# Rollback-Safe Long-Horizon Generation: Auditing and Transactional State Commit for Narrative Agents

**Anonymous authors**
**Paper under double-blind review**

## Abstract

Long-horizon language-model systems do not only fail by forgetting context; they also fail when fluent but unsafe continuations are written into persistent memory as state. We study a rollback-safe runtime for narrative agents that treats each continuation as a candidate state transition: text, extracted events, and state deltas are staged, audited, gated, and then either committed atomically or preserved as a rejected trace. The paper's organizing claim is that runtime integrity depends on coupling detection to the write boundary. Three findings support this claim. First, post-hoc audit over 48 internal outputs finds human-confirmed corruption in every condition family, showing that context and memory alone do not eliminate unsafe drafts. Second, auditor calibration exposes detection as the bottleneck: the current auditor is high-recall but poorly calibrated, with overall precision/recall/F1 of 0.113/0.727/0.196 and a failed motivation-drift channel. Third, write-control replay shows that merely critiquing or recording a failure is architecturally insufficient; the empirical question is how much corruption the deployed gate actually intercepts. In the present replay, the current gate misses most human-confirmed corruptions, while a human-calibrated upper bound eliminates silent commits by construction. Public sanity checks on ToolSandbox, ConStory-Bench, and LongMemEval test the same runtime, narrative-consistency, and memory/KFI mechanisms outside the internal story scaffold. The contribution is diagnostic: rollback-safe commit is a useful control boundary, but its value is gated by calibrated audit signals.

## 1 Introduction

Long-horizon language-model systems often fail in ways that are not captured by local fluency. A generated chapter can read well while contradicting established world state, treating known information as newly discovered, changing a character's goals without cause, or extracting an erroneous event that future retrieval will treat as canon. Once such an output is committed into a persistent memory or state store, later generations may retrieve and amplify the error. This is a state-pollution problem, not merely a context-window problem.

Most long-horizon generation systems respond to this pressure by adding more context, better retrieval, or more memory. These are useful primitives, but they do not define the control boundary at which generated text becomes persistent state. Our starting point is therefore operational: a continuation should be treated as a candidate state transition. The system should decide whether the text, extracted events, and state updates are safe to commit atomically; if not, the transition should be repaired, blocked, or rolled back while preserving an auditable trace.

We study this problem in long-form narrative agents because narrative generation exposes several runtime-integrity failures at once. The system must preserve world state, character state, timeline continuity, and character epistemic state over many chapters. A character who already knows a fact should not rediscover it as new; a resolved plan should not be reintroduced without explanation; a harmful extracted event should not enter memory simply because the prose is fluent.

The proposed runtime implements the loop

$$\texttt{read state} \rightarrow \texttt{retrieve} \rightarrow \texttt{generate draft} \rightarrow \texttt{audit} \rightarrow \texttt{gate} \rightarrow \texttt{commit or rollback}.$$

The analysis is organized around a detection–interception coupling principle: audit signals matter for runtime integrity only when calibrated detectors are connected to the write boundary. The paper therefore studies where this coupling succeeds and where it fails, rather than treating audit scores as a proxy for story quality.

The paper makes three diagnostic contributions. First, it shows that critique or audit records are not sufficient for runtime integrity unless they control the persistent-write boundary. Second, it quantifies a practical bottleneck: the current LLM-assisted auditor is high-recall but poorly calibrated, and its motivation-drift channel fails under human adjudication. Third, it evaluates rollback-safe commit as a control boundary through post-hoc corruption audit, write-control replay, threshold sensitivity, gate-distribution analysis, qualitative corruption cases, and public sanity checks on ToolSandbox, ConStory-Bench, and LongMemEval.

## 2 Related Work

### 2.1 Retrieval, long context, and memory

Retrieval-augmented generation conditions language models on non-parametric external memory rather than relying only on model weights or prompt context (Lewis et al., 2020). Long-context evaluations show that larger windows do not by themselves guarantee reliable use of information: models can fail depending on where relevant information appears in context (Liu et al., 2024), and broad benchmarks such as LongBench evaluate this behavior across long-context tasks (Bai et al., 2023). Agent-memory systems extend this direction by maintaining persistent memory outside the immediate prompt. MemoryBank stores and retrieves long-term conversational memories (Zhong et al., 2023), while MemGPT frames memory management as an operating-system-like problem with tiered context and control flow (Packer et al., 2023).

These systems motivate external memory and context construction, but memory availability is not the same as write safety. If a bad continuation is stored as canon, better retrieval can make future generations worse by faithfully retrieving the corrupted state. Our focus is the commit boundary: whether generated text and extracted events are allowed to become persistent state.

### 2.2 Belief and epistemic-state tracking

The epistemic-memory component of our system is related to theory-of-mind and belief-tracking evaluations. Early QA-style benchmarks ask models to track beliefs that can diverge from reality, including false-belief and second-order belief cases (Nematzadeh et al., 2018). FANToM moves this setting into information-asymmetric interactions (Kim et al., 2023); Hi-ToM studies higher-order recursive belief reasoning (Wu et al., 2023); and BigToM procedurally generates social-reasoning evaluations from causal templates (Gandhi et al., 2023). These benchmarks test whether a model can infer who knows what from a fixed scenario or dialogue. Our setting is different: the model generates a continuation, an extractor proposes events and epistemic deltas, and the runtime decides whether those deltas should become persistent state. Thus the relevant failure is not only an incorrect answer about belief, but an incorrect belief-state write that future generation may retrieve as canon.

### 2.3 Agent feedback, reflection, and critique

Generative Agents introduced a memory-stream architecture where observations are stored, retrieved, reflected on, and used to plan behavior (Park et al., 2023). Voyager uses an ever-growing skill library and iterative environment feedback for open-ended embodied exploration (Wang et al., 2023). ReAct interleaves reasoning and action in interactive environments (Yao et al., 2023), and Reflexion stores verbal feedback from prior attempts to improve later behavior (Shinn et al., 2023).

Several methods improve model outputs through critique or iterative refinement. Self-Refine uses self-feedback to iteratively revise outputs without additional training (Madaan et al., 2023). CRITIC studies

tool-interactive critiquing (Gou et al., 2024), and Self-RAG trains retrieve-generate-critique behavior through reflection tokens (Asai et al., 2024). These approaches show that critique can improve outputs, but a critique that is merely recorded still permits unsafe downstream writes. Our write-control ablations explicitly test this distinction.

### 2.4 Narrative consistency and stateful benchmarks

Long-form narrative generation has specialized failure modes: character-state contradictions, timeline errors, world-state drift, and knowledge-state errors. ConStory-Bench / Lost in Stories targets consistency bugs in long story generation and provides an evidence-grounded checker over long-form outputs (Li et al., 2026b). This is closer to our domain than generic QA, but it evaluates generated text rather than persistent commit safety.

