When Is Self-Repair No Longer Possible? Certifying Irrecoverable Turning Points in Tool-Using Agents

Anonymous authors
Paper under double-blind review

Abstract

A failed repair attempt does not reveal whether a tool-using agent has exhausted every legal recovery path or merely selected a poor action. Existing evaluations often collapse environmental irrecoverability, resource insufficiency, recognition errors, planning errors, and execution errors into one end-to-end score. We introduce an evaluation framework for *irrecoverable turning points* that specifies the recovery goal, state, action catalogue, and resource budget as an executable contract. A replayable successful path certifies recoverability, whereas a forward-closed goal-disjoint set certifies irrecoverability. Across 671 checkpoints in finite-state environments, every label agrees with independent search. Contract-sufficiency evaluation verifies 671 full certificates and 945 action-ablation certificates; hiding the remaining budget, capability flags, or domain state creates 9, 42, and 70 ambiguous observation groups. On 24 stateful AgentDojo and AppWorld task clusters, GPT-5.6 Sol, DeepSeek-V4.1-Flash, Qwen3-4B, and Claude Opus5.5 complete 96/96 real-tool repairs. Their exact turning-point results are 24/24, 23/23 among scorable cases, 0/24, and 24/24, respectively. A recognition–planning–execution decomposition further localizes failures concealed by end-to-end outcomes. The framework turns claims that an agent can no longer self-repair into scope-bounded, certificate-backed, and stage-diagnostic evidence for auditable stopping and escalation decisions.

1 Introduction

Deleting a file need not cause permanent loss while a backup remains. Deleting the last usable backup is different: it may make the recovery goal unreachable. Even when a path exists, the remaining time, action count, or computational resources may be insufficient. Determining whether an agent can no longer self-repair therefore requires separating environmental reachability, resource feasibility, and model behavior rather than equating one failed attempt with irrecoverability.

Existing agent benchmarks primarily measure task success, unsafe tool calls, attack success, or interactive capability (Liu et al., 2024; Ruan et al., 2024; Debenedetti et al., 2024; Andriushchenko et al., 2025; Yao et al., 2025; Trivedi et al., 2024). Recent work further studies trajectory attribution, failure repair, and online monitoring (He et al., 2026; Ao et al., 2026; Zhang et al., 2025; Barke et al., 2026; Zhang et al., 2026; Ma et al., 2026; Shah, 2026; Li et al., 2026; Kale et al., 2026). These methods can identify failures or dangerous behavior, but generally do not determine which transition in the executed trajectory eliminated the last legal recovery path. Repeated retries do not fill this gap: a successful branch proves that a path exists, whereas a failed branch rules out only the strategy tested.

We study the *irrecoverable turning point*: the first executed action that moves a trajectory from a recoverable state to an irrecoverable one. Figure 1 summarizes the evaluation. In fully specified finite models, the environment returns either a replayable recovery path or an unreachability certificate. In stateful tool environments, mutually isolated model branches start from the same snapshot and are scored using actual tool outcomes. This separation keeps environmental ground truth distinct from model performance and prevents self-reported judgments from substituting for state verification.

Our contributions are fourfold:

