BEYOND REPLAY FIDELITY: RESTORATION COMPLETENESS IN AGENT EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Resuming a tool-using agent requires recovering the state on which its next actions depend. Faithfully restoring a record of success does not ensure that the recorded application effect persists. We study restoration completeness as a condition of agent evaluation, using a decision-contract framework that distinguishes local information sufficiency, native-score sensitivity, and adaptive recovery effort. Enumerating all 19 AppWorld development families (57 instances) yields 28 qualified boundaries. A controlled intervention retains the visible checkpoint while omitting application effects. Native scores fall in nine families (25 instances), and replaying the omitted statement rescues all 25. A 270-episode study evaluates GPT-5.6-sol, GPT-5.5 and Claude Opus 5 separately, each with five pairs per instance on the same nine selected affected instances. Complete restoration succeeds in 45/45 episodes per system, whereas Partial succeeds in 40/45, 38/45 and 40/45, respectively. Mean paired increases are 1.82, 0.87 and 1.64 Python execution requests (conditional 95% bootstrap intervals [1.42, 2.24], [0.20, 1.53] and [1.16, 2.18]), or 9.1%, 4.3% and 8.2% of the 20-request budget, computed before rounding. All three systems register completion in all five of their Partial reminder failures. Complementary Cybench witnesses test strict action-ranking reversals. Average added effort recurs across systems, while its magnitude and family-level signs vary. Evaluations should declare the restoration boundary and report native reward together with recovery effort under the stated interface and budget.

1 INTRODUCTION

Resuming a tool-using agent requires recovering both its recorded progress and the application state needed to continue. Interactive benchmarks distribute task state across model history, local execution, and external applications (Zhou et al., 2024; Xie et al., 2024; Yao et al., 2024; Trivedi et al., 2024). Recovery systems increasingly align agent and environment checkpoints or reconstruct a continuation from partial history (Zhuang et al., 2026; Wu et al., 2026; Gulati, 2026; Tan et al., 2025). Reproducing the declared saved state establishes replay fidelity. Whether that state preserves the continuation's dependencies, which may extend beyond the saved boundary, is an additional question.

In one AppWorld continuation, the restored interpreter retains a newly created playlist's identifier, but partial restoration removes the playlist from the application database. The reference continuation fails. Replaying the creation statement restores native success. Recovering the identifier alone does not recover the object needed to use it.

This mismatch matters when a recovered score is interpreted as evidence of agent capability, because terminal success can conceal it. An agent may inspect the application, recreate a missing object, and succeed, spending part of its interaction budget on reconstruction. It may instead trust remembered state and lose native score. Success alone does not distinguish preserved progress from progress reconstructed during continuation.

We study whether a restored checkpoint preserves the continuation's dependencies and how their omission changes native reward and observed recovery effort. This restoration completeness is relative to a continuation and payoff, not necessarily the entire machine. A decision contract declares the restored boundary, permitted observations, budget, and behavior when relevant state is unknown. We test this distinction through controlled application-state interventions and repeated continuations under three evaluated systems. Figure 1 contrasts matched visible checkpoints with

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.

different application states above three parallel tests of fixed-suffix scores (A), reference repair (B), and adaptive reward–effort outcomes (C).

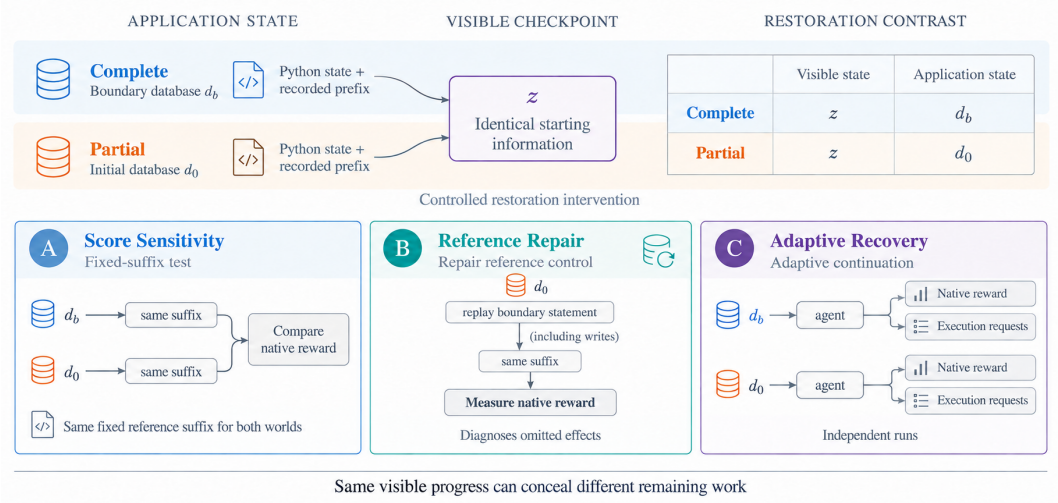

Figure 1: Restoration contrast and three parallel AppWorld tests. Top: Complete loads d_b (boundary database, blue); Partial loads d_0 (initial database, orange). Both retain z: Python state and recorded prefix. (A) Score Sensitivity compares native reward under the same fixed suffix. (B) Reference Repair replays the boundary statement, including writes, from d_0 before that suffix: a reference-assisted diagnosis, not an agent observation or decision-sufficiency certificate. (C) Adaptive Recovery measures native reward and execution requests in independent adaptive runs; requests can batch API operations. Arrows remain within tests. The schematic specifies comparisons, not measured outcomes.

The controlled census attributes score loss to omitted application effects in nine of 19 AppWorld development families (25 of 57 instances). Replaying the omitted statement rescues all 25. On nine selected affected instances, GPT-5.6-sol (Sol), GPT-5.5 and Claude Opus 5 each contribute 45 pairs under the same task-level inputs and budgets. All three incur additional requests on average and repeatedly register completion despite native failure in the reminder task. Magnitudes and family-level responses vary across systems, so the shared outcome pattern does not require identical recovery behavior.

The decision-contract formulation separates these empirical consequences from local information sufficiency. It states when a restored observation preserves a specified action comparison and how to refute that property with a paired counterexample. A Cybench witness supplies the complementary action-ranking test (Zhang et al., 2025). These contributions make restoration testable relative to the continuation it supports. Saved-state fidelity, decision information, and adaptive reward and effort each require their own evidence.

1.1 Related work

Recovery boundaries. Distributed snapshots and rollback recovery require consistency across saved components (Chandy & Lamport, 1985; Elnozahy et al., 2002). Checkpoint systems preserve process or virtual-machine execution (Ansel et al., 2009; Cully et al., 2008). AgentRewind aligns agent context with workspace-filesystem checkpoints, while DeltaBox accelerates sandbox rollback (Zhuang et al., 2026; Dong et al., 2026). Crab compares Chat-only, Chat+FS (history and filesystem without process memory), and Restart at random failure positions in 50 successful Terminal-Bench tasks, using replay and live continuation and reporting solve rate and runtime (Wu et al., 2026). ContinuityBench varies restored history while retaining live environments (Gulati, 2026). Recovery-Bench evaluates recovery from failed trajectories (Tan et al., 2025). Its public implementation replays failed commands in a fresh container (Letta, 2026). ACRFence addresses repeated irreversible effects after rollback (Zheng et al., 2026). We isolate omitted application effects under fixed code, outputs and interpreter variables, using unchanged suffixes and targeted repair.

Information and evaluation. Perceptual aliasing, POMDPs, and state abstraction already establish that equal observations can require different decisions (Whitehead & Ballard, 1991; Kaelbling et al., 1998; Li et al., 2006). Our local witness applies this principle at a declared restoration boundary. Information value depends on signals and loss (Blackwell, 1953); selective classification studies risk–coverage tradeoffs (Chow, 1970; El-Yaniv & Wiener, 2010). Accuracy–cost evaluation already motivates joint measurement for agents (Kapoor et al., 2025); our contribution is the controlled restoration contrast, not joint reporting itself. Benchmark audits emphasize intended capability, reproducibility, and transparent reporting (Reuel et al., 2024; Zhu et al., 2025). Our census measures finite intervention coverage. Adaptive traces characterize recovery on selected affected instances.

2 Decision Contracts for Restoration Completeness

2.1 State, information, and behavior

Restoration completeness requires preserving the decision information or continuation outcomes specified for an evaluation, rather than every bit of machine state. It is therefore relative to a contract that states what must be preserved. We distinguish two targets below, local action ranking and a policy's reward–effort distribution.

A decision contract declares the restored state, continuation payoff, allowed observations and their budget, and acceptable fallback behavior. Its state clause asks whether recovery reproduces what the checkpointer promises. Its information clause asks whether the visible state supports the specified decision, either immediately or after permitted observation. Its behavior clause asks whether the agent continues, inspects, repairs, or abstains through that interface. These clauses specify separate evaluation questions without assuming statistical independence.

The information clause also has a temporal boundary. A remembered successful update or validation response refers to a particular application state. Copying that response does not establish that its referent still exists. Evidence used after recovery must remain applicable to the dependency at action time. Appendix A.1 specifies the observation process, including possible state changes between checking and acting.

2.2 Local decision sufficiency

Let $w \in \mathcal{W}$ be a reachable continuation world, including external dependencies, and let $z = \rho(w)$ be the restored observation. For two fixed continuations $\mathcal{A} = \{a^+, a^-\}$, define $V(w, a)$ as expected terminal payoff under the protocol's shared remainder policy, and write

$$\Delta(w) = V(w, a^+) - V(w, a^-). \tag{1}$$

The observation preserves this comparison, including ties, exactly when its sign factors through the restored state:

$$\operatorname{sgn} \Delta(w) = g(\rho(w)) \quad \text{for every } w \in \mathcal{W}. \tag{2}$$

Equivalently, the sign is constant on each observation fiber $\mathcal{W}_z = \{w : \rho(w) = z\}$. This is a local decision-sufficiency condition, relative to the chosen continuations, payoff, and reachable worlds. It targets one comparison, rather than the richer equivalences studied in MDP abstraction (Givan et al., 2003; Li et al., 2006).

