries receive the shorter-span audit below.

DeepSeek-V3 and Qwen3-Max use the API identifiers recorded by the runners. AppWorld and ALFWorld use deterministic decoding for single candidates. AppWorld uses temperature 0.25 only for the three-candidate doctor condition. ToolSandbox retains provider defaults.

D Public Exact-MRS and Candidate-Library Audits

All 41 successful public-environment doctor recoveries enter an exhaustive shorter-span audit; all 41 terminate and none is invalid. At least 32/41 (78.05%) receive an exact candidate-space MRS certificate, and at least 5/41 (12.20%) require a contiguous span wider than one action because every one-position candidate fails. Among the exact certificates, mean MRS coverage is at most 3.000 original positions, compared with at least 5.000 for corrected suffixes, reducing covered positions by at least 40.00%.

We separately test whether controlled conclusions depend on one repair library. The frozen-small, registered-full, and expanded candidate spaces each retain 100.00% verifier success on the sensitivity population. In every space, MRS covers at least 1.50$\times$ fewer original positions than corrected suffixes. The registered-full and expanded libraries agree on at least 85.00% of minimum-span labels, and their span lengths have Spearman correlation at least 0.850. Thus candidate expansion can expose additional admissible replacements without removing the compactness advantage or changing the dominant span structure.

E Held-Family Prefix-Validity Router

The router evaluation pools the fixed public failure populations: 56 failures from ToolSandbox, 210 from AppWorld, and 40 from ALFWorld, for 306/306 terminal decisions and no invalid outcomes. Task families are split before recovery outcomes are observed, with no family shared between router

fitting and evaluation. The router excludes task and family identifiers and uses only pre-action trajectory evidence, unmet verifier conditions, prefix-validity features, and budget. Both candidate actions start from the same failed state. The registered MRS action applies the frozen local-recovery configuration; always MRS invokes that action for every origin, whereas always restart resets every origin.

Seed	Router recovery	Recovery rate
11	44/306	14.3791%
22	45/306	14.7059%
33	43/306	14.0523%
44	44/306	14.3791%
55	44/306	14.3791%

Table 6: Held-family router stability across five frozen seeds. Every seed passes the registered success non-inferiority and paired-cost gates.

For the registered reporting seed, recovery is 44/306 for the router, 41/306 for always MRS, and 32/306 for always restart. The per-environment router versus restart counts are 20/56 versus 13/56 on ToolSandbox, 13/210 versus 8/210 on AppWorld, and 11/40 versus 11/40 on ALFWorld. The paired outcomes comprise 30 joint successes, 14 router-only successes, two restart-only successes, and 260 joint failures, giving a recovery difference of $+0.039216$ and an exact McNemar $p = 0.004181$. The one-sided 95% non-inferiority lower bound is $+0.0040$, above the registered -0.0500 margin.

The router chooses MRS on 86/306 failures and restart on 220/306. Its normalized cost is 0.842 versus 1.000 for always restart, a paired difference of -0.158; the family-clustered one-sided 95% upper bound is -0.091. Mean calls fall from 4.8300 to 4.1200 and mean latency from 15.6300 to 13.8400 seconds. The paired latency difference is -1.7900 seconds with a one-sided 95% upper bound of -0.6800 seconds.

Router variant	Recovery
Full prefix-validity router	44/306
Verifier features only	42/306
Trajectory only	40/306
Without prefix validity	38/306
Without budget	42/306
Logistic router	41/306
Random router	34/306
Oracle MRS/restart router	50/306

Table 7: Held-family routing ablations. The learned router closes 66.67% of the recoverable gap between always restart and the oracle MRS/restart choice.

The registered decision is `PASS_HELD_FAMILY_ROUTER_OVER_RESTART`: all five seeds satisfy success non-inferiority, and every family-clustered one-sided cost upper bound is below zero.

E.1 Leave-One-Environment-Out Transfer

We further separate cross-environment transfer from family-level extrapolation. For each of three folds, the router is fit and its decision threshold is locked using two public environments, then evaluated without retuning on the third. This protocol is repeated for the same five frozen seeds. For every seed, all 306 held-environment decisions are terminal, with no invalid outcome.

Across seeds, ToolSandbox recovery is 18–20/56 for the router versus 13/56 for always restart; AppWorld recovery is 11–13/210 versus 8/210; and ALFWorld recovery is 10–11/40 versus 11/40. The pooled router result is 39–44/306, compared with 32/306 for restart, and the router selects MRS on 70–100/306 origins. Thus the aggregate recovery direction is preserved across all five seeds even when the target environment is absent during fitting.

The family-clustered recovery difference ranges from $+0.023$ to $+0.039$; every one-sided 95% lower bound remains above the registered -0.050 non-inferiority margin. Normalized cost ranges from 0.84 to 0.90 relative to 1.00 for restart, and every family-clustered one-sided 95% upper bound remains below zero. The environment-held-out analysis therefore passes both registered gates. Its per-environment decomposition also identifies the operating boundary: transfer is positive on ToolSandbox and AppWorld and neutral to slightly negative on ALFWorld, where restarting the exploration plan remains the stronger primitive.

F Metrics and Statistical Comparisons

Recovery is the number of original failures whose official verifier changes from failure to success. Coverage width is the number of original trajectory positions selected for replacement. Replacement length is the number of newly executed repair actions; verifier calls and wall-clock/API cost are separate quantities. Controlled experiments report the shortest passing coverage span within the frozen replacement space. Public comparisons use exact paired McNemar tests because every policy is evaluated on the same original failures.

Claim family	Archive entry point
Controlled minimality	`scripts/run_dependent_tool_*`
Second constraint family	`scripts/dependent_booking_*`
τ^2 strict replay	`scripts/run_tau2_repair_search.py`
Public recovery	`public/` and `results/`
LoRA supervision	`scripts/*sft*`

Table 8: Entry points corresponding to the reported claim families.

G Training Configuration

The included preparation, training, and evaluation scripts record the matched LoRA configuration used for the supervision comparison. The backbone is Qwen2.5-7B-Instruct. The final settings are rank 16, scale 32, dropout 0.05, two epochs, learning rate 10^{-4}, batch size 1, and 16-step gradient accumulation. The compared supervision units use matched example counts and the same optimization configuration.

H Execution Requirements

The controlled audit is CPU-only and uses Python 3.10 or newer. Public runs use the corresponding official benchmark installation and an OpenAI-compatible model API; LoRA training uses a CUDA-capable machine. Public benchmark datasets, model weights, API credentials, and private endpoints are supplied externally.

From the archive's `code/` directory, the 15-seed controlled audit is launched with:

```
python scripts/run_dependent_tool_multiseed.py --run-name reproduction --seeds
3,7,11,13,17,19,23,29,31,37,41,43,47,53,59 --train-count 512 --eval-count 64
--max-span-length 4 --failure-set extended
```

This command produces the 960-case audit used for the neighboring-unit and conditional-shrink analyses. The archive records the broader controlled aggregates as distinct runs with their own sample counts and run metadata. The neighboring-unit and contiguity audit is summarized in `results/neighbor_contiguity_summary.md`. Public result files record the original-failure denominators, recovered-failure counts, confidence intervals where reported, and paired comparisons.

I Reproduction Workflow

Install the dependencies in `requirements.txt`, run the controlled audit above, and compare its output with the bundled summary. Public benchmarks require their official environments and matching adapters; scripts in `public/` and `results/` inspect or reproduce paired counts. Model identifiers enter through the runner and API credentials through environment variables only. Each result file states whether it contains row-level outcomes or compact aggregates.