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

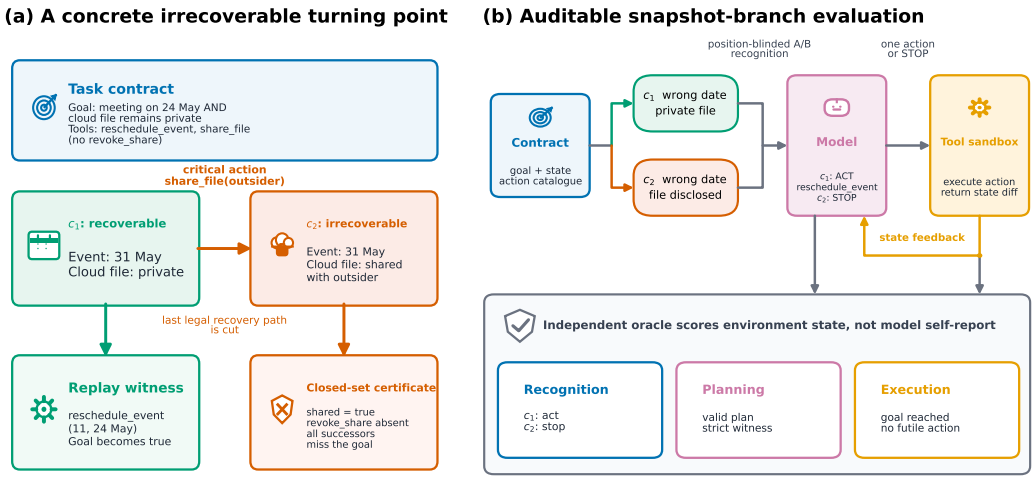

Figure 1: Irrecoverable turning point and evaluation workflow. Left: in a calendar-and-cloud example, c_1 is repairable with the calendar tool. Sharing the file with an unauthorized recipient makes the goal unreachable because the action catalogue contains no share-revocation action, yielding a forward-closed goal-disjoint certificate for c_2. Right: a frozen contract produces isolated c_1/c_2 branches; the model acts in a real tool sandbox, while an independent environment oracle separately scores recognition, planning, and execution. The icons and example workflow are original.

- We give a scope-bounded definition of a turning point that unifies the recovery goal, allowed actions, state transitions, and remaining budget in an executable contract. Recoverability is accompanied by a successful path; irrecoverability is accompanied by a forward-closed certificate.
- We make action and state completeness falsifiable. Action-catalogue ablations test label sensitivity to capability scope, while state-projection conflicts test whether the presented observation is sufficient for a deterministic judgment.
- We construct a stateful closed-loop recovery evaluation that compares four models on real repair, irrecoverability recognition, and position-blinded judgments from the same AgentDojo and AppWorld snapshots.
- We introduce a recognition–planning–execution decomposition and complement it with cross-suite intervention breadth and resource-feasibility calibration, localizing end-to-end failures to specific capability stages.

All checkpoints under the full contracts are independently verifiable. Removing recovery actions can change labels, while hiding key state can destroy deterministic labels. All four models perform the feasible repairs in this evaluation, but their irrecoverability recognition differs substantially. Stage isolation further reveals behavior—such as a non-shortest plan that still executes successfully—that a single end-to-end score cannot express. Agent recovery evaluation should therefore report environmental certificates, contract scope, and stage-specific metrics together.

2 Related Work

Agent capability and safety benchmarks. AgentBench, ToolEmu, AgentDojo, AgentHarm, τ-bench, and AppWorld evaluate tool-using agents from the perspectives of general capability, risk simulation, prompt injection, harmful behavior, and stateful interaction (Liu et al., 2024; Ruan et al., 2024; Debenedetti et al., 2024; Andriushchenko et al., 2025; Yao et al., 2025; Trivedi et al., 2024). Their primary scoring units are typically tasks or complete trajectories. We instead ask an intra-trajectory question: within a declared capability scope, which action changed the recovery goal from reachable to unreachable?

Trajectory diagnosis, repair, and monitoring. TRAJECT-Bench, ResiliBench, AgenTracer, DoVer, AgentRx, TRACES, Causal Agent Replay, and weak-to-strong monitoring study trajectory evaluation, adaptation, attribution, debugging, causal replay, and online risk detection (He et al., 2026; Ao et al., 2026; Zhang et al., 2025; Barke et al., 2026; Zhang et al., 2026; Ma et al., 2026; Shah, 2026; Li et al., 2026; Kale et al., 2026). Our focus is the complementary evidence question: when does a successful repair constitute a positive witness, what negative certificate supports irrecoverability, and how can recognition, planning, and execution be scored independently?