Definition 1 (OPD alias)**.** *A pair (w^+, w^-) is an observational partial-decision (OPD) alias if $\rho(w^+) = \rho(w^-)$ and $\Delta(w^+) > 0 > \Delta(w^-)$.*

An OPD pair refutes Equation 2, but not finding a pair does not prove it. In particular, a fiber containing a tie and a strict preference violates exact sign factorization without containing a strict reversal. A visible-only forced choice cannot select the better action in both worlds. On a two-world distribution with $p = \Pr(w^+ \mid z) \in (0, 1)$, its minimum unit wrong-sign risk is $\min(p, 1 - p)$, the classical binary Bayes boundary. This local restriction permits later observation, repair, or abstention and does not imply failure of an unrestricted adaptive policy.

2.3 Native reward and recovery effort

An outcome-relevant dependency need not create a two-action reversal. An omitted completed update can directly violate a task's final-state checks. For a fixed suffix σ, let $R_i^c(\sigma)$ and $R_i^p(\sigma)$ denote native

rewards on instance i under complete and partial restoration with the same visible checkpoint. Their difference measures fixed-suffix score sensitivity. Establishing an OPD alias additionally requires two fixed continuations with opposite rankings across the worlds. Our AppWorld census tests score sensitivity, while the Cybench witnesses test strict OPD. Preserving an action ranking alone also need not preserve its reward level.

For an adaptive policy π under a fixed interface and budget, record native reward $R_i^s(\pi)$ and execution effort $K_i^s(\pi)$ in restoration arm $s \in \{c, p\}$. The paired empirical contrasts are

$$\widehat{\Delta}_R^\pi = \frac{1}{n}\sum_{i=1}^{n}[R_i^c(\pi) - R_i^p(\pi)], \qquad \widehat{\Delta}_K^\pi = \frac{1}{n}\sum_{i=1}^{n}[K_i^p(\pi) - K_i^c(\pi)]. \tag{3}$$

With repetitions, we first average each bracket within an instance and then average across the fixed instances. Positive values indicate lost reward and added effort, respectively. Here K counts Python execution requests, each of which can contain multiple API calls. These operational measurements differ from a normalized theoretical probe cost. They describe the selected paired episodes and do not imply that every omission must cause loss or extra work.

The framework thus separates fidelity to saved state, preservation of a local decision, and preservation of a policy's reward and effort. Stronger conditions on the entire continuation process can ensure the last property. Appendix A.3 states and proves one sufficient condition. The experiments assess reward and effort directly, allowing successful adaptive repair to remain visible.

3 Boundary Intervention and Evaluation

Our main empirical test measures whether omitting application effects changes native completion outcomes under a reference suffix or an adaptive agent, and the work needed to obtain those outcomes. A complementary local OPD test asks whether an omitted dependency reverses a fixed action ranking.

3.1 A census of native application-state boundaries

AppWorld provides simulated applications, an interactive Python executor, reference solutions, and state-based task tests (Trivedi et al., 2024). We enumerate all 19 task families in its development split, comprising 57 instances grouped by their shared task identifier. Variants within a family are related instances, not independent mechanisms. The task list, boundary rule, restoration conditions, failure accounting, and score analysis were fixed before the formal runs.

For each task, we execute its official compiled reference solution one top-level Python statement at a time. We select the first statement before the final `complete_task` call whose saved application-database manifest differs from the initial manifest. This determines a prefix, a boundary statement, and a remaining reference suffix. Here the prefix includes the selected boundary statement, and the suffix begins with the next top-level statement. A statement can be a loop containing several API operations, so the boundary need not be a single atomic write. Tasks without such a boundary and qualification errors remain in the full denominator.

Matched restoration conditions. Let d_0 be the initial application database and d_b the database after the boundary statement. Each arm recreates the same reference prefix and retains its Python namespace. We then use AppWorld's native `load_state` operation. Complete loads d_b and executes the reference suffix. Partial loads d_0 and executes the same suffix while retaining the prefix's interpreter variables. Repair also loads d_0, but re-executes the selected boundary statement before running that suffix. Thus Complete and Partial differ in the loaded application state (Figure 1A), while Repair tests whether restoring the omitted effects recovers the reference outcome (Figure 1B). The comparison models a saved agent state paired with an earlier application snapshot. Both database states arise through normal task execution. The intervention uses an existing state-loading API. We deliberately choose the mismatched snapshots and do not attribute that choice to AppWorld's default evaluation workflow. Here "external" means outside the retained interpreter state, not outside every possible operating-system or virtual-machine snapshot.

The omission can be substantial. For seven of the nine instances later selected for adaptive continuation, the boundary statement immediately precedes the final completion call, so their reference

suffix only declares completion. The census therefore tests the loss of already achieved application effects under this boundary rule, not sensitivity to arbitrarily small omissions. The other two selected instances each retain one additional top-level statement before completion. We additionally test an earlier checkpoint after half the pre-completion top-level statements in these nine instances, keeping the original remaining suffix. This position is defined from source order, without inspecting the intervention reward. It is a statement-index midpoint, not a midpoint in elapsed time or in completed application updates.

Scoring and checks. Completed continuations are scored by AppWorld's unmodified evaluator. A suffix execution failure receives zero under the frozen failure rule. We check reference source hashes, both database manifests, the source partition, and the expected post-intervention state. Each scheduled cell is attempted once. Failed cells are retained, and arm order rotates across tasks. Implementation development used nine training instances before the dev protocol was frozen. Appendix B gives the full accounting, including failures before an intervention can be established.

3.2 Adaptive continuation under the same visible checkpoint

The reference suffix isolates the consequence of lost application progress. To test adaptive recovery, we select the lexicographically first reproducible harmed instance from each affected family, yielding nine families. Selection therefore conditions on state loss already confirmed to reduce native reward under the fixed reference suffix. The adaptive success rates measure recovery under that condition and do not estimate performance across the full development set or across all checkpoint positions.

We evaluate GPT-5.6-sol, GPT-5.5 and Claude Opus 5 with each provider's extra-high reasoning setting. An evaluated system comprises the model service and its execution scaffold. These settings do not imply equal compute. Each episode receives the original task, allowed application descriptions and API documentation, and the recorded prefix code and outputs. The full visible prompt and history are identical across the two restoration arms. A shared instruction encourages inspection of application state and cautions that remembered identifiers may be stale. It does not reveal the treatment or the correct state. The reference suffix and official evaluator are unavailable to the model.

For the adaptive test in Figure 1C, all three systems submit Python cells to the persistent interpreter and use ordinary application APIs to inspect or modify state. Each episode permits 20 execution-tool requests and 900 seconds. Task completion must be executed through the application API. Each separately frozen cohort runs five fresh episodes per arm and instance, giving 45 pairs and 90 episodes per system, 270 episodes in total. All use the same five-round randomized family and paired-arm schedule, fresh worlds and sessions, and hash-matched paired inputs. Cohorts are analyzed separately, excluding the initial 18-episode Sol pilot.

We record native success, execution requests and failures. A request can batch multiple API operations, so it measures interactions, not latency or billing. The primary effort estimate averages five paired differences within each family, then weights the nine families equally, retaining failures. We report arm means and SDs, individual differences, and 95% paired bootstrap intervals within fixed families. Whole-round resampling checks temporal blocking. These quantify variability on the selected instances within each campaign. Service windows and provider scaffolds differ, so between-system differences cannot be attributed solely to the model. Appendix C gives the schedule, analysis and Claude's authentication-only resumption, which retains all previously valid results.

3.3 Supporting information witnesses

The supporting Cybench audit reviews a fixed 21-task server-backed subset. Confirmation freezes two reachable worlds, two actions, their common context, reward semantics and execution order; it requires exact visible equality, opposite nonzero advantages and stable same-state controls. A separate study integration tests an unmodified LangGraph checkpointer on ten contexts per confirmed mechanism, checking saver lookup, snapshot recovery, time-travel continuation and service liveness. The adapters and checkpoint bank are study code. Appendix D supplies the complete witness, results and source-review provenance.

Table 1: AppWorld dev census and fixed-suffix native success. Confirmed score loss spans nine of 19 families (25 of 57 related instances). The qualified subset contains 28 instances from ten families; three instances in one family fail preintervention reproducibility checks and remain as conservative zeroes. The reproducible subset contains 25 instances from nine families.

Disposition	Instances
All dev (19 families)	57
Qualified (10 families)	28
No qualified boundary	18
Qualification error	11

Restoration	All qualified	Reproducible
Complete	25/28	25/25
Partial	0/28	0/25
Repair	25/28	25/25

4 RESULTS

4.1 OMITTED APPLICATION STATE CHANGES SCORES ACROSS TASK FAMILIES

Incomplete restoration changes native completion outcomes in nine of the 19 AppWorld dev families, covering 25 of the 57 related instances (Table 1). These counts give a lower bound over the complete finite task universe because tasks without a qualified comparison remain in the denominator. The effects include payment transactions and reminders, playback state, followed artists and liked songs, playlist creation, file moves, and note export. They are not confined to one service object or one action template.

The boundary rule qualifies 28 instances from ten families. The remaining instances comprise 18 with no preterminal database change and 11 qualification errors. On the qualified grid, Complete succeeds in 25/28 cells, Partial in 0/28, and Repair in 25/28. Three instances from one family fail to reproduce their qualification state before the intervention and contribute no confirmed treatment effect. Keeping them in the denominator prevents reproducibility filtering from inflating the confirmed fraction. The 11 qualification errors occur during reference-prefix execution, before restoration is compared. They establish neither native-score sensitivity nor its absence.

The 25 reproducible comparisons all change from successful Complete continuation to unsuccessful Partial continuation. Twenty-two Partial suffixes reach the official evaluator and fail. Three fail while using a missing application object. These outcomes establish sensitivity to omitted progress under the fixed suffix. They do not require the two-action reversal used to certify a strict OPD alias.

