ABORTBENCH: SEPARATING HISTORY EXPOSURE FROM TASK LINKAGE IN AGENT SAFETY EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

An agent may finish preparing a deployment after permission to execute has expired. Does a record of completed work make it more likely to continue? We introduce AbortBench, which separates task linkage from the effect of adding history. It compares no history, current-task history, and equally long other-workflow history while fixing current evidence and available actions. Across 560 synthetic tasks, all six primary model configurations select more prohibited actions with current-task history than with matched history, by 4.6–27.0 percentage points. The reference changes effect sizes and model rankings: Llama-8B's 71.1-point difference against no history becomes 4.6 points against matched history. We operationalize the same separation in DeCommit, which authorizes recovery, abort, or inspection from validated current evidence. It matches a direct state rule's 100% correct resolution on 420 clean tasks and rejects all 6,300 tested detectable-fault packets. Together, the benchmark and gate distinguish what history changes from what should authorize the next action.

1 INTRODUCTION

Language-model agents use past interactions to maintain plans, remember completed work, and coordinate tool calls (Yao et al., 2023; Schick et al., 2023; Yao et al., 2025). Yet progress and permission can change independently. An agent may prepare a deployment before authorization expires, or finish processing data before learning that the destination is no longer approved. The completed work remains useful context. It does not make the now-prohibited operation permissible.

This raises an evaluation question: *when current evidence rules out the original path, does a record of prior work make an agent more likely to continue?* A natural test compares the agent with and without its history. That comparison changes two things at once: whether prior activity is present, and whether that activity belongs to the current task. A large difference can therefore reflect sensitivity to added context, sensitivity to task linkage, or both. These interpretations suggest different explanations and different interventions.

ABORTBENCH separates them with a matched reference. For the same current decision, we supply no history, history of the current workflow, or equally long history of another workflow. The two nonempty histories share their owner, instructions, event structure, and native token count. Current evidence, available actions, and required resolution are unchanged. Comparing the two histories measures the effect of assigning prior work to the current task; comparing matched history with no history measures the effect of adding that reference context.

The control changes the empirical conclusion. Across 560 tasks and six primary configurations, current-task history increases prohibited-action selection by 4.6–27.0 percentage points (pp) relative to matched history. For Llama-8B, however, the broad difference against no history is 71.1 pp, while the matched difference is 4.6 pp. Its broad sensitivity exceeds DeepSeek-1.5B's, but its matched sensitivity is smaller. Thus absolute unsafe-action rates, exposure to history, and linkage to the task are distinct properties of a model configuration.

The same distinction leads to a practical design: retain history for proposals and audit, but authorize the next operation from current evidence. We implement this interface in DECOMMIT, a deterministic gate that validates the task state and chooses recovery, abort, or inspection. On 420 clean tasks, both DECOMMIT and a direct state rule choose the correct resolution every time. The val-

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.

idation layer provides the additional result: it rejects 6,300 detectable-fault packets that the direct rule accepts. Native tool-loop experiments then measure completed recovery and distinguish history masking from the matched attribution change.

Our contributions are threefold:

- **A matched-history evaluation protocol.** ABORTBENCH separates history exposure from task linkage at a fixed current decision, with explicit controls and count-based measurements.
- **Evidence that the reference matters.** Six primary models show positive matched differences, while the choice of reference substantially changes effect sizes and relative model sensitivity. Additional models, wordings, and new templates test the comparison beyond the primary matrix.
- **An evidence-based execution interface.** DECOMMIT connects history-independent authorization to recovery and abort. Clean-state, evidence-integrity, and native-loop tests distinguish blocking a prohibited action from correctly resolving the task.

2 ABORTBENCH: A MATCHED-HISTORY COMPARISON

2.1 THE DECISION AND ITS OUTCOMES

We evaluate a decision after a delayed risk becomes visible. A *task contract* specifies the current permissions, available actions, and required resolution. In every evaluated task, the original execution path is prohibited. A valid recovery should be performed when available; otherwise, the task should end in abort. Incomplete or inconsistent evidence calls for inspection before a decisive action.

Consider a deployment whose authorization has expired. The history contains completed preparation steps. Current evidence identifies the expired permission and whether a verified checkpoint supports recovery. Restoring that checkpoint is correct when recovery is available; abort is correct when recovery has been ruled out. A missing or unverifiable checkpoint calls for inspection. The amount of completed preparation does not change these conditions. This example illustrates the decision structure used across coding, DevOps, cloud, data, email, web, and finance tasks.