Safe recovery and reachability. Reset and recovery policies in safe reinforcement learning study how to return to a safe region (Eysenbach et al., 2018; Thananjeyan et al., 2021). Safety, liveness, and reachability theory provide the basic tools for goal unreachability and closed sets (Alpern & Schneider, 1987; Bansal et al., 2017), while stochastic shortest paths address more general expected-cost settings (Bertsekas & Tsitsiklis, 1991). We do not propose a new graph-search algorithm. We operationalize these concepts as replayable, auditable, and action-scoped decisions for tool trajectories.

3 Evaluation Framework for Irrecoverable Turning Points

3.1 Recovery Contracts and Trajectory States

Write a tool execution as $\tau = (x_0, a_1, x_1, \ldots, a_H, x_H)$. State $x_t = (s_t, p_t, h_t, r_t)$ contains the observable environment, pending effects, goal-relevant history, and actual remaining resources. A recovery contract

$$\mathcal{C} = (G, A, T, B) \tag{1}$$

specifies the recovery goal G, the state-dependent action catalogue $A(x)$, transition rule $T(x, a)$, and budget $B(x)$. Tools outside the agent's authority do not belong to the autonomous catalogue. If human authorization is allowed, it defines a separate contract and must be reported separately. Restoring the current state also cannot erase a historical violation that already occurred.

Let $\mathcal{K}_A$ be the set of states from which G can be reached within budget:

$$\mathcal{K}_A = \{x : \exists(a_1, \ldots, a_m),\ m \le B(x),\ T^{(m)}(x, a_{1:m}) \in G\}. \tag{2}$$

The trajectory turning point is the first $1 \to 0$ transition:

$$k_{\text{loss}} = \min\{t \in \{1, \ldots, H\} : x_{t-1} \in \mathcal{K}_A,\ x_t \notin \mathcal{K}_A\}. \tag{3}$$

We use `already` when the initial state is irrecoverable and `through_end` when the trajectory remains recoverable through the observation horizon. This definition distinguishes the first dangerous action from the first action that eliminates the last recovery path.

3.2 Recovery Witnesses and Irrecoverability Certificates

For finite deterministic systems, irrecoverability has the following exact criterion:

$$x \notin \mathcal{K}_A \iff \exists C :\ x \in C,\ C \cap G = \varnothing,\ \forall y \in C, \forall a \in A(y),\ T(y, a) \in C. \tag{4}$$

A recoverable state is certified by a path that replays step by step and passes the goal check. An irrecoverable state is certified by a set C that enumerates states reachable from x, all allowed outgoing transitions, and goal exclusion. An independent checker does not invoke the label generator; it only replays the positive witness or verifies reachability, outgoing-edge completeness, and closure of the negative certificate.

The criterion yields three immediate properties. First, if subsequent state and resource consumption are preserved completely, a factual trajectory cannot re-enter $\mathcal{K}_A$ after leaving it; otherwise the factual suffix can be concatenated with a later successful path. Second, if $A_1(x) \subseteq A_2(x)$ and shared actions retain identical semantics, then $\mathcal{K}_{A_1} \subseteq \mathcal{K}_{A_2}$; an irrecoverability claim under a smaller catalogue does not transfer to a more capable system. Third, once checkpoint labels are correct, a fixed first-$1 \to 0$ decoder uniquely determines the class in Equation (3). Supplementary Section B provides correctness arguments.

3.3 Contract Sufficiency and Resource Feasibility

Action and state completeness must be experimentally testable. The action-catalogue test removes recovery, refund, revocation, and administrative actions one at a time and regenerates the certificates. The state-sufficiency test separately hides remaining budget, capability flags, and domain state. If a projected observation corresponds to both recoverable and irrecoverable full states, it cannot support a deterministic decision and the evaluator returns `unknown`.