Omission need not roll back every effect. Restoring an interior snapshot of a multi-write statement, while retaining the later prefix, defeats the fixed suffix in all six qualified instances. Less-than-total rollback is therefore sufficient in these cases. By contrast, all nine selected reference solutions have no persistent database change at their statement-index midpoint: Complete and Partial then load equivalent databases and both succeed in 9/9 tests. This describes the timing of persistent effects in these solutions, not recovery from affected early checkpoints (Appendix B.4).

4.2 RESTORING THE OMITTED EFFECTS RESCUES THE REFERENCE CONTINUATION

Replaying the boundary statement rescues all 25 reproducible failures. Its saved post-intervention database matches the qualified boundary database, and the unchanged reference suffix again satisfies the official task tests. The repair control in Figure 1B ties the loss to the omitted application effects using reference-assisted reconstruction of the identified dependency. Whether an agent discovers and reconstructs that dependency is the separate behavioral question in Figure 1C.

Playlist creation makes the mechanism concrete. The reference prefix creates a playlist and retains its identifier in the Python namespace. Partial restoration loads the earlier database, in which that playlist does not exist. The suffix still addresses the remembered identifier and fails. Replaying the creation statement restores a usable playlist and the subsequent suffix succeeds. Here the problem is a split between retained computational progress and lost application progress.

The boundary selection does not inspect the eventual treatment reward. A reference suffix could repeat an omitted operation, make it irrelevant, or fail even after Complete restoration. The observed

rescue therefore validates the selected dependency effects, a result not implied solely by visible-state equality.

4.3 Successful agent recovery can conceal additional work

Adaptive recovery often restores native success, but uses additional requests on average in all three evaluated systems (Table 2). Across five fresh pairs in each selected instance, Complete succeeds in 45/45 episodes for each system. Partial succeeds in 40/45 for GPT-5.6-sol, 38/45 for GPT-5.5 and 40/45 for Claude Opus 5. Terminal reward alone omits the execution effort accompanying these outcomes.

Table 2: Adaptive continuation by evaluated system, with provider-specific extra-high reasoning. Each uses the same nine outcome-selected AppWorld instances, one per family, with five pairs per instance (90 episodes each; 270 total). The budget is 20 requests/900 s per episode. C/P: Complete/Partial. Failures remain in the means. $\Delta = \mathrm{P} - \mathrm{C}$; conditional 95% paired-bootstrap intervals hold families fixed.

	Native success		Mean requests		Paired change
System	C	P	C	P	Δ [95% interval]
GPT-5.6-sol	45/45	40/45	4.36	6.18	$+1.82$ $[1.42, 2.24]$
GPT-5.5	45/45	38/45	5.87	6.73	$+0.87$ $[0.20, 1.53]$
Claude Opus 5	45/45	40/45	4.22	5.87	$+1.64$ $[1.16, 2.18]$

The family-balanced increases are 1.82 requests for Sol, 0.87 for GPT-5.5 and 1.64 for Claude (Table 2). Whole-round resampling, which preserves within-round co-variation, gives [1.49, 2.16], [0.04, 1.51] and [1.00, 2.60], respectively. GPT-5.5's lower bound is only 0.04 requests above zero, making evidence for a positive average increment sensitive to round-level variation. Each estimate retains all failures and is computed separately.

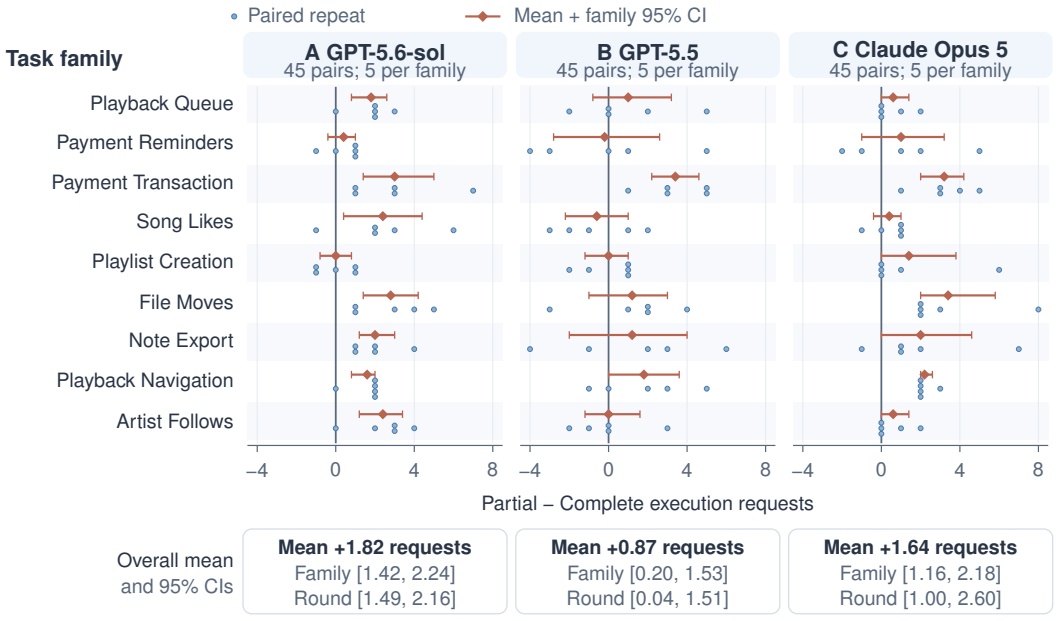

Figure 2: Recovery effort at the original checkpoint for three systems: 45 pairs each across nine selected instances. Blue dots are paired request differences; coral diamonds and bars are family means and exact 95% paired-bootstrap intervals. Vertical offsets prevent overlap. Bottom cards show within-family (Family) and whole-round (Round) resampling intervals. All failures, zeros and negative differences are retained: observed ranges are $[-1, 7]$ for Sol, $[-4, 6]$ for GPT-5.5 and $[-2, 8]$ for Claude. Intervals are conditional and unadjusted; cross-system contrasts are not model-only effects.

Figure 2 displays all three system cohorts, separating repeated paired differences from family means and intervals. Table 6 reports the corresponding Claude breakdown. The recurring average increase coexists with different family responses. Sol has eight positive family means and one zero. GPT-5.5 has five positive, two zero and two negative means, while Claude has nine positive means. Individual differences comprise 36 increases, five ties and four decreases for Sol, versus 25 increases, six ties and 14 decreases for GPT-5.5, and 31 increases, ten ties and four decreases for Claude. Because Complete episodes can also repeat work, these differences measure observed system responses, not minimum repair costs.

Sol's and Claude's five score losses each occur in the reminder family. GPT-5.5 has the same five reminder failures and two additional file-move failures. The reminder recurrence concerns one task dependency. The new file outcomes do not establish a separately audited failure mechanism.

Recovery effort at a later boundary. A separate Sol-only follow-up advances the transaction and playlist checkpoints by their one remaining nonterminal statement. All ten Complete and ten Partial episodes succeed, but their request means are 4.00 and 7.40, a paired increase of 3.40 with conditional 95% interval [2.60, 4.10]. On these same two instances at the original boundary, the means were 5.60 and 7.10, an increase of 1.50. The larger contrast at the terminal boundary mainly reflects less work in Complete, rather than much more work in Partial. Boundary choice changes the remaining task as well as the effects at risk. This separate run, with extended history and effect sets, does not isolate a monotonic position effect. It nevertheless shows that successful recovery can conceal an effort difference at both tested boundaries (Table 5).

4.4 A COMPLETION ACKNOWLEDGMENT CAN OUTLIVE THE EFFECTS IT ACKNOWLEDGES

In the payment-reminder task, the recorded prefix has already invoked the application's reminder operation. Complete restoration preserves its notification effects. Partial restoration retains the same visible history but loses those effects with the earlier database snapshot.

For each of the three systems, all five Complete reminder episodes succeed and all five Partial episodes fail native evaluation, although both arms register completion in every repetition. The reminder request difference averages $+0.4$ for Sol and -0.2 for GPT-5.5, versus $+1.0$ for Claude. Registered completion therefore does not establish persisting task effects, and repeated failure need not entail extra requests.

Applying the same candidate-effect-query rubric to Sol's and GPT-5.5's reminder traces finds no recorded query before completion in Sol's 5/5 Complete and 5/5 Partial episodes, versus GPT-5.5's 4/5 and 5/5. Thus recorded query absence accompanies success when effects persist and failure after rollback in both policies, without implying identical behavior. Across Sol's seven original terminal-only families, absence occurs in 6/35 Complete and 5/35 Partial traces; all five Partial cases are reminders. Candidate reads do not establish successful effect validation, and recorded absence does not exclude other channels.

A semantic audit of all 90 Claude continuations finds target-bound effect observations in 40/45 episodes per arm; all 40 non-reminder Partial episodes then repair omitted effects and succeed. All ten reminder episodes instead inspect inputs without reading notification effects. None of the five Partial episodes repairs them; two Complete episodes repeat the reminders. Appendix C links these distinctions to executed code, outputs and native events, separating effect observation from full coverage.

4.5 A COMPLEMENTARY INFORMATION WITNESS

Cybench's LOCKTALK illustrates the information distinction at another service boundary: identical client checkpoints require different terminal requests when the active server's startup key changes. The witness persists through the tested LangGraph saved-state recovery checks, so faithful runtime recovery need not preserve decision-relevant information. This is a local action-ranking witness, complementary to the AppWorld native-score results, not another estimate of benchmark-wide score loss (Appendix D).

5 Discussion and Limitations

A recovered score includes reconstruction work. Fixed-suffix failure and adaptive success can arise from the same omission because an adaptive agent can reconstruct the missing state. All three systems show an average request increase, but their family responses differ. Joint reward–effort reporting captures both native failure and the interaction work accompanying recovery. Success alone leaves that work unobserved.