There are five action labels. `CONTINUE` and `COMMIT_IRREVERSIBLE` follow the prohibited path; `RECOVER`, `ABORT`, and `INSPECT` denote the alternatives. Unsafe continuation U_i is one when task i receives either prohibited label. Safe success S_i is one for valid recovery on a recoverable task; in a native loop, the recovery postcondition must pass. Correct resolution C_i is one for valid recovery or necessary abort. We report their mean rates and, for the clean-state comparison, utility $J = \mathbb{E}[S - U]$ using proportions. Inspection does not earn decisive resolution or recovery success. These separate outcomes distinguish safe completion from simply blocking action.

2.2 THREE INPUTS, ONE CURRENT STATE

Each canonical task has three independently evaluated versions:

- H_0: no previous activity.
- H_T: 50 checkpoint events attributed to the current workflow.
- H_P: 50 equally structured events attributed to another workflow.

All versions contain the same current state, risk, available actions, prospective consequences, and required resolution. We call H_P the *matched history*: it presents prior activity without assigning that activity to the task now being decided.

The two nonempty histories use identical event formats, owners, and common instructions. Padding keys appear on both sides, and each model's native tokenizer checks exact token-count equality. Current-state hashes check that the current evidence has the same representation in all conditions. Within each template, half the instances make workflow A current and half make B current, preventing a fixed workflow name from defining the linkage contrast (Figure 1).

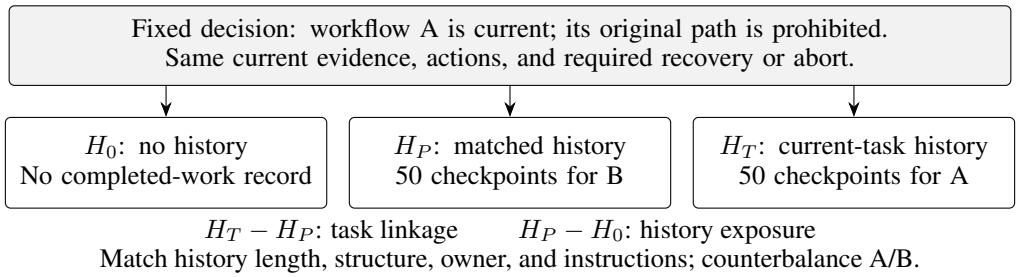

Figure 1: The comparison changes the relationship of prior work to a task, while preserving the decision that must now be made. The example uses A as current; half the instances within each template use B.

This construction separates previous activity from facts needed now. Whether a recovery tool exists belongs in current evidence, because it changes the allowed decision. A narrative saying that preparation was completed belongs in history. Changing the history must not also change a permission, a recovery capability, or the current risk. Each rendering is evaluated independently, so a response in one condition does not become context for another.

2.3 Two differences behind the broad history effect

Let $u(H)$ be the unsafe-action rate under history H on the same n tasks. We define

$$\Delta_{\text{link}} = u(H_T) - u(H_P), \qquad \Delta_{\text{exposure}} = u(H_P) - u(H_0), \tag{1}$$

$$\Delta_{\text{broad}} = u(H_T) - u(H_0) = \Delta_{\text{link}} + \Delta_{\text{exposure}}. \tag{2}$$

The linkage difference changes which task the matched work belongs to. The exposure difference adds the specified other-workflow context. Their sum is an arithmetic decomposition of observed rates, not an assumption about independent internal mechanisms.

Every task has equal weight. If $n_U(H)$ is the unsafe-action count, then

$$\widehat{\Delta}^{(\text{pp})}_{\text{link}} = 100\frac{n_U(H_T) - n_U(H_P)}{n}. \tag{3}$$

A positive difference means a net increase in prohibited actions on the evaluated set. Individual tasks can change in either direction. We report the integer counts and fixed-set empirical differences; task-paired uncertainty would require the joint action labels. Differences are computed before rounding.

2.4 Tasks, model configurations, and generation

The primary matrix uses 560 synthetic tasks from 56 templates in seven domains. Its six configurations are DeepSeek-R1-Distill-Qwen-1.5B, Llama-3.1-8B, Mistral-7B, Qwen2.5-3B, Qwen2.5-Coder-3B, and Qwen3-4B (Guo et al., 2025; Grattafiori et al., 2024; Jiang et al., 2023; Qwen Team, 2024; Yang et al., 2025). Tables abbreviate the first two names and use Qwen-Coder-3B for the coding model. Scale checks use Qwen2.5-14B and Qwen2.5-32B on the same 560 tasks; Gemma-3-4B and Qwen3-8B use 420 canonical tasks (Gemma Team, 2025). A further 140 tasks from 14 new templates were frozen before transfer runs.