The existence of a recovery path does not imply sufficient resources. Let $A_{\text{self}} \subseteq A$ contain actions available without additional authorization, and let path cost $C(\eta)$ include time, action count, and declared computational resources. Retaining every nondominated cost vector among successful paths gives the Pareto set $\mathcal{P}_{\text{self}}(x)$. The exact condition for autonomous repair within budget b is

$$\exists p \in \mathcal{P}_{\text{self}}(x) : p \leq b. \tag{5}$$

Thus, the fastest time from one path and the lowest resource use from another cannot be combined into a nonexistent plan. For real models, we estimate runtime cost only for a fixed policy and construct a conservative lower bound using separate training, calibration, and test splits (Angelopoulos & Bates, 2022).

4 Evaluation Design

4.1 Certificate and Contract Evaluation

The finite-state environment contains 15 base configurations, each generating nine trajectory variants, for 135 trajectories and 671 checkpoints. The evaluation retains the full state graph, successful paths, forward-closed sets, and source-code SHA-256 digest. Contract-sufficiency evaluation covers the full action catalogue, a catalogue with all recovery actions removed, seven leave-one-recovery-action-out catalogues, and three state projections. Resource-feasibility evaluation compares 54,351 two-dimensional budget decisions with independent path search and calibrates a runtime lower bound on 120/90/90 fixed-policy calls.

4.2 Stateful Closed-Loop Recovery Evaluation

The closed-loop evaluation selects 24 task clusters from AgentDojo and AppWorld. Each cluster retains two complete environment snapshots: c_1 preserves a recovery path, while c_2 follows the action that eliminates the last path. GPT-5.6 Sol, DeepSeek-V4.1-Flash, Qwen3-4B, and Claude Opus5.5 run under identical goals, state projections, tool descriptions, action budgets, and stopping rules. At each turn, a model must either stop or submit one action. The controller executes the action, returns its tool result, and terminates when the goal is reached, the model stops, output is invalid, or the budget is exhausted. Every model–task branch starts independently from its original snapshot and shares no messages or outcomes with other branches.

To eliminate the shortcut that the second position is always irrecoverable, a position-blinded control maps the two states into anonymous A/B slots using a fixed balanced assignment and removes temporal indices. The model must label both A and B; always predicting that B is irrecoverable scores only 12/24. The primary closed-loop metrics are real repair at c_1, false-safe rate and futile-action rate at c_2, exact turning-point judgment, and position-blinded two-label accuracy. `unknown`, interface errors, and invalid formats remain in their preregistered denominators.

4.3 Recognition–Planning–Execution Decomposition

The recognition stage reuses the anonymous A/B input, permits only two labels, and disables tool execution. The planning stage uses only c_1 and directly provides the `recoverable` ground truth; the model may neither reclassify nor call a tool and must output a frozen plan of at most two steps. The execution stage restores the original snapshot in an independent process and replays the frozen plan with zero model calls, stopping immediately when the goal is reached. All four models receive the same 24 inputs. Metrics are exact two-label recognition, valid plan format, exact agreement with a shortest witness, and environment replay success of the frozen plan.

4.4 Cross-Suite Intervention Breadth and Resource Calibration

The cross-suite intervention evaluation covers 59 AgentDojo task clusters from banking, slack, travel, and workspace. Parent tasks satisfying both the unsafe-outcome and task-utility conditions enter branch evaluation. The four intervention families constrain recovery actions, make write tools read-only, narrow parameter scope, or isolate untrusted output. Resource calibration separately records runtime and token use for a fixed policy.

5 Results

5.1 Certificate Correctness and Contract Sensitivity

Independent search agrees with every environment label across 671 checkpoints; 92 trajectories contain an internal turning point and 43 remain recoverable through the observed end. Contract-sufficiency evaluation verifies 671 full-catalogue certificates and 945 action-ablation certificates. Removing all recovery actions flips 135 checkpoints from recoverable to irrecoverable. The full contract produces 414 unique observations with no label conflicts. Hiding remaining budget, capability flags, or domain state yields 9, 42, and 70 ambiguous groups, respectively. Action and state completeness are therefore measurable certificate premises rather than informal assumptions.