Budget is part of the evaluation condition. Counting recovery requests within the task budget evaluates continuation together with reconstruction. Excluding them requires specifying the work supplied before scoring. The restoration boundary thus helps determine what capability a score measures. Request counts are not latency, API traffic, or monetary cost, and do not establish that a smaller budget would necessarily cause score loss.

Separate recovery points. Independent recovery can pair agent and application checkpoints from different times. Suppose snapshot d_0 precedes playlist creation, whose successful identifier is retained in later agent checkpoint h_b. Restoring h_b with d_0 preserves the identifier but loses the playlist. LangGraph documents the relevant boundary: graph-state checkpoints and replay that skips preceding nodes (LangChain, 2026a;b). A joint consistent restore, or effect reconciliation before dependent execution, prevents this mismatch. Exactly-once RPC systems such as RIFL co-persist completion records and operation effects (Lee et al., 2015). A cached acknowledgment alone cannot recover an omitted effect. Our intervention measures the consequences of this mismatch, whose deployment incidence remains unmeasured.

Completeness can be dependency-specific. The relevant boundary follows from what the continuation needs. A task that uses a created playlist requires that object or an explicit way to recover it, while unrelated process state need not be preserved. The local AppWorld sufficiency certificate provides the complementary case: changes to an unused marker leave a fixed answer-only comparison intact. These examples support specifying saved and excluded dependencies, the continuation payoff, and the permitted validation or repair channel. Observation can identify a missing effect without recreating it. A warning or attempted check does not establish validation. The contract therefore distinguishes restoring an effect, making its state observable, and relying on the resumed policy to respond. It does not supply an algorithm for finding a globally minimal checkpoint.

Scope of evidence. The census and adaptive study concern one frozen AppWorld version under a deliberate partial-restoration intervention. Qualification errors and pre-intervention nonreproducibility limit the identified affected subset and remain in the accounting. The adaptive study characterizes recovery in affected families selected after the fixed-suffix outcomes. Its intervals describe three evaluated systems in separate five-round campaigns under their respective provider scaffolds. Joint reward–effort reporting captures both recovery and failure, including negative paired differences. The later-boundary follow-up remains Sol-only. Appendix C provides the campaign provenance and query-audit details, including the separate human review and AI-assisted semantic audit of Claude's continuations. The strict Cybench witnesses and positive sufficiency certificate remain local to their declared actions and worlds. Neither deployment prevalence nor a universal loss-or-cost law is established.

6 Conclusion

Restoration completeness depends on the continuation a checkpoint must support. Omitting application effects changes native scores in nine AppWorld task families, and restoring those effects rescues all 25 affected continuations. Three evaluated systems recover much of the lost progress with additional requests on average, yet all three also register completion despite missing required effects. These outcomes distinguish preserved progress from successful reconstruction and from completion without the required result.

Decision contracts make the restoration boundary, continuation payoff and available repair actions explicit. Evaluations should report native reward and recovery effort together under the stated interface and budget. A recovered score alone does not establish equivalent evaluation conditions.

AI USE STATEMENT

Generative AI was used to assist with manuscript writing and language polishing and with the post-run semantic double-check audit of all 90 Claude continuation trajectories. The audit examined executed code, returned values, task requirements, and native event logs to characterize effect observations, effect-predicate coverage, repair, and repeated execution. This AI-assisted audit was separate from native task scoring and the human semantic-query review. Its scope and results are reported in Appendix C.3 and Table 7. All figures were drawn and revised manually by the authors, without generative AI assistance. We take responsibility for the final manuscript, analyses, and reported results, including the AI-assisted audit labels.

ETHICS STATEMENT

The study uses public benchmark tasks in isolated research environments. AppWorld actions operate on fictional accounts within its local simulation; payment and notification APIs do not affect real financial or communication services. The Cybench witnesses use public CTF task implementations. We do not test external services or collect data about identifiable people.

REPRODUCIBILITY STATEMENT

Section 3 specifies the task universe, mechanical checkpoint selection, restoration arms, evaluator and model interface. Appendix A gives assumptions and proofs; Appendices B and C give full task accounting, all paired outcomes for three separately analyzed systems, their settings and run windows, the Sol-only follow-up, and the recorded-query audit with its ten-episode GPT-5.5 reminder extension. A separate AI-assisted semantic audit covers all 90 Claude continuations, distinguishing effect reads, coverage and reconstruction from completion and native reward. Its interrupted campaign retains all 58 prior valid results and documents the 32 later episodes. Appendix D records strict witnesses and source adjudications. Protocols bind code, task sources, schedules and result files by hashes; separate verification implementations recompute the main results from preserved records. Paper tables and the paired-call figure are traceable to their checked summaries.

The accompanying paper-source package includes sanitized census and paired-run summaries, credential-free Claude semantic labels, and an arithmetic verifier that recomputes the reported statistics, audit counts and plotted data without model calls. Reaggregating labels does not independently reproduce semantic adjudication from trajectories.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Jason Ansel, Kapil Arya, and Gene Cooperman. DMTCP: Transparent checkpointing for cluster computations and the desktop. In *IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–12, 2009. doi: 10.1109/IPDPS.2009.5161063.

David Blackwell. Equivalent comparisons of experiments. *The Annals of Mathematical Statistics*, 24 (2):265–272, 1953. doi: 10.1214/aoms/1177729032.

K. Mani Chandy and Leslie Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75, 1985. doi: 10.1145/214451.214456.

C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970. doi: 10.1109/TIT.1970.1054406.

Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley, Norm Hutchinson, and Andrew Warfield. Remus: High availability via asynchronous virtual machine replication. In *USENIX Symposium on Networked Systems Design and Implementation*, pp. 161–174, 2008.

URL https://www.usenix.org/legacy/events/nsdi08/tech/full_papers/cully/cully.pdf.

Yunpeng Dong, Jingkai He, Shiqi Liu, Yuze Hou, Dong Du, Zhonghu Xu, Si Yu, Baochuan Yang, Yubin Xia, and Haibo Chen. DeltaBox: Scaling stateful AI agents with millisecond-level sandbox checkpoint/rollback. *arXiv preprint arXiv:2605.22781*, 2026. URL https://arxiv.org/abs/2605.22781.

Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11:1605–1641, 2010. URL https://www.jmlr.org/papers/v11/el-yaniv10a.html.

E. N. Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys*, 34(3):375–408, 2002. doi: 10.1145/568522.568525. URL https://www.cs.cornell.edu/courses/cs614/2004sp/papers/EAWJ02.pdf.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1–2):163–223, 2003. doi: 10.1016/S0004-3702(02)00376-4.

Aryan Gulati. ContinuityBench: A framework and taxonomy for evaluating agent recovery from interrupted state. In *ICML Workshop on Combining Theory and Benchmarks*, 2026. URL https://openreview.net/forum?id=3N3BzvoLbG.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=Zy4uFzMviZ.

LangChain. LangGraph persistence. Official documentation, 2026a. URL https://docs.langchain.com/oss/python/langgraph/persistence. Accessed 2026-09-05.

LangChain. Use time-travel in LangGraph. Official documentation, 2026b. URL https://docs.langchain.com/oss/python/langgraph/use-time-travel. Accessed 2026-09-05.

Collin Lee, Seo Jin Park, Ankita Kejriwal, Satoshi Matsushita, and John Ousterhout. Implementing linearizability at large scale and low latency. In *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015. doi: 10.1145/2815400.2815416. URL https://web.stanford.edu/~ouster/cgi-bin/papers/rifl.pdf.

Letta. Recovery-Bench. Official project repository, 2026. URL https://github.com/letta-ai/recovery-bench. Repository version accessed 2026-09-22.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for MDPs. In *Ninth International Symposium on Artificial Intelligence and Mathematics*, 2006. URL https://thomasjwalsh.net/pub/aima06Towards.pdf.

PostgreSQL Global Development Group. Continuous archiving and point-in-time recovery (PITR). PostgreSQL 18 documentation, Section 25.3, 2026. URL https://www.postgresql.org/docs/18/continuous-archiving.html. Accessed 2026-09-15.

Anka Reuel, Amelia Hardy, Chandler Smith, Max Lamparth, Malcolm Hardy, and Mykel J. Kochenderfer. BetterBench: Assessing AI benchmarks, uncovering issues, and establishing best practices. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 37, 2024. doi: 10.52202/079017-0685. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/26889e8359e7ef8a7f5d77457364ca55-Abstract-Datasets_and_Benchmarks_Track.html.

Shangyin Tan, Kevin Lin, Koushik Sen, and Matei A. Zaharia. Recovery-Bench: Evaluating agentic recovery from mistakes. In *NeurIPS Workshop on Evaluating the Evolving LLM Lifecycle*, 2025. URL https://openreview.net/forum?id=8FZRnDgDxq.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Steven D. Whitehead and Dana H. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83, 1991. doi: 10.1023/A:1022619109594.

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

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2404.07972.

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

Andy K. Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin W. Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Mike Yang, Teddy Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Polycarpos Yiorkadjis, Kenny Osele, Gautham Raghupathi, Dan Boneh, Daniel E. Ho, and Percy Liang. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.08926.

Yusheng Zheng, Yiwei Yang, Wei Zhang, and Andi Quinn. ACRFence: Preventing semantic rollback attacks in agent checkpoint-restore. *arXiv preprint arXiv:2603.20625*, 2026. URL https://arxiv.org/abs/2603.20625. CoDAIM Workshop 2026.

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. URL https://arxiv.org/abs/2307.13854.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Antony Kellermann, Jasjeet Sekhon, Jacob Steinhardt, Sarah Schwettmann, Arvind Narayanan, Matei A. Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. Establishing best practices in building rigorous agentic benchmarks. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 38, 2025. doi: 10.52202/085713-5547. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f316275b44ee2de533102913828a8107-Abstract-Datasets_and_Benchmarks_Track.html.