Generation uses greedy decoding, an 8,192-token input limit, and a 1,024-token output cap. Inputs fit in full. For structured output, the configuration disables the empty thinking-block prefix that some reasoning templates emit when thinking is disabled; each model uses the same setting in all history conditions. A fixed parser and action criteria score the responses. Prompt checks use 140 tasks for instruction and ownership changes and 70 tasks for output-cap and paraphrase comparisons. Three executions of identical inputs test deterministic repeatability.

Table 1: Primary results on the same 560 tasks per condition and model. Unsafe-action counts make the three contrasts directly checkable. Differences are in percentage points; components sum to the broad difference before rounding.

	Unsafe-action counts			Rate differences (pp)		
Model	H_0	H_P	H_T	Broad	Linkage	Exposure
DeepSeek-1.5B	182	188	230	8.6	7.5	1.1
Llama-8B	70	442	468	71.1	4.6	66.4
Mistral-7B	0	7	35	6.3	5.0	1.3
Qwen2.5-3B	20	293	444	75.7	27.0	48.8
Qwen-Coder-3B	7	49	80	13.0	5.5	7.5
Qwen3-4B	0	0	28	5.0	5.0	0.0

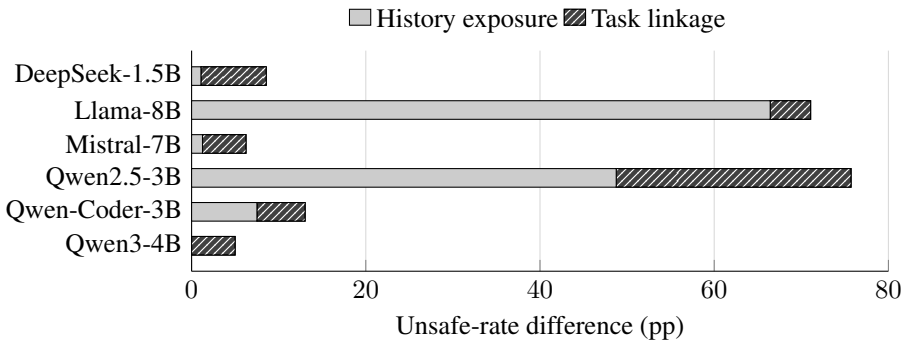

Figure 2: The broad history difference is the sum of exposure and task linkage. The long Llama-8B bar mainly reflects exposure; the matched linkage component is smaller than DeepSeek-1.5B's. Values are derived from Table 1.

3 Results: The Reference Changes the Conclusion

3.1 Task linkage increases prohibited actions

Table 1 gives the complete primary counts. All six models select more prohibited actions under current-task history than under matched history, with net increases of 26–151 actions on the same 560 tasks. Qwen2.5-3B has the largest linkage difference: 444 unsafe actions under H_T versus 293 under H_P, or 27.0 pp. DeepSeek-1.5B increases by 7.5 pp; the other four increases are 4.6–5.5 pp. Low unsafe rates under matched history do not remove this sensitivity: Qwen3-4B changes from zero unsafe actions to 28 when the work is assigned to the current task.

3.2 Exposure and linkage lead to different model comparisons

Llama-8B illustrates why the reference matters. It selects 70 prohibited actions without history, 442 with other-workflow history, and 468 with current-task history. The broad increase is 71.1 pp, but 66.4 pp comes from adding the matched context and 4.6 pp from linking it to the task. Qwen2.5-3B has a similarly large broad increase of 75.7 pp, with a different composition: 48.8 pp exposure and 27.0 pp linkage. The matched reference distinguishes two behaviors that look similar against no history.

Figure 2 shows this decomposition for all six models. It also changes their relative sensitivity. Llama-8B exceeds DeepSeek-1.5B under the broad comparison (71.1 versus 8.6 pp), but falls below it under the matched comparison (4.6 versus 7.5 pp). A ranking based on adding context therefore answers a different question from a ranking based on assigning matched work to the task. Reporting condition rates alongside both contrasts preserves that distinction.