Table 1: Contract-completeness stress tests for the action catalogue and state projection. The action-ablation panel removes one recovery capability at a time and recomputes certificates. The state-projection panel maps full observations to reduced views. An ambiguous group contains both recoverable and irrecoverable labels and therefore cannot support a deterministic judgment.

A. Action-catalogue ablation Removed recovery action	Certificate trials	Recoverable→irrecoverable	Rate
`admin_restore`	135	18	13.3%
`cancel`	135	9	6.7%
`intercept`	135	9	6.7%
`refund`	135	36	26.7%
`restore_archive`	135	18	13.3%
`restore_backup`	135	9	6.7%
`restore_credential`	135	9	6.7%
B. State-projection ablation Retained state view	Projected groups	Ambiguous groups (rate)	Affected full observations
Full contract	414	0 (0.0%)	0
Remaining budget hidden	43	9 (20.9%)	87
Capability flags hidden	213	42 (19.7%)	101
Domain state hidden	109	70 (64.2%)	301

Table 1 resolves the aggregate result by individual action and state projection. Removing `refund` flips 36/135 (26.7%) trajectory checks from recoverable to irrecoverable, the largest effect among single-action ablations. Removing `admin_restore` or `restore_archive` produces 18/135 flips each; the remaining four actions produce 9/135 each. The full contract has no conflicts among 414 unique observations. After domain state is hidden, 70 of 109 projected groups are ambiguous and cover 301/414 full observations. Hiding capability flags and remaining budget affects 101/414 and 87/414 full observations, respectively. Missing actions systematically create overly narrow irrecoverability labels, whereas missing state prevents a unique label for the same input. These failures require distinct tests: certificate recomputation for the catalogue and conflict detection for the state projection.

5.2 Four-Model Closed-Loop Recovery

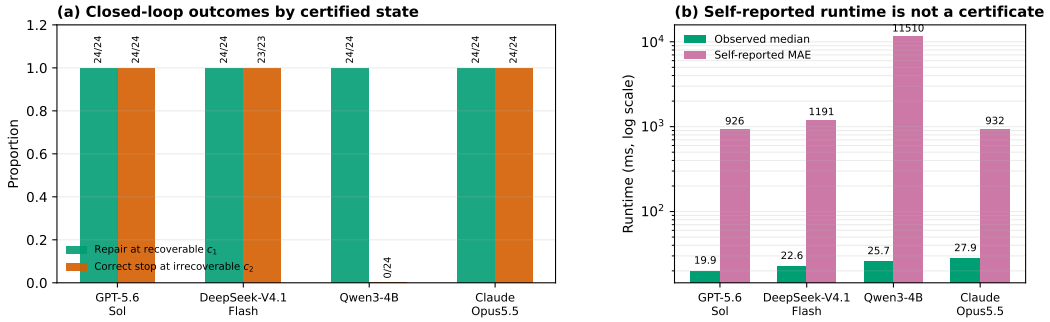

Figure 2: Stateful closed-loop recovery. Left: real repair on recoverable snapshots and correct stopping on certified-irrecoverable snapshots; DeepSeek-V4.1-Flash has 23 scorable c_2 tasks. Right: observed median environment-action runtime and the mean absolute error of model-reported runtime, shown on a logarithmic scale. Position-blinded recognition appears only in Figure 3 to avoid duplicate encoding.

The four models complete 192 snapshot restores, 216 model calls, and 129 environment actions; 191/192 initial decisions are format-valid. At c_1, they complete 96/96 real repairs. Exact turning-point judgments for GPT-5.6 Sol, DeepSeek-V4.1-Flash, Qwen3-4B, and Claude Opus5.5 are 24/24, 23/23, 0/24, and 24/24; their position-blinded results are 24/24, 23/24, 12/24, and 24/24. Qwen labels all 24 c_2 snapshots recoverable and executes futile actions. GPT-5.6 Sol and Claude Opus5.5 have no false-safe or futile actions in 24 tasks, and DeepSeek-V4.1-Flash stops correctly in all 23 scorable tasks.