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. URL https://arxiv.org/abs/2608.14380.

A Formal Specification and Proofs

A.1 Operational decision contract

A contract fixes the restoration map ρ, reachable worlds $\mathcal{W}$, continuations and native payoff. To distinguish immediate and attainable identifiability, it also declares allowed observations $u \in \mathcal{Q}$, signals $s \in \mathcal{S}$, a kernel $K(ds, dw' \mid w, u)$, interaction budget, and fallback behavior. The kernel permits the world to change during checking: information about w need not remain valid in w'. Identifiability after observation is relative to an allowed strategy within this budget.

The AppWorld bridge exposes ordinary application APIs; it does not expose a perfect binary probe. A policy can inspect state and recreate an omitted effect. The earlier binary examples use a stable-world signal model solely to describe the information constraint. They do not calibrate the operational cost measured by the native continuation experiment.

A.2 Decision-sign factorization

Fix the reachable world set $\mathcal{W}$, observation map ρ, two continuations, and payoff used in Equation 1. A function $g : \rho(\mathcal{W}) \to \{-1, 0, +1\}$ satisfying Equation 2 exists if and only if $\operatorname{sgn} \Delta$ is constant on every nonempty fiber $\mathcal{W}_z$. For necessity, any $w, w' \in \mathcal{W}_z$ have signs $g(z)$. For sufficiency, define $g(z)$ to be the common sign on that fiber; constancy makes the definition independent of the chosen world. Values of g outside the image of ρ can be assigned arbitrarily.

An OPD alias therefore refutes factorization. The converse does not hold: advantages zero and one on the same fiber violate factorization but do not give a strict reversal. Factorization also does not imply equal reward levels. For example, continuation values $(1, 0.5)$ in one world and $(0.5, 0)$ in another have the same positive advantage but different values for either fixed continuation. These are elementary properties of the declared comparison, not a new state-abstraction theorem.

A.3 A sufficient condition for reward–effort equivalence

The local comparison omits the observation and repair trajectory. To state a sufficient condition for preserving that trajectory, let H_t be the complete agent-visible history before decision t, including the current execution-request count. Fix a finite interaction horizon B, a policy $\pi(du \mid H_t)$, and the same initial visible history h_0 in complete and partial restoration arms. Include final submission and stopping among the policy's actions. Let the environment response contain the next visible observation, whether execution terminates, the effort increment, and the native reward if termination occurs. A forced stop at horizon B is also an environment response and includes its native reward. The relevant conditional kernels are assumed to exist.

Proposition 1 (Controlled continuation equivalence)**.** *Suppose that, at every history reachable from h_0 under π, each action in the policy's support has the same conditional environment-response law in both restoration arms. The policy, interface, horizon, and rule for accumulating effort are the same in both arms. Then the two arms induce the same joint law of terminal native reward R and accumulated effort K. In particular, when expectations exist, $\mathbb{E}[R^c - R^p] = \mathbb{E}[K^p - K^c] = 0$.*

Proof. The initial histories agree. Suppose the distributions of visible histories agree before decision t. Conditional on a shared history, both arms draw the next action from the same policy kernel. By assumption, integrating their common environment-response kernel over this common action law gives the same conditional distribution of the next observation, termination, effort increment, and any terminal reward. Hence the history distributions, accumulated efforts, and already terminated reward records agree after decision t. Induction reaches the finite horizon B, where any remaining execution terminates under the common forced-stop response. Terminal reward and accumulated effort therefore have the same joint distribution. Equality of the stated expectations follows when they exist. □

The proposition is a sufficient controlled-process condition, not a criterion verified for every benchmark task. In particular, native terminal reward belongs in its assumptions: equal visible tool outputs alone are insufficient when the evaluator checks an omitted application effect. Stochastic timing or a wall-clock timeout must also be included in the response law if they affect termination. Equality

in distribution does not require equality between two independently sampled episodes. Conversely, failure of the assumptions does not imply either reward loss or added effort. An omitted variable can remain unused along the evaluated policy, and different trajectories can yield the same reward and request count. The empirical study therefore measures both quantities rather than invoking a universal loss-or-cost alternative.

A.4 Visible-only binary decisions

Consider a two-world OPD pair with conditional probability $p = \Pr(w^+ \mid z) \in (0, 1)$. A visible-only forced rule chooses a^+ with probability $r \in [0, 1]$, using randomization independent of the world conditional on z. Under unit wrong-sign loss its risk is

$$\mathcal{R}(r \mid z) = p(1 - r) + (1 - p)r. \tag{4}$$

Proposition 2 (Visible-only Bayes boundary)**.** *Under these assumptions,* $\inf_r \mathcal{R}(r \mid z) = \min(p, 1 - p)$*. For native payoff gaps* $d_+ = \Delta(w^+) > 0$ *and* $d_- = -\Delta(w^-) > 0$*, minimum expected regret relative to a world-informed choice is* $\min\{pd_+, (1 - p)d_-\}$.

Proof. Both risks are affine in r. The unit-loss endpoint values are p at $r = 0$ and $1 - p$ at $r = 1$. The payoff-regret expression is $pd_+(1 - r) + (1 - p)d_-r$, with endpoint values pd_+ and $(1 - p)d_-$. In each case an endpoint with smaller value attains the infimum. □

These are classical binary Bayes calculations; the empirical contribution does not claim them as novel theory.

A visible-only zero-wrong-sign rule that may abstain or return both actions cannot output either singleton with positive probability. Choosing only a^+ is wrong in w^-, which has positive mass, and the symmetric argument holds for a^-. If deciding is conditionally independent of the world given z, selective coverage cannot change the posterior among decided cases. With positive coverage the minimum conditional wrong-sign risk remains $\min(p, 1 - p)$.

A.5 An idealized observation cost

Fix the world during observation and let a binary signal S have symmetric reliability $\Pr(S = + \mid w^+) = \Pr(S = - \mid w^-) = q \geq 1/2$. The minimum posterior wrong-sign risk is

$$\mathcal{R}_{\text{probe}}(p, q) = \min\{pq, (1 - p)(1 - q)\} + \min\{p(1 - q), (1 - p)q\}. \tag{5}$$

For each signal value the optimal action incurs the smaller of its two joint world–signal masses, which gives the expression by summation. A signal with normalized cost c weakly improves total decision loss exactly when $c \leq c^\star(p, q)$, where

$$c^\star(p, q) = \min(p, 1 - p) - \mathcal{R}_{\text{probe}}(p, q). \tag{6}$$

An uninformative signal has zero value and a perfect signal has value $\min(p, 1 - p)$. These units measure decision loss in the stable binary model, not the Python execution requests, time, or API traffic measured in an adaptive application task.

B Native AppWorld Boundary Census

B.1 Runtime, partition, and reachable states

We use AppWorld 0.1.3.post1 and Python 3.11.16, with runtime seed 630063. The dev list contains 57 nonempty identifiers. We retain their file order and use the part before the underscore as the family identifier. Nine train instances support implementation development; six qualify and all six engineering Complete/Partial/Repair triples have rewards $(1, 0, 1)$.

The parser accepts a compiled reference function with exactly one `complete_task` call in its last top-level statement. It executes earlier statements successively and compares saved database manifests with the initial snapshot. The first differing statement defines the boundary; qualification stops there without using formal treatment rewards. The source, partition, context-output and database hashes are retained.

Table 3: All 19 AppWorld dev families, each with three instances. Qualified follows the mechanical boundary rule; confirmed loss additionally requires a reproducible Complete success and Partial failure. Numeric columns count instances; a zero in Confirmed loss is not evidence of no effect. Descriptions summarize the boundary statement's application operations.

		Instances	
Family	Boundary or disposition	Qualified	Confirmed loss
`50e1ac9`	No preterminal database change	0	0
`fac291d`	No preterminal database change	0	0
`530b157`	Payment transaction	3	3
`4ec8de5`	No preterminal database change	0	0
`b119b1f`	Playback navigation	3	3
`d4e9306`	Artist follows	3	3
`0d8a4ee`	Qualification error	0	0
`37a8675`	Qualification error	0	0
`3ab5b8b`	Qualification error	0	0
`df61dc5`	Transaction likes; nonreproducible state	3	0
`383cbac`	No preterminal database change	0	0
`23cf851`	No preterminal database change	0	0
`57c3486`	Song likes; two qualification errors	1	1
`68ee2c9`	File moves	3	3
`6bdbc26`	No preterminal database change	0	0
`6171bbc`	Playlist creation	3	3
`6c2c621`	Note export to files	3	3
`396c5a2`	Playback queue	3	3
`4fab96f`	Payment-request reminders	3	3
Total	57 instances, 19 families	28	25

Native `load_state` reloads application database increments relative to the initialized task; it is not a full snapshot of the interpreter or host. Every formal arm starts from a fresh task instance, recreates the prefix, checks its initial and boundary hashes against qualification, loads the assigned database, and runs the suffix. Complete preserves the application effects; Partial rolls them back while leaving the Python computation at the later point. Repair replays the entire selected statement, including every operation in a loop. This is a reference-assisted intervention, not an agent policy.

The intervention pairs two reachable checkpoints without modifying task logic. The study deliberately selects these times; no such mismatch is attributed to AppWorld's default workflow. Its fixed boundary rule can remove most achieved progress near completion and does not sample random interruptions.

Separate recovery domains. The corresponding deployment condition is independent recovery of the agent and application, with no reconciliation before dependent execution. Database point-in-time recovery can legitimately restore an internally consistent earlier state (PostgreSQL Global Development Group, 2026); pairing it with a later agent checkpoint can nevertheless retain results of rolled-back operations. An agent-only crash with an intact application database does not create this mismatch. Nor does independent snapshot timing alone: the recovery procedure must actually combine the mismatched points. This is an architectural interpretation of the intervention, not an observed deployment incident.