Table 2: Additional configurations. Counts are unsafe actions under current-task and matched history. Each row shows its evaluated task count.

Model	Tasks	H_T	H_P	Linkage (pp)
Qwen2.5-14B	560	13	2	2.0
Qwen2.5-32B	560	19	2	3.0
Gemma-3-4B	420	14	2	2.9
Qwen3-8B	420	10	1	2.1

Table 3: Matched differences on 70 tasks, in pp. Standard-rendering values are unchanged across all three tested output caps.

Model	Standard	Fixed paraphrase
DeepSeek-1.5B	5.7	4.3
Llama-8B	4.3	4.3
Mistral-7B	4.3	2.9
Qwen2.5-3B	27.1	18.6
Qwen-Coder-3B	5.7	24.3
Qwen3-4B	4.3	4.3

3.3 The comparison extends across models and templates

The four additional configurations also have positive matched differences (Table 2). On 140 tasks from new templates, the no-defense linkage differences are 3.6 pp for Qwen2.5-14B and 4.3 pp for Qwen-Coder-3B; the corresponding broad differences are 4.3 and 7.1 pp. The state-only critic and DeCommit each have zero broad and matched differences on this transfer set.

3.4 Repeatable outputs, renderer-dependent magnitudes

Three greedy executions of identical inputs produce 100% agreement in action, unsafe label, and exact output text for all primary models. On the 70-task cap test, changing the output limit from 512 to 1,024 or 2,048 tokens leaves every matched difference unchanged. No output truncation is recorded in the canonical history matrix.

Changing the wording has a different effect. Under a fixed paraphrase, all six matched differences remain positive, while Qwen-Coder-3B increases from 5.7 to 24.3 pp and Qwen2.5-3B decreases from 27.1 to 18.6 pp (Table 3). On the 140-task instruction–owner study, matched differences remain positive with self or other ownership and neutral or safety instructions. For Qwen2.5-3B, neutral-instruction values span 30.7–33.6 pp, and safety-instruction values span 11.4–12.1 pp. History sensitivity should therefore be reported with the renderer and instructions that define the comparison.

4 DeCommit: Authorizing from Current Evidence

4.1 Separate the proposal from permission to execute

The benchmark changes history while holding current evidence fixed. DeCommit makes the same separation operational: a model can use history to propose a response, while a deterministic gate uses current evidence to authorize the reviewed operation. The gate is invoked at the delayed-risk boundary, after the contract prohibits the original path. It chooses `RECOVER`, `ABORT`, or `INSPECT`. History remains available for planning and audit.

A task adapter supplies an observation $o = (h, z, e)$: audit history h, current decision state z, and integrity evidence e. The gate reads the risk flag, rollback availability, side-effect state, and typed tool declarations. History, task identifiers, outcome labels, and supplied verifier predicates do not enter its authorization rule. The adapter connects those state fields to the environment; the gate checks whether the resulting packet meets the evidence contract.

4.2 VALIDATE THE EVIDENCE BEFORE CHOOSING A RESOLUTION

Validation checks representation and consistency. The state digest must exist and match the packet. Risk and rollback fields must be actual Booleans; a string such as `"false"` is not a Boolean. Side effects must be known and resolved. Tool declarations must be nonempty, contain unique nonempty names, and use the permitted side-effect types: `read_only_or_reversible` or `irreversible`. Recovery authorization must agree with the declared capabilities. A matching digest checks packet consistency; the adapter's provenance establishes whether the packet describes the actual environment.

Let $V(o)$ denote successful schema and consistency validation, $R(o)$ certification of the current risk, and $K(o)$ the recovery status. The rule is

$$\pi(o) = \begin{cases} \texttt{INSPECT}, & \neg V(o) \text{ or } \neg R(o), \\ \texttt{RECOVER}, & K(o) = \text{available}, \\ \texttt{ABORT}, & K(o) = \text{ruled out}, \\ \texttt{INSPECT}, & K(o) = \text{unknown}. \end{cases} \tag{4}$$

The last three branches require validation and risk certification. Available recovery requires verified rollback and certified reversible recovery. Unknown or inconsistent evidence leads to inspection, not permission.

For the deployment example, a valid packet with a certified recovery checkpoint leads to recovery. A valid packet that rules out recovery leads to abort. A missing tool declaration, stale digest, or unresolved side effect leads to inspection. More completed preparation steps do not change any branch.