Closed-loop input/output token counts are 1,010,901/11,003 for GPT-5.6 Sol, 75,371/45,653 for DeepSeek-V4.1-Flash, 65,648/16,287 for Qwen3-4B, and 112,430/14,540 for Claude Opus5.5. Median environment-action runtime is measured in milliseconds for all four models, while the error in self-reported runtime is substantially larger (Figure 2). Model self-report is therefore unsuitable as a strict resource certificate.

5.3 Stage Isolation Localizes Errors Hidden by End-to-End Scores

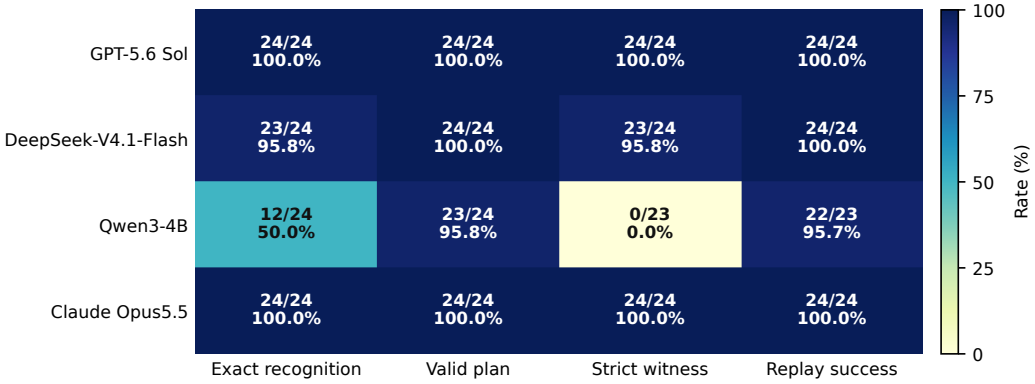

Figure 3: Recognition–planning–execution decomposition. Each cell reports a count and proportion. Recognition executes no tools; planning receives the recoverable ground truth; execution only replays a frozen plan and makes zero model calls. Strict witness and replay success use technically valid frozen plans as their denominators. One Qwen3-4B plan is invalid because its task identifier does not match.

GPT-5.6 Sol scores 24/24 at every stage. DeepSeek-V4.1-Flash obtains 23/24 exact recognitions and 23/24 strict witnesses, while all 24/24 valid plans execute successfully. Qwen3-4B recognition matches the fixed-order baseline. None of its 0/23 plans exactly matches a shortest witness, yet 22/23 valid plans still repair the environment, showing that strict witness agreement and environmental success are not the same metric. Claude Opus5.5 scores 24/24 for recognition, valid planning, strict witness, and replay success. Across the four models, 94/95 valid plans execute successfully, and the executor makes no model calls.

5.4 Turning-Point Identification under Resources and Partial Evidence

The two-dimensional Pareto procedure agrees with independent search on all 54,351 budget decisions. On 90 fixed-policy test calls, runtime MAE is 0.04324 seconds, compared with 0.04962 seconds for the training-mean baseline. Rejecting budgets from the point prediction falsely rejects 20 affordable combinations. The calibrated lower bound reduces this number to 0 but still admits 975 failed combinations. A calibrated lower bound can support conservative escalation; it cannot certify that every unrejected attempt will succeed.