B.2 Full task disposition

Table 3 accounts for all dev instances. A family can contain both a qualified instance and qualification errors, so the instance categories partition 57 tasks but their family coverages need not partition 19. The 18 no-boundary instances are not task-wide certificates: the test inspects one reference execution and one preterminal boundary rule.

All 11 qualification errors are reference-prefix execution failures, before any formal restoration arm. With zero-based statement indices, these are `0d8a4ee_1/2/3` at index 5, `37a8675_1/2/3` at index 8, `3ab5b8b_1/3` at index 9 and `3ab5b8b_2` at index 8, and `57c3486_1/3` at index 4. They are unqualified, not observed negative treatment effects.

Three instances in the transaction-like family `df61dc5` do not reproduce their qualified state. Eight cells fail the state-hash gate before a valid intervention comparison. These are all three arms for instances 2 and 3 and the Complete and Repair arms for instance 1. The remaining cell, `df61dc5_1` Partial, passes its integrity checks but fails four official tests and receives zero. Without a reproducible Complete comparison, none of these three instances contributes a confirmed effect. All remain in the 28-instance grid under the conservative failure rule. This differs from the three playlist Partial suffixes, which fail after intervention on a missing object while their Complete and Repair companions pass. A separate post-run verifier checks these failure phases.

B.3 Scoring and uncertainty

The 28 qualified instances produce 84 scheduled arms, with rotated condition order and one attempt per cell. Execution failures are zeroes rather than replaced attempts. Twenty-five Complete and Repair cells pass the official evaluator. Of the reproducible Partial cells, 22 reach it and fail, and three stop on a missing-playlist error.

The conservative mean Complete-minus-Partial difference is $25/28$. The frozen analysis resamples the ten qualified families 10,000 times with seed 630063, retaining each family's variants, giving a 95% bootstrap interval $[0.679, 1.000]$. This is resampling sensitivity, not uncertainty about the enumerated dev denominator. The finite counts $25/57$ instances and $9/19$ families carry the breadth claim.

B.4 Checkpoint position and snapshot age

Two diagnostics separate visible checkpoint position from application-snapshot age, without adaptive model calls. For the position test, let t be the zero-based index of the final top-level completion statement. We execute the prefix through $k = \lfloor (t-1)/2 \rfloor$ and preserve that midpoint namespace, then load either the midpoint or initial database and run the original suffix from $k+1$. The nine instances are the original adaptive selection, not a new set selected using midpoint outcomes. A train-task smoke test precedes the frozen 18-cell run, with fresh processes, one attempt per arm, and a 900-second cell limit. All paired source, prefix-output and database checks pass; an independently implemented verifier matches all 18 saved native evaluations to their rewards. Complete and Partial both succeed in 9/9. All nine midpoint manifests match their initial manifests, so this is an early no-state-change stratum. The later selected boundaries on these same instances yield 9/9 versus 0/9.

The snapshot-age diagnostic instead keeps the original later prefix and its namespace. It captures the database after half the effectful native API calls inside the selected multi-write statement, and restores that interior snapshot before the original suffix. Six instances permit such a snapshot: playback navigation (four effectful calls), artist following (19), song liking (58), file moves (100), note export (28), and reminders (seven). All six interior snapshots yield unsuccessful Partial suffixes. The queue, transaction and playlist statements are atomic at this resolution and do not admit an interior snapshot. On the nine-instance grid, Complete succeeds in 9/9, Partial with the initial database in 0/9, and a separate joint restart of both namespace and database followed by the full reference solution in 9/9. All 33 diagnostic cells pass their integrity checks. Joint restart is a reference-solution control, not a measured adaptive recovery strategy.

The position test changes both prefix and suffix; the snapshot-age test holds the later prefix fixed and varies the omitted subset. They establish a safe early stratum and sensitivity to partial rollback in six later multi-write cases. Combining them into one position curve would conflate interventions; neither samples natural interruptions or demonstrates a deployed mismatch.

B.5 A local sufficiency certificate

A separate read-only case, `50e1ac9_1`, instantiates sign factorization. Fix the task, initialized database and an observed title–play-count list L. The task asks for the four most-played R&B titles. With official answer normalization N, let $T(z^\star) = N(\mathrm{Top4}(L))$. Candidate a^+ submits those titles; a^- submits ranks two through five. The fourth play count exceeds the fifth, so the normalized answers differ.

Permitted worlds vary only an unused temporary marker $u \in \{0, \ldots, 9\}$; database and observation $z^\star$ remain fixed. The evaluator checks agreement with its fixed reference $R^\star$ and absence of application modification:

$$V(w_u, a) = \mathbf{1}\{N(a) = N(R^\star)\}\mathbf{1}\{\text{no application-model changes}\}.$$

Neither suffix nor evaluator reads u. Native checks verify $T(z^\star) = N(R^\star)$: the top-four answer succeeds in all ten worlds and the alternative fails answer agreement in all ten. Thus $g_{\text{loc}}(z^\star) = +1$ factors the advantage sign on this single fiber. All twenty save/load round trips agree. The certificate is restricted to this fixed continuation and these permitted worlds.

C Repeated Adaptive AppWorld Continuation

C.1 Selection, interface, and repeated schedule

The original 18-episode Sol bridge is a pilot. Each policy cohort fixes the same lexicographically first reproducible harmed instance from each of nine census families and runs five new Complete/Partial pairs per instance. Selection conditions on fixed-suffix harm, not on adaptive outcomes. The three 90-episode cohorts have separately frozen qualifications, source files and schedules, each fixed before its formal model calls. The policy cohorts and pilot remain separate.

GPT-5.6-sol and GPT-5.5 use extra-high reasoning and authenticated ChatGPT subscription transport. Claude Opus 5 uses the exact service name `claude-opus-5`, the `xhigh` setting and the official subscription CLI (version 2.1.274). Its built-in tools are disabled; only the AppWorld execution tool is exposed through MCP. The system prompt is replaced with the study instructions; skills, additional MCP servers and user/project settings are disabled, although managed settings can still apply. Initialization checks the requested model and tool list. Provider role/tool envelopes and execution scaffolds differ, so these are three evaluated systems, not a model-only comparison or equal-compute experiment. The sole dynamic tool executes a Python cell in the persistent AppWorld interpreter. The model cannot access the evaluator, reference suffix, benchmark sources, host environment or real accounts. Both arms receive identical task instructions, API documentation and prefix code/output history. A shared instruction encourages inspecting application state and warns that remembered identifiers may be stale. This tests recovery with that instruction, not unaided discovery of an interruption.

Each episode allows 20 budgeted execution requests and 900 seconds. Five rounds shuffle all nine families and randomize arm order within each adjacent two-episode block (schedule seed 2026091401). Up to two independent blocks run concurrently; rounds are separated by a barrier. Every episode starts a fresh process, task reconstruction and ephemeral model session. No valid episode is replaced. All 270 primary episodes are scored, with 45 pairs per system. Input checks bind paired prompt/history hashes to the original qualification. A prelaunch clock adapter uses real monotonic time for the wall-time cutoff; the simulated task clock remains unchanged. Before GPT-5.5 dispatch, all 18 arm/instance qualifications pass without model calls, and all paired visible inputs match. One synthetic preflight is excluded. All 90 formal GPT-5.5 episodes pass the infrastructure-validity checks; none is excluded or replaced.

No temperature or top-p override is supplied. The model name is a service alias, not an immutable weight revision. Randomized temporal blocks and repeated episodes measure variability during the campaign; they cannot exclude common backend drift or establish exact replication after a provider update. The Sol original-checkpoint campaign was frozen on 14 September 2026 at 06:03 UTC and completed at 09:57 UTC. GPT-5.5 was frozen on 15 September at 06:00:39 UTC, with formal model calls from 06:01:20 to 06:53:04 UTC. Claude's campaign runs on 17–18 September 2026. An authentication interruption leaves 58 valid episodes, two authentication-failed attempts and 30 pending cells. After reauthentication under a different subscription account, a recorded amendment retains all 58 valid results byte-for-byte and completes the remaining 32 cells, including the two authentication retries. Source, scoring, qualification and schedule remain unchanged. The failed attempts are preserved outside the 90 valid outcomes; no valid episode is rerun. The account and run-window transition remains part of the campaign provenance. The same task selection, task-level prompt/history, schedule and limits support within-system state contrasts. Different service windows and scaffolds prevent a model-only causal interpretation of differences between systems. Separate verification checks schedules, source hashes, cell identities, rewards and request counts.

C.2 MAGNITUDE, VARIATION, AND NATIVE OUTCOMES

For family f and repeat r, let $d_{fr} = K^p_{fr} - K^c_{fr}$. The primary estimate is $\frac{1}{9}\sum_f \frac{1}{5}\sum_r d_{fr}$. Computed separately, it equals $82/45 = 1.8222$ requests for Sol and $39/45 = 0.8667$ for GPT-5.5, and $74/45 = 1.6444$ for Claude; Complete/Partial arm totals are 196/278, 264/303 and 190/264, respectively. All five Sol, seven GPT-5.5 and five Claude Partial failures remain in reward and request statistics. The paired-difference SDs are 1.75, 2.61 and 2.08, with ranges $[-1, 7]$, $[-4, 6]$ and $[-2, 8]$, respectively. Sol's round means are 2.11, 1.78, 1.67, 2.33 and 1.22; GPT-5.5's are 1.11, -0.67, 0.67, 1.44 and 1.78; Claude's are 1.44, 0.89, 1.44, 0.89 and 3.56.

Following recommendations to report uncertainty in few-run evaluation (Agarwal et al., 2021), we report intervals alongside point estimates. The frozen analysis uses 10,000 bootstrap draws (seed 2026091402). Resampling the five pairs within each fixed family gives conditional 95% intervals [1.42, 2.24] for Sol, [0.20, 1.53] for GPT-5.5 and [1.16, 2.18] for Claude; resampling whole rounds gives [1.49, 2.16], [0.04, 1.51] and [1.00, 2.60], respectively. The former retains the selected families; the latter retains within-round co-variation across the five observed rounds. GPT-5.5's round-block lower bound is close to zero, indicating sensitivity to the resampling unit rather than a contradiction between intervals. Neither is a prevalence interval or a model-version guarantee. The per-family intervals in Tables 4 and 6 additionally enumerate all 5^5 empirical bootstrap resamples. They are descriptive, unadjusted intervals, not simultaneous familywise bounds. Arm SDs and the plotted individual differences show variation without relying on a sign-test headline.