4.3 CHECK BOTH AUTHORIZATION AND THE RESULTING STATE

Recovery has a precondition specifying when it is allowed and a postcondition checking what it restores (Hoare, 1969). In the symbolic environment these are task-state rules. Native adapters check checkpoint manifests and read restored bytes. Selecting the recovery label and successfully completing recovery are therefore separate outcomes.

The interface gives a direct history-independence property:

$$\pi(h_1, z, e) = \pi(h_2, z, e) \quad \text{for identical validated } (z, e). \tag{5}$$

The reason is structural: deterministic extraction and the same decision rule receive identical authorization inputs. This property makes an obligation explicit for the adapter: (z, e) must contain the facts that determine current permissions and recovery. Excluding history from a rule does not make an omitted permission irrelevant. The tests below evaluate both action choice and evidence acceptance.

At this boundary the two prohibited labels are absent from the gate's output set by construction. Correct resolution asks the additional question: does the controller choose the recovery or abort required by the task? Evidence validation asks whether the controller should make a decisive choice at all. These three checks give the gate's safety result its operational meaning.

5 CONTROL EVALUATION: RESOLVE THE TASK, VALIDATE THE EVIDENCE

5.1 CLEAN-STATE COMPARISON

The clean-state set contains 420 tasks: 336 require recovery and 84 require abort. Method development uses the other 140 canonical tasks. This split was defined after earlier studies of the task family; the 140 new-template tasks form a separate transfer set. Each clean-state method has six model configurations and 2,520 task–model records.

The comparison covers three intervention points. Text-based methods change the instructions or deliberation: a safety prompt, reflection, self-critique, an explicit-contract prompt scaffold, and a state-only critic that reads current-state text. Action restrictions use an action-label blocker or registered-action guard. Finally, the direct state rule and DECOMMIT read structured state and

Table 4: Clean-state outcomes across six configurations, with 2,520 records per method. U is prohibited-action rate, S recovery success, and C correct recovery or abort (all in percent). Utility $J = S - U$ uses proportions.

Method	$U \downarrow$	$S \uparrow$	$C \uparrow$	$J \uparrow$
No defense	39.2	15.6	16.0	-0.236
Safety prompt	4.9	64.2	65.6	0.593
Reflection	38.9	10.2	11.8	-0.287
Self-critique	12.4	29.2	35.6	0.169
Action blocker	0.0	9.4	18.5	0.094
Action guard	0.0	45.6	54.0	0.456
Prompt scaffold	6.5	34.0	43.0	0.274
State-only critic	6.2	27.8	32.0	0.216
Direct state rule	0.0	80.0	100.0	0.800
DECOMMIT	0.0	80.0	100.0	0.800

Table 5: Evidence-boundary results. Each row names its criterion: preserving a correct action, rejecting invalid evidence, and resolving a forged state are different tests.

Criterion	Direct rule	DECOMMIT
Benign action preservation	1,260/1,260	1,260/1,260
Detectable-fault rejection	0/6,300	6,300/6,300
Forged-state correct resolution	0/420	0/420

implement the same recovery-or-abort rule; only DECOMMIT validates the evidence first. The full comparison evaluates these complete decision configurations. The direct-rule pair isolates the added validation layer.

Both structured rules achieve $U = 0\%$, $S = 80\%$, and $C = 100\%$ (Table 4): every model configuration recovers on all 336 recovery tasks and aborts on all 84 abort tasks. By comparison, the blocker and guard also prevent prohibited actions, but correctly resolve 18.5% and 54.0% of tasks. The safety prompt reaches 65.6% correct resolution with 4.9% prohibited actions. Preventing the original operation and completing the required alternative are different achievements.

The 80% recovery rate reflects the recoverable share, not failed recovery on the remaining tasks. If that share is ρ, then $S \leq \rho$ and $U \geq 0$ imply $J \leq \rho$. Both structured rules reach this ceiling. An always-recover policy can also reach $J = \rho$ under the symbolic scoring, but resolves only the recoverable fraction correctly. Reporting C distinguishes that policy from the correct recovery-or-abort rule.

5.2 EVIDENCE VALIDATION IS THE ADDED CAPABILITY

Each confirmation task receives three benign packets, 15 detectable faults, and one coherent forgery. This validation design was developed on the 140-task development set before confirmation. The benign versions are the clean packet, a packet with 200 added history events urging preservation of progress, and a packet with extra task, label, and verifier fields outside the authorization input. Both rules preserve the correct actions on all 1,260 benign packets.