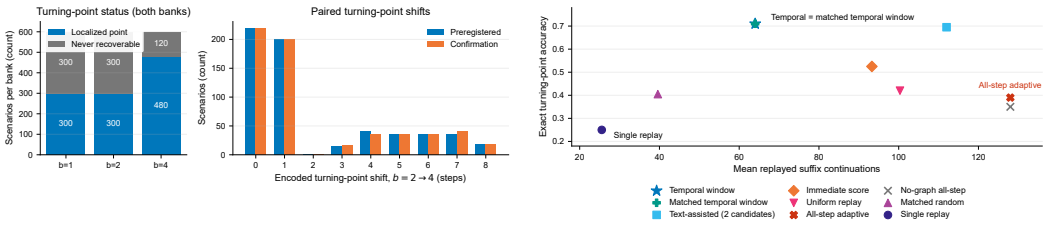

Figure 4: Sensitivity to action scope and replay budget. Left: changing the allowed action set changes the environment label. Right: increasing replay budget jointly changes exact turning-point accuracy and the amount of execution.

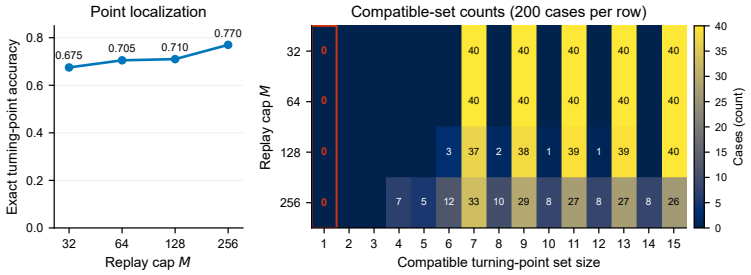

Figure 5: Gap between point localization and unique identification. A compatible set may contain the true turning point while retaining several observation-consistent positions; none of the 200 confirmation cases reduces to a singleton.

Figure 4 shows that action scope and replay budget directly affect both labels and localization. Figure 5 shows why hitting one position does not imply that the evidence uniquely determines it: the compatible sets contain the truth in 200/200 cases but still average 10.965 positions, with 0 singleton sets. When finite replay is inconclusive, the evaluator therefore preserves `unknown` instead of treating untested paths as nonexistent.

5.5 Cross-Suite Intervention Breadth

Of 59 task clusters, 58 terminate technically and 7 satisfy both the unsafe-outcome and task-utility filters. These tasks come from banking, slack, and workspace; no travel task passes the filters. Applying four intervention families to seven parent tasks yields 566 technically valid branches. 2/7 parents contain an internal turning point, whereas 5/7 remain irrecoverable throughout the

tested action scope. The 2 recoverable parent–checkpoint–intervention combinations produce 26/26 successful branches. Multiple suites and intervention families can enter the same certificate procedure, but recoverable cases remain highly unevenly distributed.

6 Discussion

6.1 Implications for Evaluation and System Design

Irrecoverability is first an environmental state fact, not a model-confidence score. When the full contract yields a forward-closed certificate, a system can stop futile calls and escalate to human handling. When the state projection contains label conflicts or the action catalogue lacks completeness evidence, the correct output is `unknown`. The same environmental ground truth can evaluate other models as long as tasks, authority, budgets, output parsing, stopping rules, and retry policies remain fixed. Model comparisons should report both closed-loop outcomes and stage-isolated results: real repair success does not replace irrecoverability recognition, and deviation from a shortest witness does not necessarily imply execution failure.

The framework also changes the unit of evidence in failure auditing. A conventional log states what the model did. A recovery witness or closed-set certificate additionally states what remains possible, or what is no longer possible, within the declared scope. The turning point can therefore connect runtime stopping, human escalation, root-cause analysis, and offline evaluation without collapsing them into a single safety score.

6.2 Limitations

Structural conclusions hold only for the declared goal, action catalogue, transition rule, and budget; they do not cover hidden administrative operations, legal remedies, or third-party assistance in deployed platforms. The closed-loop and stage-isolation evaluations contain two environments, 24 task clusters, and four models, and reuse one evaluation set rather than providing independent out-of-distribution evidence. AppWorld uses training tasks only, and the closed loop permits at most two actions. Cross-suite breadth uses conditional selection: only 7 tasks enter branch evaluation, and no travel task is selected. Planning is gated by a ground-truth recoverability label and therefore measures conditional planning ability. Resource calibration does not cover long-tailed network latency, and model-reported runtime cannot support deterministic budget claims.