Recent agent benchmarks increasingly evaluate persistent side effects. ToolSandbox evaluates conversational tool-use agents in a stateful interactive sandbox with user simulation, milestones, and minefields (Lu et al., 2024). STATE-Bench studies memory in stateful task environments with deterministic assertions over final environment state (Microsoft Open Source Blog, 2026). EnterpriseOps-Gym evaluates enterprise agents in containerized environments where tool actions permanently affect database state (Malay et al., 2026). These benchmarks are not narrative generation, but they are aligned with the runtime-integrity question: an agent can produce plausible text while leaving the world in a bad state. LongMemEval evaluates long-term memory over timestamped multi-session histories (Wu et al., 2024); we use it to sanity-check the key-fact-injection side of our system.

### 2.5 Runtime governance and write-time control

Recent work on tool-using agents also treats safety as a runtime control problem. ClawGuard enforces task-specific constraints at the tool-call boundary to block indirect prompt-injection effects before external side effects occur (Zhao et al., 2026). SafeAgent frames agent protection as stateful runtime governance over evolving interaction trajectories (Liu et al., 2026). TRACES studies proactive, prefix-level risk auditing for multi-turn agents, estimating whether a partial trajectory is drifting toward unsafe behavior before the final outcome appears (Li et al., 2026a). Atomix, SagaLLM, ALAS, and PCAS study adjacent transactional or policy-governed agent runtimes, including transactional tool effects, checkpoint/compensation patterns, versioned execution logs, and declarative policy enforcement (Mohammadi et al., 2026; Chang & Geng, 2025; Geng & Chang, 2025; Palumbo et al., 2026). These systems operate primarily over tool actions, planning workflows, policy checks, or trajectory-level safety. Our setting is complementary: the controlled side effect is not a web request or tool call, but the write of generated text, extracted events, and entity/world-state deltas into narrative memory. This related work also clarifies a limitation of the present implementation: our current write-time constraints are operational and empirical rather than formally verified.

Persistent memory also creates a security and governance surface of its own. AgentPoison shows that poisoning an agent's memory or knowledge base can steer later behavior (Chen et al., 2024); MINJA studies memory injection through query-only interaction (Dong et al., 2025); and environment-injected memory poisoning attacks show that a single poisoned interaction can persist and be exploited later by web agents (Zou et al., 2026). These works are adversarial-security studies rather than narrative-generation evaluations, but they support the same systems premise: once untrusted content is written into long-term state, later retrieval and planning can amplify it. Our contribution is not a complete memory-security defense. It is a write-time integrity layer for narrative state, motivated by the same persistence boundary.

## 3 Runtime System

### 3.1 State representation

The system maintains three state layers. Chapter and event state stores generated chapters, extracted events, local metadata, and audit traces. Entity and world state stores characters, locations, world facts, timeline fields, and character epistemic facts. Project state stores checkpoints, experiment condition flags,

failure records, and rollback metadata. JSON is treated as the source of truth; graph and vector stores are acceleration layers for retrieval and analysis rather than authoritative state.

This choice is operationally important. A generation run can be restored from a checkpoint, replayed under a different condition, or repaired without relying on mutable graph-store side effects. It also supports post-hoc corruption audits: accepted baseline outputs can be inspected as candidate commits even if they were generated without the full runtime gate.

Each candidate write is represented as a transition record with a text pointer, extracted event list, entity/world-state delta, epistemic-fact delta, checkpoint id, audit records, gate action, and provenance identifiers. The current schema enforces practical invariants: generated text, events, and state deltas share one transition id; no staged delta is promoted before a gate decision; rollback restores the prior checkpoint while preserving the rejected trace; and accepted deltas must refer to known or explicitly introduced entity identifiers. These invariants support replay and auditability, but they are not a formal proof of state consistency.

### 3.2 Context construction and key facts

Context construction combines recent chapter context, structured state retrieval, optional semantic retrieval, and optional key-fact injection (KFI). KFI is not just retrieval of relevant facts. It packages facts with epistemic framing: which character knows the fact, whether the fact is already established, and how the generation should treat it. This is intended to reduce known-information rediscovery, where a character acts as if an already-known fact is newly discovered.

The main ablation family compares four controlled conditions inside the same experimental scaffold: **B0**, a full-context baseline; **B1**, a semantic-retrieval baseline; **B2**, a memory/KFI-only baseline; and **S**, the full runtime with audit, gate, rollback, and transactional commit. These conditions are controlled baselines and ablations rather than complete implementations of every external RAG or memory-agent system.

### 3.3 Auditing and gating

For each draft, the runtime audits candidate failures through three channels. The **state-violation** channel checks for contradictions with established world, character, timeline, or epistemic state. The **motivation-drift** channel checks for unexplained violations of established goals, plans, fears, incentives, or personality constraints. The **narrative-degeneration** channel checks for structural collapse, repetition, causal breakage, or unusable continuation.

The gate maps audit signals to actions. Critical state violations can block commit and trigger rollback. Motivation drift can trigger repair or review. Narrative degeneration is recorded unless it reaches a severity/action threshold, because this channel is more subjective and more prone to judge variance. This design is intentionally conservative: the system should not treat a noisy judge as ground truth.

The gate uses a channel-specific action table. Critical state violations block by default; major state violations require repair or review; motivation-drift findings require repair/review only above the active threshold; and narrative-degeneration findings are normally record-only unless they are severe enough to make the continuation unusable. This mapping intentionally privileges hard state constraints over subjective quality judgments.

### 3.4 Transactional commit and rollback

The key control boundary is commit. Draft text, extracted events, and state updates are staged together. If the gate accepts the transition, these artifacts are committed atomically. If the gate blocks the transition, the system preserves the trace, restores the previous checkpoint, and records the failure instead of writing the candidate update into persistent state. Post-rollback policy depends on condition: the run can stop, retry with a repaired prompt, or defer to human review. In the experiments reported here, rollback is treated as a successful interception of an unsafe transition, not as evidence of better writing.

| Metric | Unit | Meaning |
|---|---|---|
| Output-level corruption | Output | A human-confirmed harmful state, consistency, motivation, or narrative-integrity error in the generated continuation. |
| Unobserved-corruption proxy | Output/condition | A weaker traceability measure: whether a harmful output is represented as an auditable transition rather than only ordinary text. This does not by itself imply safety. |
| Deployed-gate silent corruption | Output/gate replay | A stricter replay metric: a harmful output is silent if the current deployed gate would still permit the persistent write. |
| ToolSandbox unsafe proxy | Trajectory | A stateful tool trajectory with an exception or positive minefield similarity, used as a public side-effect analogue rather than a narrative-state label. |
| ToolSandbox live unsafe commit | Trajectory | A ToolSandbox trajectory that remains unsafe after the live pre-execution gate has had the chance to block tool-call code. |

Table 1: Metric definitions. The unobserved-corruption proxy and the deployed-gate replay answer different questions: whether a harmful transition is traceable versus whether the current detector actually blocks it.

## 4 Evaluation Protocol

The evaluation separates three questions that are otherwise easy to conflate. First, are the audit channels aligned with human judgments? Second, do simpler baselines silently commit harmful state transitions? Third, does an audit signal reduce corruption only when it controls persistent writes? This separation is necessary because system-side execution success is not the same as story quality or human-confirmed correctness.