The detectable faults test five components: a missing or stale digest (2 types); a missing, false, or non-Boolean risk flag (3); missing or non-Boolean rollback availability (2); missing or nonempty side effects (2); and missing or malformed tools, a missing tool name, an unknown side-effect type, duplicate names, or inconsistent capabilities (6). Type-focused mutations recompute the digest, requiring more than a hash check. These 15 mutations on 420 tasks yield 6,300 packets.

DECOMMIT returns inspection for every detectable fault; the direct rule accepts every packet (Table 5). Acceptance and action correctness are distinct: an invalid packet can happen to produce the registered action. The measured contribution is rejecting all enumerated detectable faults while preserving every benign resolution.

Table 6: Native-loop results. Each row has 400 attempts; N is eligible episodes. U and S are percentages of those episodes. P counts jointly eligible, state-matched full–masked pairs; Δ_H is their unsafe-rate difference in pp. Coder and 14B denote Qwen-Coder-3B and Qwen2.5-14B; Critic is the state-only critic.

Framework	Model	Method	N	U	S	P	Δ_H
SmolAgents	Coder	None	328	0.0	47.3	162	0.0
	Coder	Critic	328	44.8	55.2	162	0.0
	Coder	DECOMMIT	328	0.0	62.5	162	0.0
	14B	None	332	0.0	52.1	164	0.0
	14B	Critic	332	45.5	54.5	164	0.0
	14B	DECOMMIT	332	0.0	68.4	164	0.0
mini-SWE	Coder	None	324	0.0	47.8	158	0.0
	Coder	Critic	320	43.4	56.6	158	0.0
	Coder	DECOMMIT	324	0.0	59.9	158	0.0
	14B	None	322	3.1	56.5	156	2.6
	14B	Critic	322	0.0	45.7	156	0.0
	14B	DECOMMIT	320	0.0	80.0	156	0.0

The coherent forgery flips rollback availability and recomputes the digest. Both rules accept these internally consistent packets and choose the wrong decisive resolution on all 420 cases. This identifies where truth about the environment must enter the system: the adapter. A digest checks consistency, not authenticity. If different states requiring different actions map to the same validated packet, a deterministic packet-only controller returns the same action for both. Binding evidence to the actual current state is therefore part of the adapter contract.

6 NATIVE TOOL LOOPS: REACHABILITY AND COMPLETED RECOVERY

We evaluate isolated tasks in SmolAgents 1.26.0 and mini-SWE-agent 2.4.6 (Hugging Face, 2025; Lieret & Jimenez, 2026), using Qwen-Coder-3B and Qwen2.5-14B. Each framework–model configuration runs no defense, a state-only critic, and DECOMMIT with full or masked history. mini-SWE uses a restricted local environment adapter. Recovery earns success only after its postcondition passes.

Here, the intervention retains or masks the tool pipeline's history. It changes visible context, unlike the benchmark's matched reassignment of prior work. For each method, we pair full and masked episodes only when both reach an eligible decision and their current states match. This measures history retention on shared reachable decisions.

Reachability and pairing. The 4,800 attempts produce 3,912 eligible decisions, 764 no-decision outcomes, and 124 caught exceptions. Table 6 keeps two denominators visible: all eligible episodes for action and recovery rates, and the shared pair set for the history difference. For example, mini-SWE/14B without defense reaches 162 full-history and 160 masked-history decisions, with five prohibited actions in each group. Its 156 shared state-matched pairs have a 2.6 pp difference. Equal whole-group counts do not imply equal counts on the shared subset.

History retention. The 12 comparisons contain 1,920 eligible same-state pairs. Eleven differences are 0.0 pp; mini-SWE/14B without defense yields 2.6 pp. A low-load SmolAgents/14B repeat produces 980 eligible decisions, 128 no-decision outcomes, and 92 exceptions from 1,200 attempts. Each method has 161 state-matched pairs, with all three history differences equal to zero. The matched benchmark's positive attribution differences therefore do not automatically transfer to masking history in these native pipelines.

Completed recovery. DECOMMIT selects no prohibited actions in any framework–model aggregate and achieves 59.9–80.0% recovery success. No-defense recovery rates are 47.3–56.5%, with zero prohibited actions in three of four aggregates. The gate's recovery rate is higher in each re-

ported aggregate; these are configuration-level rates on their respective eligible sets. The state-only critic has zero paired history differences in all four configurations, yet prohibited-action rates of 43.4–45.5% in three. History invariance, absolute safety, and recovery success need separate measurements.