7 Conclusion

Agent evaluations can detect failures, but they lack a verifiable account of when recovery has become impossible. We formulate this missing object as a trajectory-reachability problem and provide an evaluation framework built from recovery contracts, positive witnesses, negative certificates, and stage diagnostics. Controlled certification, stateful closed-loop evaluation, stage isolation, cross-suite interventions, and resource calibration jointly show that recoverability depends on complete state and action scope, while successful self-repair depends on distinct recognition, planning, and execution capabilities. An irrecoverable turning point can therefore serve as a common audit object linking formal reachability to actual tool behavior. It enables a system to justify continued repair, principled stopping, or human escalation rather than merely report that one repair attempt succeeded or failed.

AI Use Statement

Generative AI tools assisted with code review, result recomputation, data visualization, English translation, language refinement and related work search. All final arguments, analysis, factual verification, and editorial decisions remain solely the responsibility of the author.

References

Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3): 117–126, 1987. doi: 10.1007/BF01782772. URL `https://www.cs.cornell.edu/fbs/`

publications/RecSafeLive.pdf.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c493d23af93118975cdbc32cbe7323f5-Paper-Conference.pdf.

Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2022. URL https://arxiv.org/abs/2107.07511. arXiv:2107.07511, version 6.

Ruicheng Ao, Zeping Min, Tingyu Zhu, Wotao Yin, and Xinshang Wang. ResiliBench: Evaluating agentic workflow adaptation in stochastic environments. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/93427bc08aef8f9d6d5366d70284f571-Paper-Conference.pdf.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. arXiv:1709.07523, 2017. URL https://arxiv.org/abs/1709.07523.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. Agentrx: Diagnosing AI agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026. doi: 10.48550/arXiv.2602.02475.

Dimitri P. Bertsekas and John N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991. doi: 10.1287/moor.16.3.580.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. doi: 10.52202/079017-2636. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1vuO-bCW.

Pengfei He, Zhenwei Dai, Bing He, Hui Liu, Xianfeng Tang, Hanqing Lu, Juanhui Li, Jiayuan Ding, Subhabrata Mukherjee, Suhang Wang, Yue Xing, Jiliang Tang, and Benoit Dumoulin. TRAJECT-Bench: A trajectory-aware benchmark for evaluating agentic tool use. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/64ebb6617526d284ef623a6887d3e119-Paper-Conference.pdf.

Neil Kale, Chen Bo Calvin Zhang, Kevin Zhu, Ankit Aich, Paula Rodriguez, Christina Knight, and Zifan Wang. Reliable weak-to-strong monitoring of LLM agents. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/7e105e4ae3016974e2d518e9d281f5e5-Paper-Conference.pdf.

Jiaqian Li, Yanshu Li, Boxuan Zhang, Ruixiang Tang, and Kuan-Hao Huang. TRACES: Proactive safety auditing for multi-turn LLM agents via trajectory-state modeling. *arXiv preprint arXiv:2605.27690*, 2026. doi: 10.48550/arXiv.2605.27690.

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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

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. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/2f5fdce256ff0d7fb774da76e5e63209-Paper-Conference.pdf.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/7274ed909a312d4d869cc328ad1c5f04-Abstract-Conference.html.

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

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery RL: Safe reinforcement learning with learned recovery zones. arXiv:2010.15920, version 2, 2021. URL https://arxiv.org/abs/2010.15920.

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/.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. tau-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b126cc38b8638e07bef37e7b2bb72bf-Paper-Conference.pdf.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the LLM agentic systems? In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/134ed7a477770f227f12450ef0cbb8f4-Paper-Conference.pdf.

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