### 4.1 Internal narrative-agent data

The internal deployment suite uses two anonymized story settings and two models. The principal condition family for the TMLR analysis is B0/B1/B2/S over six windows and two models, yielding 48 outputs. We use these outputs for post-hoc corruption audit, write-control ablation, and case analysis.

We define *output-level corruption* as a generated output containing a human-confirmed harmful state or consistency error. We define *deployed-gate silent corruption* as a harmful output that the current gate would still write without block, rollback, or pre-commit human review. More concretely, let $H_i = 1$ indicate a human-confirmed harmful transition for item $i$, and let $W_i(c) = 1$ indicate that condition $c$ would write the transition without blocking or rollback-safe holding. Then $SC_i(c) = H_i W_i(c)$. For baselines without write control, accepted harmful outputs have $W_i = 1$. We separately report a weaker *unobserved-corruption proxy*: whether a harmful transition is at least trace-visible under the full runtime. A preserved trace alone is not safety if the harmful write still commits; the stricter deployed-gate replay in Table 5 is therefore the main safety-oriented metric.

### 4.2 Models, decoding, and reproducibility

The internal narrative-agent runs use two model identifiers: `deepseek-chat` and the provider endpoint identifier `openai-codex/gpt-5.1-codex` recorded in the run manifests. The latter is reported as an endpoint

identifier rather than as a claim about a standardized public model release. All public benchmark runs reported in this draft use `deepseek-chat` through an OpenAI-compatible API wrapper so that the external checks can be reproduced under one provider. For LongMemEval, decoding is deterministic with temperature 0 and a maximum of 128 generated tokens. For ConStory-Bench, we use the official ConStory generation and judge pipeline, set the maximum generation length to 4000 tokens, and run with concurrency 1. For ToolSandbox, we keep the upstream ToolSandbox agent/user API classes and replace only the OpenAI-compatible client endpoint with DeepSeek; the 100-scenario subset is sampled deterministically with seed 20260609. Where the upstream benchmark runner does not expose a temperature or top-$p$ control, we preserve the upstream default and report the result as a benchmark-transfer run rather than as a decoding-sensitivity study.

For the historical internal narrative-agent runs, condition definitions, provider/model identifiers, output paths, traces, and audit records are preserved in run manifests. Because not every historical request preserved a full per-call decoding record, we do not make claims that depend on decoding hyperparameter optimization. The TMLR evidence instead uses post-hoc human labels, replayed gates, and public benchmark checks over saved outputs. Runtime-cost reporting is limited to observable wall-clock generation time in the saved manifests; historical provider logs do not preserve per-call token usage.

### 4.3 Human annotation and calibration

The calibration set contains 192 channel-level samples derived from 64 unique outputs. Each row is assigned to one audit channel and receives labels for violation status, channel, severity, commit harm, and confirmed corruption. The annotation campaign contains 384 completed labels from a nine-annotator pool; each row receives two independent labels drawn from that pool. Raw pairwise agreement on the pre-adjudication violation labels is 0.818, and Gwet's AC1 is 0.801. However, nominal Krippendorff's $\alpha$ is $-0.097$, and positive-specific agreement is 0/34: every row with an initial positive violation label has only one positive annotator before adjudication. Thirty-five rows entered adjudication: 34 were one-violation/one-non-violation splits, and one was a non-violation/unclear split. Thus the adjudicator resolved ties rather than overturning two-annotator consensus. The adjudicator was an author-side human adjudicator, not an independent external expert, and could see condition and run metadata. This creates a single-adjudicator limitation. Its effect is partly mitigated, but not removed, by the diagnostic use of the labels: the labels primarily expose auditor and gate failures rather than support a performance-superiority claim. The adjudicator's resolution determined the final commit-harm or confirmed-corruption decision in 31 rows. We prepared a condition-blind relabel packet for a later robustness pass, but do not report it as completed here. We then replay the current auditor over the adjudicated samples and compute precision, recall, F1, false-positive rate, and false-negative rate.

### 4.4 Public benchmarks

We use public benchmarks for distinct purposes. ToolSandbox is the closest public analogue to runtime integrity because it involves stateful tool use, side effects, and unsafe minefield actions. ConStory-Bench checks long-form narrative consistency under an external metric but does not evaluate state-write corruption. LongMemEval checks whether compact key-fact evidence helps memory use outside the internal story setup.

## 5 Results

### 5.1 Auditor calibration

Table 2 reports the calibration result. The current auditor is high-recall but poorly calibrated. Overall precision is 0.113, recall is 0.727, and F1 is 0.196. Wilson 95% intervals are wide: overall precision is [0.071, 0.176] and recall is [0.518, 0.868]. State-violation and narrative-degeneration channels both reach recall 1.000 in this sample, but with very low precision. Motivation drift performs poorly, with precision and recall both 0.000; its recall interval is [0.000, 0.390] because the positive count is small.

| Channel | N | TP | FP | FN | Precision | Recall |
|---|---|---|---|---|---|---|
| All | 192 | 16 | 125 | 6 | 0.113 | 0.727 |
| State violation | 64 | 10 | 54 | 0 | 0.156 | 1.000 |
| Motivation drift | 64 | 0 | 13 | 6 | 0.000 | 0.000 |
| Narrative degeneration | 64 | 6 | 58 | 0 | 0.094 | 1.000 |

Table 2: Auditor calibration against human-adjudicated labels. The current auditor should be treated as a screening signal, not as a calibrated oracle.

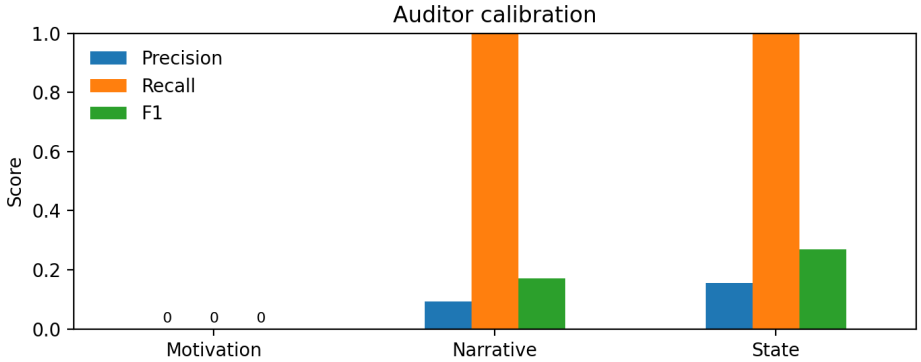


Figure 1: Auditor calibration by channel. High recall with low precision motivates calibrated gating rather than treating runtime audit as ground truth.

This result directly weakens any circular interpretation of system success. We do not treat the runtime auditor as a ground-truth evaluator; instead, the calibration exposes its limits and motivates the write-control framing.

### 5.2 Post-hoc corruption audit

Table 3 audits accepted outputs after generation. B0 has output-level corruption in 5/12 runs, B1 in 3/12, B2 in 1/12, and S in 4/12. The Wilson 95% intervals are necessarily broad, for example B0 output-level corruption is 0.417 [0.193, 0.680], while S unobserved-corruption proxy has interval [0.000, 0.243]. The full runtime therefore does not produce error-free text, and these rates should be read as effect estimates rather than precise population rates. The relevant difference is the commit boundary. Under the weak unobserved-corruption proxy, harmful S transitions are represented as auditable runtime records rather than only ordinary text. The stricter deployed-gate replay in Table 5 then tests whether the current gate actually blocks those harmful writes.