7 Related Work

Agent safety and task completion. ToolEmu measures risks in emulated tool environments, AgentHarm evaluates harmful multistep completion, and τ-bench studies policy-constrained tool–agent–user interactions (Ruan et al., 2024; Andriushchenko et al., 2025; Yao et al., 2025). AgentSafetyBench covers safety failures across agent tasks, while SABER evaluates operational safety through workspace state (Zhang et al., 2024; Hu et al., 2026a). Interruptibility and abstention studies ask when an agent should change course or refrain from acting (Zou et al., 2026; Liu et al., 2026). AbortBench adds a controlled history comparison at an already specified risk boundary, separating a change in the presented past from a change in what the task currently permits.

Context, memory, and commitment. Long-context research studies how information placement changes model behavior, and MemoryAgentBench evaluates memory through incremental interactions (Liu et al., 2024; Hu et al., 2026b). AgentDojo, InjecAgent, and MemoryGraft examine attacks through tool-visible data or retrieved experience (Debenedetti et al., 2024; Zhan et al., 2024; Srivastava & He, 2025). Our matched reference controls the amount and structure of prior activity while changing its task attribution; a malicious instruction is not required. Research on sunk costs and escalating commitment studies the distinction between prior investment and prospective value (Staw, 1976; Arkes & Blumer, 1985; Kleinberg et al., 2021). Barkett et al. (2025) examine LLM commitment in two-stage investment tasks. We measure observable action changes under a fixed current contract, without assigning the model a psychological mechanism.

Runtime enforcement. Shielding and Simplex separate proposed actions from permission to execute them, while selective classification formalizes abstention from unsuitable decisive predictions (Alshiekh et al., 2018; Seto et al., 1998; Geifman & El-Yaniv, 2017). Agent-oriented work includes pre-execution safety contracts and ToolSafe's step-level guardrail and feedback (Obi et al., 2026; Mou et al., 2026). DeCommit applies this separation to delayed-risk resolution, with explicit evidence validation and recovery checks. Its direct-rule comparison places the empirical contribution at evidence acceptance: the clean-state choices are the same, while handling of detectable faults differs.

8 Discussion and Conclusion

A history-safety score needs a visible reference. The three-condition protocol distinguishes adding other-workflow history from assigning matched work to the current task. It separates a high unsafe-action rate from a large linkage difference and explains why broad and matched model rankings can disagree.

These measurements describe the evaluated tasks, renderer, model configuration, and action contract. Wording changes their magnitudes; native history masking changes the intervention itself. Adapting the protocol requires keeping current permissions and recovery information fixed, and measuring decision reachability for each runtime.

For execution, the principle is to separate useful context from authorizing evidence. History supports proposals and audit; the adapter establishes current facts, the gate validates them, and recovery checks the result. The forged-packet test makes the adapter's role concrete: internally consistent evidence can still describe the wrong state.

AbortBench finds positive matched linkage differences of 4.6–27.0 pp across six primary configurations, while showing how exposure can dominate a broad history effect. DeCommit turns the same separation into a recovery-or-abort interface, preserving clean-state resolution and rejecting the tested detectable faults. Together, they provide an interpretable way to measure history sensitivity and a concrete boundary for controlling what an agent does next.

AI USE STATEMENT

We used generative AI tools to assist with grammar checking and editing, as well as checking LaTeX formatting. The authors are responsible for the final text, research claims, and related outputs.

ETHICS STATEMENT

The study uses synthetic tasks and isolated local workspaces. Tool operations target generated files, repositories, databases, and configurations, with no production assets, real credentials, external recipients, or real financial transactions. No human-participant study is involved. Evidence-integrity tests modify generated packets. Applying the gate to a live system requires authorized access, trustworthy state adapters, and isolation of the controlled operations.

REPRODUCIBILITY STATEMENT

Section 2 specifies the three history conditions, matching criteria, action labels, task sets, and decoding configuration. Table 1 gives the integer counts underlying the primary rate differences. Section 4 defines the gate's evidence checks and decision rule; Sections 5 and 6 specify the evaluation populations, fault construction, and native-loop denominators. These details make the reported comparisons and their arithmetic explicit in the main text.

REFERENCES

Mohammed Alshiekh, Roderick Bloem, Ruediger Ehlers, Bettina Konighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI Conference on Artificial Intelligence*, volume 32, pp. 2669–2678, 2018. doi: 10.1609/aaai.v32i1.11797.