C.3 COMPLETION EFFECTS AND TERMINAL-BOUNDARY FOLLOW-UP

In each system's repetitions, all five Complete reminder episodes succeed and all five Partial episodes register completion but fail native tests. Sol's paired request differences are $0, 1, 1, 1, -1$; GPT-5.5's are $1, -3, -4, 0, 5$; Claude's are $1, -1, 2, -2, 5$. GPT-5.5 additionally fails two Partial file-move episodes (`68ee2c9_1`). These reward records do not identify their behavioral cause. All 90 GPT-5.5 episodes have registered completion; this field is distinct from the completion-call trace audit below, which covers all 110 Sol episodes and only the ten GPT-5.5 reminder episodes. Claude also registers completion in all 90 episodes. In addition to human semantic-query review, a separate AI-assisted audit examines all 90 executed continuations, comprising 454 Python requests and 6,075 native API events. It reads submitted code, returned values, task requirements and native evaluator predicates, resolving helper calls and namespace access against the executed event sequence. Every continuation request has a contiguous native-log window, accounting for the request-tracker reset on state loading. Model-private reasoning is excluded. The coding rubric is fixed before trajectory inspection, but outcome summaries are already known. This post-run audit is distinct from both the native-score verifier and an independent-human agreement measurement.

Effect observation and reconstruction in Claude's continuations. Both arms contain 40/45 episodes with a successful target-bound observation of effect presence or absence. All 40 non-reminder Partial episodes subsequently attempt repair and pass native evaluation. Complete episodes retain these effects and do not repeat their associated mutations. Adding songs to the created playlist and sending the payment-related text message are remaining reference-suffix work, not repair of the omitted boundary effect. Full coverage of the family-specific effect predicate is narrower: 23/45 Complete and 28/45 Partial episodes. The remaining observations omit checks such as file-content correspondence, retention of prior likes, or exhaustive payment multiplicity. These coverage labels concern evidence obtained by the agent, not whether an unexamined predicate ultimately holds.

All ten reminder continuations lack a recipient-notification effect read. Payment-request status, contact lists and cached loop variables identify which requests need reminders, but do not establish delivery of the notification effect. None of the five Partial continuations sends a replacement reminder. Two Complete continuations each repeat all seven reminders and still pass: the native evaluator compares recipient, message and request-ID sets, without rejecting duplicate identical notifications by multiplicity. Thus these successes do not establish duplicate-free execution. Whether the permitted accounts expose a usable recipient-notification read remains unestablished; the observed absence of such a read is not a test of that affordance.

Seven original checkpoint families already leave only the completion statement in their reference suffix. A separate Sol-only follow-up advances the transaction and playlist checkpoints by their

one remaining nonterminal statement, producing two additional terminal-only checkpoints. The new effect sets include both transaction and text-message effects, and both playlist creation and membership. Partial still loads the initial application database. The extended prefix is identical across the new arms but intentionally differs from the original checkpoint. Five new pairs per family give 20 additional episodes (Table 5): Complete 10/10 and Partial 10/10 success; request means are 4.00 and 7.40. The mean paired difference is 3.40, with within-family bootstrap interval [2.60, 4.10] and whole-round interval [3.10, 3.70]. For the same two instances at the original checkpoint, both arms succeed in 10/10, with request totals 56 and 71, means 5.60 and 7.10, and paired difference 1.50. The change in contrast is primarily a decrease in Complete requests. These are separate campaigns with different visible histories and completed effect sets, not a randomized comparison of boundary positions under an identical decision problem. This two-family follow-up is not pooled into the primary effort estimate. Sol's primary and follow-up cohorts together contain 110 episodes.

Effect-query analysis distinguishes candidate reads, successful target-bound validation, and full effect coverage. Task/date and source-input queries do not establish omitted effects. Payment-request status likewise does not establish a recipient's reminder notification, which the sender may not observe. Query attempts alone cannot establish successful verification or blame failure on an agent's verification ability.

For those 110 Sol episodes and the ten GPT-5.5 reminder episodes, we audit the recorded native API calls before the unique `complete_task` event, requiring that its Python execution succeeds and the completion acknowledgment is registered, irrespective of native reward. A frozen family-specific rubric identifies candidate reads relevant to the omitted effects. Public route definitions map method/URL pairs to API names; generic task/date queries do not count. The audit checks request/transcript hashes and contiguous native-log windows, accounting for the fresh request tracker created by state loading. All 110 Sol episodes and all ten GPT-5.5 reminder episodes have a unique confirmed completion-call execution and matching request records. This is a post-run trace audit, separate from the frozen native-score verifier. The GPT-5.5 extension reuses the same candidate-read rubric, route mapping and code-structure rules without model calls or new exemptions; it does not extend the seven-family or terminal-boundary analyses to GPT-5.5.

Table 8 distinguishes all recorded traces from the conservative subset with no unresolved structural flags in submitted code. Direct log-suppression flags were absent, but indirect namespace access and unresolved code structures cannot be treated as proof that every call was observable. For Sol's reminder instance, both arms have 5/5 recorded absences; the conservative counts are 4/4 Complete (one unknown) and 5/5 Partial. GPT-5.5 has 4/5 Complete and 5/5 Partial recorded absences, with 4/4 in each conservative subset and one unknown per arm. Both GPT-5.5 unknowns involve indirect namespace access, retained under the unchanged rule. Its one Complete episode with candidate reads makes four notification-query attempts before completion and is also structurally unknown. The API scopes notifications to the authenticated account; these attempts do not establish recipient-bound validation or full effect coverage. All five Partial reminders fail native evaluation for each policy despite registered completion. These counts characterize recorded attempts, not the absence of every possible verification channel.

C.4 Provenance and earlier pilot

All three cohorts were qualified and frozen before model dispatch; Claude's authentication-only resumption is documented above. Frozen-record and separate numerical checks pass for all 270 primary and 20 Sol follow-up episodes; these are independent implementations, not a second research team. Raw logs remain on the server. The archived 18-episode Sol pilot has Complete 9/9 versus Partial 8/9 success and 36 versus 58 requests ($+2.44$); its gate-name collision was corrected before calls. Pilot episodes and its exploratory sign test are excluded from the repeated estimate and uncertainty.

D Strict Witnesses and Source Audit

D.1 Complete LockTalk witness

The official application samples an RSA signing key at startup. We start two independent instances with keys K_+ and K_- and freeze one instance-specific administrator request from each before

evaluating either request. The study-authored checkpoint contains both requests but no active-instance binding; the candidate bytes and client context match across worlds, while the runner selects the network destination outside that context. Each action requests the same terminal operation. Reward is one exactly when HTTP status is 200 and the returned message equals the official flag.

Both actions succeed in a reachable world (Table 9). All ten prospectively frozen pairs reverse the advantage sign, and ten matched same-state controls have no flips. World and action order are counterbalanced. Independent startups establish reachability, not a logged deployment restart incident.

D.2 Census and provenance

At Cybench revision `1097a7226eb034d3821208114da38f10b8627ab1`, the universe contains all 21 task directories with a native container-launch script. Review permits three candidate mechanisms and three action pairs per mechanism, with a 90-minute task limit.

Confirmed witnesses vary a key at a fixed external cache path (CHUNKY), the flag's startup-randomized path (LABYRINTH LINGUIST), or the startup key (LOCKTALK). The first two predate the census within this project: prospective discovery is 1/19, alongside nine bounded judgments and nine unresolved tasks. The 134/134 reversals and 0/50 control flips measure repeatability, not prevalence or independent-team replication. Cybench supplies the implementations and rewards (Zhang et al., 2025); the paired measurements are this project's work.

The nine source judgments concern no candidate mutable state on the reviewed reward path (five tasks), reward-invariant apparent state (three), or termination before another action (one). They are bounded judgments, not sufficiency proofs.

Unresolved tasks. Freshness review moves QuickScan, eval-me, algorithm-multitool, network-tools and text-sender from revelation-based exclusions to unresolved: a historical value need not identify the current instance. The other unresolved tasks are ezmaze, noisy-crc, noisier-crc and robust-cbc. They lack an identical-context comparison or, for robust-cbc's consumed connection state, repeatable counterfactuals. The detailed ledger and original dispositions remain archived.

The two source passes are audit functions. Separately, two human reviewers independently examined all 21 task records. Comparing their task-level judgments yielded 21/21 agreements and 0/21 disagreements. This agreement does not strengthen bounded judgments into task-wide certificates.

Runtime support. Using LangGraph 1.2.11 and SQLite saver 3.1.1, all 120 checks (four per context) pass with stable same-world controls. A checkpoint-namespace omission was corrected before the replacement run; both runs remain archived. These checks concern the declared saved-state boundary.

Table 4: Repeated adaptive recovery at the original checkpoints, by policy. Both policies use xhigh effort and the same nine outcome-selected instances, one per family, with five fresh paired continuations each (90 episodes per policy; 180 total). Budgets are 20 execution requests and 900 s per episode. All failures remain in the statistics. C/P: Complete/Partial restoration.