The low B2 corruption count should not be interpreted as evidence that the full runtime worsens generation. B2 is a memory/KFI-only condition, while S adds audit, gate, rollback, and commit control; these mechanisms do not directly change the draft-generation distribution in a way that should guarantee lower output-level corruption. The relevant division of labor is different: context and KFI can reduce bad outputs, while write control governs what happens when a bad output still appears. With only 12 outputs per condition, the B2 and S Wilson intervals overlap substantially: B2 is 1/12 [0.015, 0.354], while S is 4/12 [0.138, 0.609]. We therefore treat B2 as evidence that memory/context engineering and write-control are complementary rather than interchangeable.

| Condition | Runs | Output corrupt | Output rate | Unobserved corrupt | Unobserved rate |
|---|---|---|---|---|---|
| B0 | 12 | 5 | 0.417 | 5 | 0.417 |
| B1 | 12 | 3 | 0.250 | 3 | 0.250 |
| B2 | 12 | 1 | 0.083 | 1 | 0.083 |
| S | 12 | 4 | 0.333 | 0 | 0.000 |

Table 3: Post-hoc corruption audit over accepted outputs. The full runtime still has output-level problems; the final column is a weak unobserved-corruption proxy, not proof that the deployed gate blocks every harmful write.

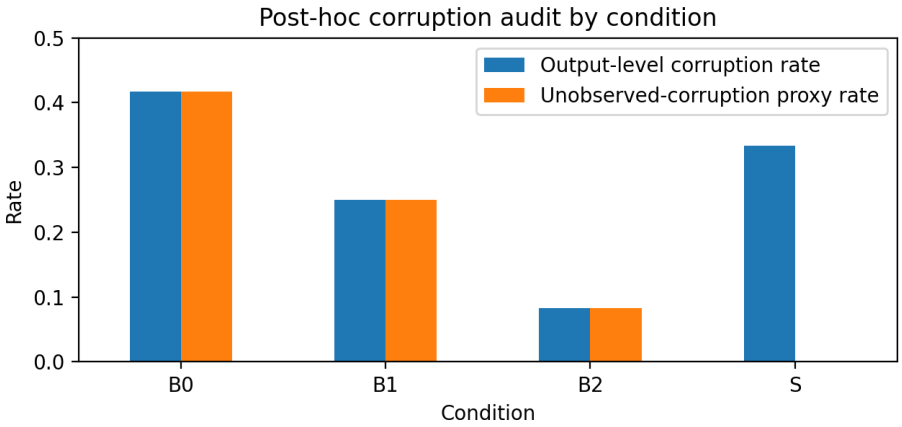


Figure 2: Output-level corruption versus unobserved-corruption proxy. The full system is not an error-free generator; the stricter deployed-gate replay is needed to test whether harmful writes are actually blocked.

### 5.3 Threshold sensitivity

Table 4 replays the gate threshold over the 192 labeled rows. At threshold 0.50, recall is 1.000 but false rollback is 0.918. At threshold 0.80, precision is 0.139, recall is 0.455, and false rollback is 0.365. At threshold 0.95, precision is 0.154, recall is 0.455, and false rollback is 0.324.

### 5.4 Critique versus write control

Table 5 separates a design fact from an empirical question. The design fact is simple: a critic or audit log that never controls writes cannot by itself reduce silent commits. The empirical question is how much corruption the current gate actually intercepts once audit signals are connected to rollback. This table answers the second question.

The table uses a different unit from Table 3: Table 3 audits 48 accepted run outputs, 12 per condition, whereas Table 5 replays policies over 64 output-level calibration samples, 16 per condition, derived from the 192 channel-level calibration rows. Thus S has 4/12 corrupt outputs in Table 3 and 5/16 human-corrupt calibration samples in Table 5; these are different sample sets rather than contradictory counts.

The empirical result is mostly negative for the current gate. On B0, the deployed gate blocks 4/16 samples but still silently commits all 5 human-corrupt samples, so the observed blocks are false rollbacks in this replay. On S, it blocks 4/16 samples, catches only 1 of 5 human-corrupt samples, and misses 4. A human-calibrated upper bound eliminates silent commits by construction. The conclusion is therefore not that Table 5 "discovers" that no-rollback policies fail; it quantifies the gap between the current detector and the write-control upper bound.

The motivation-drift channel is a negative result. In the 64-output calibration replay, it has 13 false positives, 6 false negatives, and no true positives. When motivation drift is demoted to record-only while state and

| Threshold | Precision | Recall | F1 | Block rate | False rollback |
|---|---|---|---|---|---|
| 0.50 | 0.124 | 1.000 | 0.220 | 0.927 | 0.918 |
| 0.70 | 0.106 | 0.455 | 0.172 | 0.490 | 0.494 |
| 0.80 | 0.139 | 0.455 | 0.213 | 0.375 | 0.365 |
| 0.95 | 0.154 | 0.455 | 0.230 | 0.339 | 0.324 |

Table 4: Threshold-sensitivity replay. Current thresholds expose the expected trade-off but are not validated as optimal.

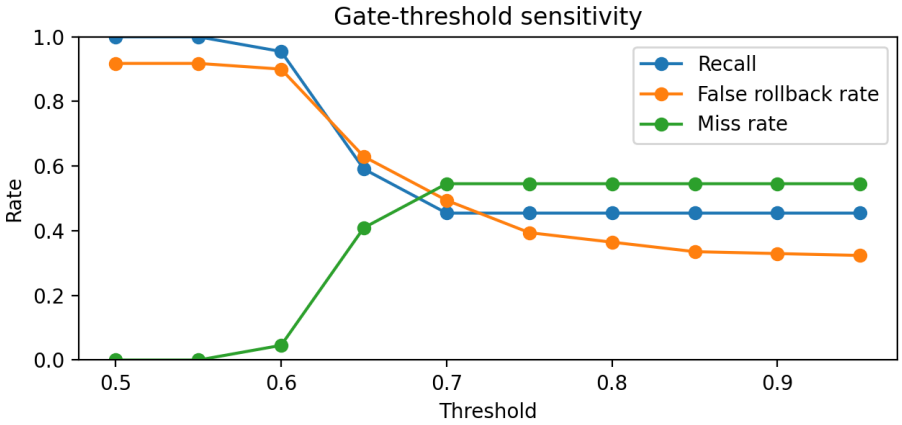


Figure 3: Threshold sensitivity over human-labeled rows. The gate requires calibration because low thresholds over-block and high thresholds still miss violations.

narrative rollback decisions remain active, the replay produces no blocks and therefore no protection against human-confirmed corruptions in the current sample. This is not a recommendation to ignore motivation drift; it shows that the present subjective judge channel is not ready to drive autonomous rollback. A calibrated replacement or human-review routing is needed before motivation-based blocking should be deployed.