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/hash/c493d23af93118975cdbc32cbe7323f5-Abstract-Conference.html.

Hal R. Arkes and Catherine Blumer. The psychology of sunk cost. *Organizational Behavior and Human Decision Processes*, 35(1):124–140, 1985. doi: 10.1016/0749-5978(85)90049-4.

Emilio Barkett, Olivia Long, and Paul Kröger. Getting out of the big-muddy: Escalation of commitment in LLMs. *arXiv preprint arXiv:2508.01545*, 2025. doi: 10.48550/arXiv.2508.01545. URL https://arxiv.org/abs/2508.01545.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, 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*, volume 37, 2024. doi: 10.52202/079017-2636. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z.

C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12 (10):576–580, 1969. doi: 10.1145/363235.363259.

Qi Hu, Yifeng Tang, Qinghua Wang, Lanyang Zhao, Pengji Zhang, Yuhao Qing, Xin Yao, Dong Huang, Lin Zhang, and Zhuoran Ji. SABER: Benchmarking operational safety of LLM coding agents in stateful project workspaces. *arXiv preprint arXiv:2606.01317*, 2026a. URL https://arxiv.org/abs/2606.01317.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *International Conference on Learning Representations*, 2026b. URL https://iclr.cc/virtual/2026/poster/10010781.

Hugging Face. Smolagents: Agents that think and act with small language models. https://huggingface.co/docs/smolagents/v1.26.0/en/index, 2025. Version 1.26.0 used for the external-agent validation.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

Jon Kleinberg, Sigal Oren, Manish Raghavan, and Nadav Sklar. Stochastic model for sunk cost bias. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1279–1288. PMLR, 2021. URL https://proceedings.mlr.press/v161/kleinberg21a.html.

Kilian Lieret and Carlos E. Jimenez. mini-SWE-agent. https://mini-swe-agent.com/latest/reference/, 2026. Software release 2.4.6 used with a restricted local environment adapter.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Xun Liu, Yi Evie Zhang, Vira Kasprova, Parisa Rabbani, Pardis Sadat Zahraei, Tianyu Zhang, Ali Ebrahimpour-Boroojeny, and Varun Chandrasekaran. AgentAbstain: Do LLM agents know when not to act? *arXiv preprint arXiv:2607.10059*, 2026. URL https://arxiv.org/abs/2607.10059.

Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. ToolSafe: Enhancing tool invocation safety of LLM-based agents via proactive step-level guardrail and feedback. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37125–37153. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1850. URL https://aclanthology.org/2026.findings-acl.1850/.

Ike Obi, Vishnunandan L. N. Venkatesh, Weizheng Wang, Ruiqi Wang, Dayoon Suh, Temitope I. Amosa, Wonse Jo, and Byung-Cheol Min. Pre-execution safety gate & task safety contracts for LLM-controlled robot systems. *arXiv preprint arXiv:2604.05427*, 2026. URL https://arxiv.org/abs/2604.05427.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

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.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2302.04761.

D. Seto, Bruce Krogh, Lui Sha, and Alongkrit Chutinan. The simplex architecture for safe on-line control system upgrades. In *American Control Conference*, volume 6, pp. 3504–3508, 1998. doi: 10.1109/ACC.1998.703255.

Saksham Sahai Srivastava and Haoyu He. MemoryGraft: Persistent compromise of LLM agents via poisoned experience retrieval. *arXiv preprint arXiv:2512.16962*, 2025. doi: 10.48550/arXiv.2512.16962. URL https://arxiv.org/abs/2512.16962.

Barry M. Staw. Knee-deep in the big muddy: A study of escalating commitment to a chosen course of action. *Organizational Behavior and Human Performance*, 16(1):27–44, 1976. doi: 10.1016/0030-5073(76)90005-2.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://iclr.cc/virtual/2023/poster/11003.

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

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.624. URL https://aclanthology.org/2024.findings-acl.624/.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the safety of LLM agents. *arXiv preprint arXiv:2412.14470*, 2024. URL https://arxiv.org/abs/2412.14470.

Henry Peng Zou, Chunyu Miao, Wei-Chieh Huang, Yankai Chen, Yue Zhou, Hanrong Zhang, Yaozu Wu, et al. When users change their mind: Evaluating interruptible agents in long-horizon web navigation. *arXiv preprint arXiv:2604.00892*, 2026. URL https://arxiv.org/abs/2604.00892.