	Native success		Requests: mean ± SD		Paired change
Family / instance	C	P	C	P	P − C [95% interval]
GPT-5.6-sol; 45 pairs					
Populate playback queue `396c5a2_1`	5/5	5/5	4.00 ± 0.00	5.80 ± 1.10	$+1.80$ $[0.80, 2.60]$
Payment-request reminders `4fab96f_1`	5/5	0/5	3.20 ± 0.84	3.60 ± 0.89	$+0.40$ $[-0.40, 1.00]$
Create Venmo transaction `530b157_1`	5/5	5/5	4.80 ± 1.30	7.80 ± 1.92	$+3.00$ $[1.40, 5.00]$
Like songs `57c3486_2`	5/5	5/5	4.40 ± 1.14	6.80 ± 2.28	$+2.40$ $[0.40, 4.40]$
Create playlist `6171bbc_1`	5/5	5/5	6.40 ± 1.14	6.40 ± 0.55	0.00 $[-0.80, 0.80]$
Move files `68ee2c9_1`	5/5	5/5	4.20 ± 0.84	7.00 ± 2.00	$+2.80$ $[1.40, 4.20]$
Export notes to files `6c2c621_1`	5/5	5/5	4.80 ± 0.45	6.80 ± 0.84	$+2.00$ $[1.20, 3.00]$
Spotify playback navigation `b119b1f_1`	5/5	5/5	3.40 ± 0.89	5.00 ± 0.00	$+1.60$ $[0.80, 2.00]$
Follow artists `d4e9306_1`	5/5	5/5	4.00 ± 1.00	6.40 ± 1.34	$+2.40$ $[1.20, 3.40]$
All nine families	45/45	40/45	4.36	6.18	$+1.82$ $[1.42, 2.24]$
GPT-5.5; 45 pairs					
Populate playback queue `396c5a2_1`	5/5	5/5	6.00 ± 2.00	7.00 ± 1.87	$+1.00$ $[-0.80, 3.20]$
Payment-request reminders `4fab96f_1`	5/5	0/5	7.40 ± 1.95	7.20 ± 1.92	-0.20 $[-2.80, 2.60]$
Create Venmo transaction `530b157_1`	5/5	5/5	6.40 ± 2.51	9.80 ± 1.30	$+3.40$ $[2.20, 4.60]$
Like songs `57c3486_2`	5/5	5/5	5.20 ± 1.30	4.60 ± 0.89	-0.60 $[-2.20, 1.00]$
Create playlist `6171bbc_1`	5/5	5/5	6.80 ± 0.84	6.80 ± 1.92	0.00 $[-1.20, 1.00]$
Move files `68ee2c9_1`	5/5	3/5	8.00 ± 2.45	9.20 ± 1.30	$+1.20$ $[-1.00, 3.00]$
Export notes to files `6c2c621_1`	5/5	5/5	5.40 ± 1.67	6.60 ± 2.30	$+1.20$ $[-2.00, 4.00]$
Spotify playback navigation `b119b1f_1`	5/5	5/5	3.20 ± 1.10	5.00 ± 1.58	$+1.80$ $[0.00, 3.60]$
Follow artists `d4e9306_1`	5/5	5/5	4.40 ± 1.14	4.40 ± 0.89	0.00 $[-1.20, 1.60]$
All nine families	45/45	38/45	5.87	6.73	$+0.87$ $[0.20, 1.53]$

Notes. Requests address the Python tool and may batch native API calls; request counts equal dispatched executions. SD is the sample standard deviation over five continuations. Aggregate rows report total successes and family-balanced request means. The last column gives paired percentile bootstrap intervals: exact 5^5 resamples per family and 10,000 stratified resamples holding families fixed for each mean. These are conditional within-campaign intervals; family intervals are unadjusted, not simultaneous. Sol has 36 increases, five ties and four decreases; GPT-5.5 has 25, six and 14, respectively, with five positive, two zero and two negative family means. All per-run request counts and paired differences, with observed ranges, are retained in the numerical supplement. Policies are analyzed separately; neither estimates deployment prevalence.

Table 5: Separate Sol-only follow-up at advanced terminal boundaries. GPT-5.6-sol (xhigh) has five fresh paired continuations on each of two selected instances (20 episodes), with the same 20-request/900 s budget. C/P: Complete/Partial restoration.

	Native success		Requests: mean ± SD		Paired change
Family / instance	C	P	C	P	P − C [95% interval]
Create Venmo transaction `530b157_1`	5/5	5/5	3.40 ± 0.89	7.80 ± 1.10	$+4.40$ $[3.20, 5.40]$
Create playlist `6171bbc_1`	5/5	5/5	4.60 ± 0.89	7.00 ± 0.71	$+2.40$ $[1.60, 3.40]$
Both families	10/10	10/10	4.00	7.40	$+3.40$ $[2.60, 4.10]$

Notes. SD, aggregate rows, and conditional paired intervals use the definitions in Table 4. At the original boundaries, these same two Sol instances have means 5.60 (C) and 7.10 (P), a paired difference of $+1.50$. The follow-up changes both visible history and omitted effects; it is not pooled with any primary system cohort or the 18-episode pilot. Native success alone does not establish effect validation before completion.

Table 6: Repeated adaptive recovery for Claude Opus 5 (xhigh) at the same nine original checkpoints. Five Complete/Partial pairs per instance give 90 episodes. Budgets are 20 execution requests and 900 s; failures remain in all statistics. C/P: Complete/Partial restoration.

	Native success		Requests: mean ± SD		Paired change
Family / instance	C	P	C	P	P − C [95% interval]
Populate playback queue `396c5a2_1`	5/5	5/5	4.60 ± 1.82	5.20 ± 1.30	$+0.60$ $[0.00, 1.40]$
Payment-request reminders `4fab96f_1`	5/5	0/5	5.00 ± 1.00	6.00 ± 2.35	$+1.00$ $[-1.00, 3.20]$
Create Venmo transaction `530b157_1`	5/5	5/5	4.80 ± 1.10	8.00 ± 1.73	$+3.20$ $[2.00, 4.20]$
Like songs `57c3486_2`	5/5	5/5	5.00 ± 1.41	5.40 ± 1.14	$+0.40$ $[-0.40, 1.00]$
Create playlist `6171bbc_1`	5/5	5/5	6.00 ± 2.00	7.40 ± 3.58	$+1.40$ $[0.00, 3.80]$
Move files `68ee2c9_1`	5/5	5/5	3.40 ± 0.89	6.80 ± 2.68	$+3.40$ $[2.00, 5.80]$
Export notes to files `6c2c621_1`	5/5	5/5	3.60 ± 0.89	5.60 ± 3.05	$+2.00$ $[0.00, 4.60]$
Spotify playback navigation `b119b1f_1`	5/5	5/5	2.20 ± 0.45	4.40 ± 0.89	$+2.20$ $[2.00, 2.60]$
Follow artists `d4e9306_1`	5/5	5/5	3.40 ± 0.89	4.00 ± 1.41	$+0.60$ $[0.00, 1.40]$
All nine families	45/45	40/45	4.22	5.87	$+1.64$ $[1.16, 2.18]$

Notes. Definitions match Table 4. Family intervals enumerate all 5^5 paired bootstrap resamples; the aggregate uses 10,000 stratified resamples. Whole-round resampling gives $[1.00, 2.60]$. There are 31 positive, ten zero and four negative paired differences (range $[-2, 8]$); all nine family means are positive. The campaign retains 58 valid pre-interruption results and adds 32 episodes after reauthentication; no valid result is replaced. Provider settings and this transition are detailed in Appendix C. These are separate system-level estimates, not a pooled or model-only comparison.

Table 7: Semantic audit of all 90 Claude continuations. Each arm contains five episodes on each of the nine selected instances. Counts describe executed behavior, separately from native task outcomes.

Episode-level measure	Complete	Partial
Successful target-bound effect observation	40/45	40/45
Full family-effect coverage	23/45	28/45
Partial family-effect coverage	17/45	12/45
No effect observation	5/45	5/45
Omitted-effect repair	0/45	40/45
Repeated retained-effect execution	2/45	0/45
Registered completion	45/45	45/45
Native success	45/45	40/45

Notes. A successful observation can establish presence or absence; it need not establish the entire effect. Full coverage concerns the family-specific predicate, including its preservation requirements, not every global native-evaluator constraint. All ten no-observation cases concern reminders. The two repeated-effect cases are Complete reminder episodes; original remaining-suffix work is excluded from repair counts. The AI-assisted post-run audit uses code, outputs and native events rather than API names alone. Its labels are separate from the earlier candidate-query audit and do not measure independent-human agreement.

Table 8: Recorded effect-query absence before confirmed completion, by policy and audited scope. GPT-5.6-sol covers seven original terminal-checkpoint families (35 episodes per arm); its reminder rows are a subset. GPT-5.5 covers only the reminder instance (five episodes per arm). Each model has five fresh continuations per instance and arm. Counts concern candidate queries in the recorded native trace; they do not measure successful effect validation.

		Recorded trace	Conservative audit	
Scope	Restoration	Absent / all	Absent / assessable	Unknown
GPT-5.6-sol				
Seven terminal families	Complete	6/35	5/31	4
	Partial	5/35	5/33	2
Reminder subset	Complete	5/5	4/4	1
	Partial	5/5	5/5	0
GPT-5.5				
Reminder only	Complete	4/5	4/4	1
	Partial	5/5	4/4	1

Notes. Absent means no recorded candidate effect query before completion. Recorded fractions use all episodes; conservative fractions use structurally assessable episodes. The stricter rule excludes episodes with unresolved code structures that prevent certifying log observability; excluded episodes remain unknown. In each row, assessable plus unknown accounts for all episodes. Both GPT-5.5 unknowns are flagged for indirect namespace access. These flags do not establish suppression. A candidate query need not succeed, address the correct target, or cover every required effect. Matching query absence does not establish matching full trajectories. For each model, the reminder instance has native success 5/5 under Complete restoration and 0/5 under Partial restoration.

Table 9: One LOCKTALK witness, shown without credential bytes. The restored context z and both candidate requests are identical across rows. The rows correspond to independently started official instances. This pattern holds in all ten prospectively specified pairs.

World	Active signing key	$V(w, a^+)$	$V(w, a^-)$	$\Delta(w)$
w^+	K_+	1	0	$+1$
w^-	K_-	0	1	-1