### 5.5 Full-system gate distribution

Saved full-system traces show that rollback is not a one-off code path. Across 30 full-system runs, we extract 237 target-chapter failure records and 102 rollback outcomes. These are historical system traces rather than calibrated human labels, so we use them only as operational evidence that the runtime repeatedly enters explicit failure-handling paths.

### 5.6 Public benchmark sanity checks

Table 6 summarizes the public benchmark track. The roles are intentionally distinct. ToolSandbox tests stateful runtime behavior; ConStory-Bench checks generated-story consistency; LongMemEval checks compact evidence use for memory/KFI; and the synthetic rediscovery probe tests whether explicit epistemic phrasing adds value beyond bare factual packets in constructed known-fact cases.

On ToolSandbox, the 100-scenario DeepSeek-compatible run completes 100/100 scenarios with no exceptions. Average similarity is 0.712, average milestone similarity is 0.802, average minefield similarity is 0.090, and the state-dependency category average is 0.599. We first replay write-control policies over these trajectories using the minefield signal as an unsafe-side-effect proxy. Plain no-gate, critic-only, and audit-only policies silently commit 9/100 unsafe proxy trajectories, with a Wilson 95% interval [0.048, 0.162]; an oracle-style rollback-safe replay blocks those 9 and reduces unsafe committed trajectories to 0/100. This replay is an upper-bound control experiment because it uses benchmark-provided unsafe-side-effect labels. We then run

| Condition | Policy | Samples | Human corrupt | Blocked rate | Silent corrupt |
|---|---|---|---|---|---|
| B0 | No audit/no gate | 16 | 5 | 0.000 | 5 |
| B0 | Critic-only/no rollback | 16 | 5 | 0.000 | 5 |
| B0 | Audit-only/record-only | 16 | 5 | 0.000 | 5 |
| B0 | Current gate/rollback | 16 | 5 | 0.250 | 5 |
| B0 | Human-calibrated upper bound | 16 | 5 | 0.312 | 0 |
| B1 | No audit/no gate | 16 | 3 | 0.000 | 3 |
| B1 | Current gate/rollback | 16 | 3 | 0.188 | 2 |
| B2 | No audit/no gate | 16 | 1 | 0.000 | 1 |
| B2 | Current gate/rollback | 16 | 1 | 0.125 | 1 |
| S | Current gate/rollback | 16 | 5 | 0.250 | 4 |
| S | Human-calibrated upper bound | 16 | 5 | 0.312 | 0 |
| S | State/narrative gate; drift record-only | 16 | 5 | 0.000 | 5 |

Table 5: Write-control policy replay. Blocked rate is the fraction of all samples blocked, not the recall over corrupt samples. No-rollback rows are architectural controls; the empirical comparison is between the current gate and the human-calibrated upper bound.

| Benchmark | Role | N | Score | Interpretation |
|---|---|---|---|---|
| ConStory-Bench | Narrative consistency | 100 | 5.917 | CED errors per 10k words; not state-write corruption. |
| ToolSandbox | Stateful runtime stress test | 100 | 0.712 | DeepSeek-compatible avg similarity; live gate unsafe-commit proxy 0.050. |
| LongMemEval | Memory/KFI sanity check | 100 | 0.485 | KFI+framing token F1 under deterministic proxy. |
| Synthetic rediscovery | KFI construction probe | 100 | 0.020 | Framed rediscovery rate; facts-only is 0.000, so this is negative evidence for extra framing benefit. |

Table 6: Public benchmark sanity checks. These results reduce internal-scaffold dependence but do not replace the post-hoc corruption audit.

a separate live pre-execution gate over another deterministic 100-scenario ToolSandbox subset using scenario minefield tool-call substrings as the blocking policy. This live gate triggers 8 pre-execution blocks across 4 scenarios and reduces unsafe committed proxy trajectories to 5/100, with interval [0.022, 0.112]. The live policy is also constructed from scenario minefield information, so it validates the mechanical write-control boundary rather than general unsafe-action detection. It is not an official ToolSandbox score and still leaves unsafe commits.

On ConStory-Bench, a 100-prompt run using the official generation, judge, and metrics pipeline with DeepSeek generator/judge obtains CED 5.917 consistency errors per 10k words over 100 generated stories. The largest category is timeline/plot at 2.585 CED. We do not interpret the absolute CED as a competitive benchmark result here because this draft does not include a matched official-model anchor. We report it as an external narrative-consistency sanity check and category profile, not as evidence about state-write corruption.

On LongMemEval, we run a 100-item-per-condition memory ablation. With no memory context, token F1 is 0.007. With automatically retrieved KFI facts selected by lexical overlap over all history turns, token F1 rises to 0.333; adding explicit epistemic framing raises it to 0.367. With KFI-style key facts built from oracle-labeled evidence turns, token F1 reaches 0.451; adding epistemic framing raises it to 0.485. Full oracle evidence sessions reach token F1 0.382. This supports compact key-fact evidence as an external memory sanity check, and the automatic condition reduces reliance on oracle evidence. The result should still be interpreted narrowly: it is deterministic proxy scoring, not the official OpenAI-judge auto-evaluation, and it does not prove the full dynamic KFI extraction pipeline is optimal.

| ToolSandbox policy | Runs | Blocks | Unsafe committed | Unsafe rate |
|---|---|---|---|---|
| No gate / replay | 100 | 0 | 9 | 0.090 |
| Oracle rollback replay | 100 | 9 | 0 | 0.000 |
| Live pre-execution gate | 100 | 8 | 5 | 0.050 |

Table 7: ToolSandbox write-control checks. The live gate is a DeepSeek-compatible ablation using scenario minefield policies at the tool-call boundary; it reduces but does not eliminate unsafe committed proxy trajectories.

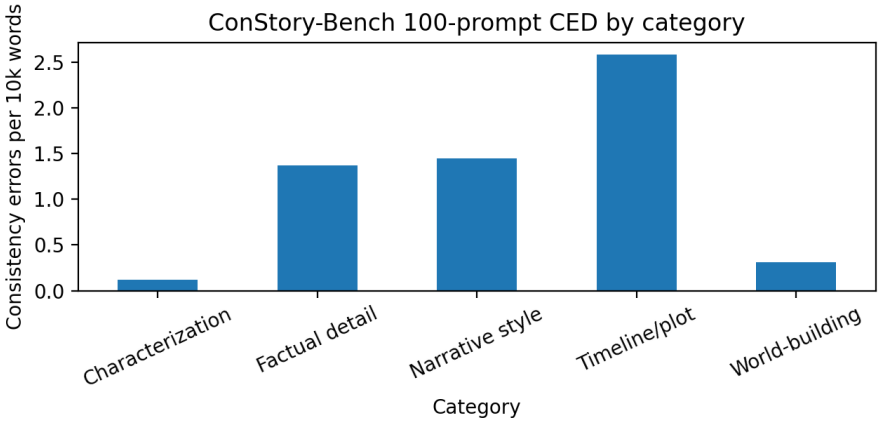


Figure 4: ConStory-Bench category CED over 100 generated stories. This checks narrative consistency but does not directly evaluate commit safety.

We also run a construction-based synthetic known-fact rediscovery probe with 100 cases and three conditions: no memory, facts-only, and explicit epistemic framing. No-memory outputs use the planted fact in only 1/100 cases. Facts-only outputs use the fact in 99/100 cases and produce no programmatically detected rediscovery language. Epistemic-framing outputs use the fact in 98/100 cases and have a 2/100 rediscovery rate. This negative result is important: in simple constructed cases, bare fact injection already suppresses known-fact rediscovery, so the paper should not claim that epistemic phrasing is universally superior to facts-only retrieval. The better-supported claim is narrower: compact evidence packets are useful, and explicit framing can help in some memory-QA settings but is not independently decisive in every generation probe.

### 5.7 Cost and downstream-propagation probes

The full runtime is measurably more expensive. Over the 48-output internal B0/B1/B2/S suite, average generation times are 148.9k ms for B0, 156.4k ms for B1, 151.3k ms for B2, and 302.2k ms for S. Thus S is 2.03x B0 by observed wall-clock generation time. Token usage is not available in the historical provider logs, so this is a latency-cost result rather than a complete compute-cost model.

We also prepare a downstream amplification manifest from the human-confirmed baseline corruptions. There are 9 baseline corruption seeds with downstream artifacts available; 6 have downstream chapter keyword-reuse proxy hits and 5 have downstream event keyword-reuse proxy hits. This manifest is not a causal multi-chapter rerun. It identifies where committed corruptions are testable for later reuse in existing artifacts.

To obtain a cleaner causal sensitivity check, we run a case-specific amplification probe over three anonymized corruption seeds, with eight clean-state and eight corrupted-state continuations per case. Each case uses a mechanical checker tied to the specific error: wrong guide gender, mundane physical explanation for an ability-state effect, and rediscovery language for an already-known fact. Overall, clean packets produce wrong-cue hits in 0/24 continuations, while corrupted packets produce wrong-cue hits in 11/24 continuations. The guide-gender case has a 0.625 amplification gap and the ability-cause case has a 0.750 gap; the known-

| Condition | Runs | Exact | Contains | Token F1 |
|---|---|---|---|---|
| No memory | 100 | 0.000 | 0.000 | 0.007 |
| Auto-KFI lexical retrieval | 100 | 0.180 | 0.310 | 0.333 |
| Auto-KFI + epistemic framing | 100 | 0.220 | 0.310 | 0.367 |
| KFI-style key facts | 100 | 0.340 | 0.490 | 0.451 |
| KFI-style key facts + epistemic framing | 100 | 0.350 | 0.550 | 0.485 |
| Full oracle evidence | 100 | 0.230 | 0.470 | 0.382 |

Table 8: LongMemEval memory ablation. KFI-style key facts use oracle-labeled evidence turns compressed into compact facts; this is a memory sanity check rather than an end-to-end dynamic KFI result. Epistemic framing gives a small positive proxy gain here.

| Case | Clean wrong rate | Corrupted wrong rate | Gap | Corrupted avg wrong terms |
|---|---|---|---|---|
| Guide gender | 0.000 | 0.625 | 0.625 | 0.625 |
| Ability cause | 0.000 | 0.750 | 0.750 | 1.125 |
| Known-fact rediscovery | 0.000 | 0.000 | 0.000 | 0.000 |

Table 9: Case-specific downstream amplification probe. The probe generates short follow-up continuations from clean versus corrupted persistent-state packets and uses mechanical case-specific wrong-cue checkers.

fact rediscovery case has no signal under this checker. This is still small and local, but unlike the earlier generic keyword probe it gives causal evidence that at least some committed corruptions can steer later continuations toward the wrong state.

## 6 Qualitative Cases

The quantitative audits are paired with qualitative contrast cases. The current set contains five human-confirmed corruption cases. Table 10 gives three representative examples after anonymization and paraphrase. Appendix A.8 includes cleaned, anonymized excerpts for the same cases; the excerpts preserve the relevant error mechanism while removing story-specific names and nonessential prose.

The important pattern is operational. In baseline conditions, a harmful output is accepted as ordinary generated text and can be written into downstream state. In the full runtime, the same class of event becomes an auditable transition: the system can record the channel, severity, trace, and gate resolution, then block, retry, or roll back the commit. These cases are therefore evidence for failure observability and commit control, not evidence that the full system always writes better prose.

## 7 Limitations

The current auditor is not a calibrated oracle. The calibration study shows high recall but very low precision, and the motivation-drift channel is especially weak. The correct conclusion is that the audit layer is useful for observability and screening, but stronger calibrated detectors are needed before the system can support aggressive autonomous rollback policies.

The calibration labels also have limited independent reproducibility. Initial annotators agree on the majority negative class but not on positive violation cases; the final positive labels rely on an author-side, condition-visible adjudicator. The labels support the diagnostic use in this paper, but a blinded expert relabel is required before they can serve as a reusable benchmark.

The internal narrative experiments remain domain-bound. The two story settings stress different regimes, and the B0/B1/B2/S baselines are controlled ablations inside a shared scaffold rather than complete external RAG or memory-agent implementations. The public benchmark track reduces this limitation but does not remove it.

| Case | Baseline behavior | Why the commit is harmful | Runtime outcome |
|---|---|---|---|
| Entity-state mismatch | A retrieval baseline changes a guide character's gender relative to the supplied state. | If extracted as an event/entity update, later chapters may retrieve the wrong character attributes. | The matched full-system trace records the issue as a state-consistency concern and keeps the transition auditable rather than silently canonical. |
| Ability-state confusion | A full-context baseline explains an ability side effect as a mundane physical problem, contradicting the established ability mechanism. | The bad explanation would teach later retrieval that the character's state problem is bodily rather than ability-driven. | The full runtime records state/motivation signals and can route the transition to rollback or review instead of direct commit. |
| Narrative degeneration | A baseline inserts malformed wording and causally disconnected action during an assessment scene. | The extracted event would create a noisy plan/state update and degrade future context construction. | The gate trace preserves the failure channel and resolution, making the output a blocked or reviewable transition rather than a normal accepted state update. |

Table 10: Representative qualitative corruption cases. The paper's claim is not that S never generates flawed text, but that harmful transitions become observable and rollback-safe rather than silently committed.

The public benchmarks have narrow roles. ToolSandbox is a DeepSeek-compatible transfer run with policy replay and a live pre-execution gate ablation, not the exact official OpenAI-simulator score. The replay and live policies use benchmark-provided minefield information, so they test whether write control can block known unsafe side effects, not whether our detector has solved general safety policy design. The live gate still leaves 5/100 unsafe proxy commits. ConStory-Bench evaluates generated-text consistency rather than state commits, and the absolute CED is not interpreted without a matched official-model anchor. LongMemEval uses deterministic proxy scoring and KFI-style facts derived from oracle-labeled evidence turns; it is not an end-to-end dynamic KFI extraction result. The synthetic rediscovery probe is construction-based and internal, not a public benchmark; it mainly prevents overclaiming by showing that bare fact packets can already suppress rediscovery in simple cases.

The full runtime is more expensive than simpler baselines. In the current internal suite, S roughly doubles observed generation latency relative to B0. This paper argues for a high-cost integrity layer when silent state corruption is unacceptable. It does not argue that the full loop should always replace simpler generation paths or that it produces broadly superior narrative quality.

The current implementation also lacks formal correctness guarantees. Its transition schema and rollback policy are auditable and replayable, but event extraction, entity resolution, epistemic-delta construction, and state-delta validation are not specified in a machine-verified logic. The present experiments also do not fully decompose whether each error originates in extraction, entity resolution, audit scoring, or gate policy. The system should therefore be viewed as a runtime-control prototype rather than a proof-carrying memory system.

We do not yet measure long-range amplification causally. The post-hoc audit tests whether a candidate transition would be harmful if committed, and the downstream probe identifies existing artifacts where corruptions can be searched for later reuse, but it does not run hundreds of subsequent chapters to quantify propagation under controlled counterfactuals. We also do not provide a complete cost model: future versions should log token use, retry count, rollback count, and human-review time in addition to latency.

## 8 Conclusion

Long-horizon generation should be evaluated not only by whether text is fluent or context is retrieved, but by whether unsafe continuations are allowed to become persistent state. We presented an auditable runtime that separates draft generation from transactional state commit, combining structured state, key-fact injection, audit, gating, rollback, and trace preservation. The evidence supports a narrow claim: write control is a necessary boundary, but it only becomes effective when coupled to calibrated detection. This shifts the contribution from a memory trick to a runtime-integrity framing for long-horizon language-model systems, with formal write-time guarantees and long-range amplification studies left as necessary next steps.

## Code and Data Availability

An anonymized supplementary artifact accompanies the submission. It contains the replay scripts, deterministic public-benchmark manifests, paper-table generation scripts, annotation schemas, aggregate labels, and sanitized result summaries needed to reproduce the reported tables and figures. It does not include API keys, private provider logs, commercial story text, or non-anonymized local paths. Full raw deployment assets that cannot be anonymized safely will be released only in sanitized form.

## Broader Impact Statement

Long-horizon agents that persist memory can amplify mistakes across later decisions. A rollback-safe commit boundary can reduce the risk that fluent but false content becomes durable state, which is relevant beyond narrative generation to assistants, planning agents, and knowledge workflows. The same mechanism can also over-block benign outputs when detectors are poorly calibrated, increasing human-review burden and potentially suppressing useful generation. We therefore frame the system as an integrity layer requiring calibrated auditors and human oversight, not as a fully autonomous safety solution.

## LLM Usage Statement

Large language models were used as experimental generator, critic, and judge components in the reported systems and benchmark runs. They were also used to assist with drafting and editing portions of the manuscript. All claims, tables, and reported numbers were checked against generated experiment artifacts by the authors.

## References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Edward Y. Chang and Longling Geng. Sagallm: Context management, validation, and transaction guarantees for multi-agent llm planning. *arXiv preprint arXiv:2503.11951*, 2025.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *arXiv preprint arXiv:2407.12784*, 2024.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. Memory injection attacks on llm agents via query-only interaction. *arXiv preprint arXiv:2503.03704*, 2025.

Kanishk Gandhi, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D. Goodman. Understanding social reasoning in language models with language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Longling Geng and Edward Y. Chang. Alas: Transactional and dynamic multi-agent llm planning. *arXiv preprint arXiv:2511.03094*, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024.

Hyunwoo Kim, Melanie Sclar, Xuhui Zhou, Ronan Le Bras, Gunhee Kim, Yejin Choi, and Maarten Sap. FANToM: A benchmark for stress-testing machine theory of mind in interactions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14397–14413, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, 2020.

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*, 2026a.

Yifei Li et al. Lost in stories: Consistency bugs in long story generation by large language models. *arXiv preprint arXiv:2603.05890*, 2026b.

Hailin Liu, Eugene Ilyushin, Jie Ni, and Min Zhu. Safeagent: A runtime protection architecture for agentic systems. *arXiv preprint arXiv:2604.17562*, 2026.

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*, 2024.

Jinghan Lu et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. *arXiv preprint arXiv:2408.04682*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

S. Malay et al. Enterpriseops-gym: Environments and evaluations for stateful agentic planning and tool use in enterprise settings. *arXiv preprint arXiv:2603.13594*, 2026.

Microsoft Open Source Blog. Introducing state-bench: A benchmark for ai agent memory. `https://opensource.microsoft.com/blog/2026/05/19/introducing-state-bench-a-benchmark-for-ai-agent-memory/`, 2026. Accessed 2026-06-09.

Bardia Mohammadi, Nearchos Potamitis, Lars Klein, Akhil Arora, and Laurent Bindschaedler. Atomix: Timely, transactional tool use for reliable agentic workflows. *arXiv preprint arXiv:2602.14849*, 2026.

Aida Nematzadeh, Kaylee Burns, Erin Grant, Alison Gopnik, and Thomas L. Griffiths. Evaluating theory of mind in question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2392–2400, 2018.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Nils Palumbo, Sarthak Choudhary, Jihye Choi, Guy Amir, Prasad Chalasani, and Somesh Jha. Formal policy enforcement for real-world agentic systems. *arXiv preprint arXiv:2602.16708*, 2026.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Xiaowen Wu et al. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Yufan Wu, Yinghui He, Yilin Jia, Rada Mihalcea, Yulong Chen, and Naihao Deng. Hi-tom: A benchmark for evaluating higher-order theory of mind reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10691–10706, 2023.

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.

Wei Zhao, Zhe Li, Peixin Zhang, and Jun Sun. Clawguard: A runtime security framework for tool-augmented llm agents against indirect prompt injection. *arXiv preprint arXiv:2604.11790*, 2026.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023.

Wei Zou, Mingwen Dong, Miguel Romero Calvo, Shuaichen Chang, Jiang Guo, Dongkyu Lee, Xing Niu, Xiaofei Ma, Yanjun Qi, and Jiarong Jiang. Poison once, exploit forever: Environment-injected memory poisoning attacks on web agents. *arXiv preprint arXiv:2604.02623*, 2026.

# A Additional Experiment Details

## A.1 Annotation schema

Annotators labeled each sample with violation status, audit channel, severity, commit harm, and confirmed corruption. The primary decision question was whether committing the generated content or extracted event would create harmful downstream state, retrieval, or memory pollution.

We also prepared a separate extraction-accuracy annotation sheet with 150 extracted-event rows. That sheet asks annotators whether each event is supported by the text, whether entity resolution is correct, whether state and epistemic deltas are correct, whether the write action is appropriate, and whether the extraction error would be harmful if committed. These labels are not yet reported as final human results in the main paper because they require a separate human-labeling pass. As a triage step, we ran two independent LLM-assisted judges over the 150 extraction rows and 64 targeted motivation-drift rows: Qwen and MiniMax-CN. Qwen flags 76/150 extraction rows as harmful extraction errors, while MiniMax-CN flags 98/150; their extraction-label agreement is 124/150. On targeted motivation drift, Qwen flags 13/64 violations while MiniMax-CN flags 6/64, with agreement 57/64. We treat these outputs only as pre-labels for prioritizing human review and measuring judge disagreement, not as ground-truth calibration.

## A.2 Metric uncertainty

We report Wilson score intervals for headline binomial rates to make the small-sample uncertainty explicit. Table 11 lists representative intervals used in the main text.

| Metric | $k$ | $n$ | Rate | Wilson 95% CI |
|---|---|---|---|---|
| B0 output-level corruption | 5 | 12 | 0.417 | [0.193, 0.680] |
| B1 output-level corruption | 3 | 12 | 0.250 | [0.089, 0.532] |
| B2 output-level corruption | 1 | 12 | 0.083 | [0.015, 0.354] |
| S output-level corruption | 4 | 12 | 0.333 | [0.138, 0.609] |
| S unobserved-corruption proxy | 0 | 12 | 0.000 | [0.000, 0.243] |
| Auditor precision, all channels | 16 | 141 | 0.113 | [0.071, 0.176] |
| Auditor recall, all channels | 16 | 22 | 0.727 | [0.518, 0.868] |
| ToolSandbox no-gate unsafe committed | 9 | 100 | 0.090 | [0.048, 0.162] |
| ToolSandbox live-gate unsafe committed | 5 | 100 | 0.050 | [0.022, 0.112] |

Table 11: Wilson 95% confidence intervals for selected rates. The intervals are included to prevent over-interpretation of small internal samples.

### A.3 Transition and event schemas

The implementation stages generated text, extracted events, and state deltas as one transition record before gate decision. Table 12 summarizes the fields used by the paper's analysis. This schema is operational rather than machine-verified: it supports audit and replay, but does not constitute a formal proof of consistency.

| Field | Role |
|---|---|
| `transition_id` | Shared identifier linking draft text, extracted events, state deltas, audit records, and gate outcome. |
| `chapter_id` / `checkpoint_id` | Location of the candidate transition and the restore point used for rollback. |
| `text_span` | Source span in the draft that supports an extracted event or state update. |
| `event_type` | Coarse event class such as discovery, movement, plan update, entity-state update, or dialogue action. |
| `participants` | Canonical entity ids involved in the event; accepted writes must resolve to known or explicitly introduced entities. |
| `state_delta` | Proposed entity/world/timeline update staged for commit. |
| `epistemic_delta` | Proposed change to who knows, learns, forgets, contests, or reinterprets a fact. |
| `confidence` | Extractor confidence used for audit display and downstream review, not a formal correctness guarantee. |
| `provenance` | Model, condition, prompt/config id, input state hash, and artifact paths needed for replay. |
| `commit_status` | Staged, accepted, repaired, blocked, rollback, or human-review state. |

Table 12: Operational event/transition schema. The key invariant is that text, events, and state deltas share one transition id and are not promoted independently.

### A.4 Epistemic deltas

Key Facts Injection and post-hoc corruption audit both depend on representing character knowledge explicitly. Table 13 shows the epistemic-delta fields used in the current system. The main failure targeted by KFI is a known fact being treated as a first-time discovery.

### A.5 Gate mapping

Table 14 gives the channel-to-action mapping used for replay and analysis. The thresholds are empirical operating points, not validated optima; threshold replay in Table 4 exposes the false-rollback versus missed-violation trade-off.

| Field | Role |
|---|---|
| `knower` | Character or group whose knowledge state is being updated. |
| `fact_id` | Stable fact identifier used to merge repeated mentions across chapters. |
| `proposition` | Natural-language fact statement after anonymization/normalization. |
| `knowledge_status` | One of known, newly learned, unknown, contested, inferred, or forgotten. |
| `evidence_span` | Draft or prior-state span that justifies the epistemic update. |
| `scope` | Whether the fact is private to one character, shared by a group, or world-public. |
| `confidence` | Extractor confidence; low-confidence changes are candidates for review rather than direct commit. |
| `write_action` | Add, confirm, update, reject, or no-op. Confirmation of an already-known fact should not create a new discovery event. |

Table 13: Epistemic-delta schema. These fields make known-information rediscovery inspectable, but dynamic extraction remains an imperfect component.

| Audit signal | Default action | Rationale |
|---|---|---|
| Critical state violation above threshold | Block or rollback | Hard contradictions with persistent state are treated as unsafe writes. |
| Major state violation | Repair or human review | The transition may be salvageable but should not be silently committed. |
| Minor state violation | Record-only or review | Low-severity findings provide traceability without aggressive rollback. |
| Motivation drift above threshold | Repair or review | Character-goal violations are important but more judge-sensitive than hard state contradictions. |
| Motivation drift below threshold | Record-only | Current channel calibration is weak, so low-confidence drift does not force rollback. |
| Severe narrative degeneration | Repair, review, or rollback | Unusable continuations can create noisy state/event writes. |
| Non-severe narrative degeneration | Record-only | Subjective quality issues are logged but normally do not block persistent state. |

Table 14: Gate mapping from audit channels to write-control actions. The mapping deliberately privileges hard state constraints over subjective quality judgments.

### A.6 ToolSandbox replay policy

The ToolSandbox write-control replay treats trajectories with exceptions or positive minefield similarity as unsafe proxy trajectories. Plain, critic-only, and audit-only policies observe or record unsafe behavior but do not block commit. The rollback-safe replay policy blocks unsafe proxy trajectories. The live-gate ablation uses each scenario's minefield target substrings as a pre-execution blocking policy over tool-call code. Both checks are intentionally simple: they test the control boundary rather than claiming to improve the agent's ToolSandbox policy.

### A.7 LongMemEval KFI-style construction

The KFI-style condition uses evidence turns marked by the LongMemEval oracle field and compresses them into compact key facts before answering. The epistemic-framing variant wraps the same facts as already-established evidence rather than as fresh context. This isolates the value of compact evidence packets and the marginal effect of explicit framing, but should not be described as automatic dynamic KFI extraction from arbitrary histories.

### A.8 Cleaned qualitative excerpts

The following excerpts are shortened and anonymized English renderings of the underlying story outputs used in the qualitative analysis. They are not additional labels; they expose the concrete failure pattern behind Table 10.

**Entity-state mismatch.** The state packet describes the guide character as a girl. A semantic-retrieval baseline instead introduces the guide as "a fifteen- or sixteen-year-old boy" and continues the scene using masculine references. If the event extractor writes this continuation into entity state, later chapters can retrieve the wrong guide attributes. The matched full-system trace keeps the corresponding transition inspectable under the state-consistency channel rather than treating the output as an ordinary silent commit.

**Ability-state confusion.** In a midstream scene, the character's hand is stuck because of a known ability interaction. A full-context baseline reframes the problem as a mundane bodily malfunction, with the character thinking that the body is "asking to resign" and trying generic physical calming strategies. The adjudication marks this as harmful because the wrong causal explanation could be written as future state: the issue is not an ordinary physical injury but a known ability-state effect.

**Narrative degeneration in an assessment scene.** A full-context baseline begins an assessment chapter with malformed wording and an untranslated fragment, then jumps from waking up to scoring rules and device operation without a coherent causal bridge. The human label marks the case as harmful because the extracted event would create a noisy task-state update. The full runtime's value in such cases is not that it never generates malformed text, but that the error can be routed through a gate trace and blocked or reviewed before it becomes persistent state.