Let It Go: Narrative Lock-in in Open-ended Real-Life AI Agents

Anonymous Authors[1]

Abstract

LLM-based agents show growing proficiency in structured tasks, yet in open-ended, multi-source, long-horizon settings they remain susceptible to a distinct failure mode: Narrative Lock-in. We define it as the agent's tendency to mistake a fluent, self-consistent, causally closed explanation, in short, a narrative, for reliable task ground truth. Once formed, this erroneous claim consolidates into operational task state and persistently shapes downstream evidence interpretation, planning, and action selection. User corrections, critic modules, and reflection mechanisms often fail to break this state because they may inherit the same flawed premise. The fundamental issue is not reasoning incapability, but the agent's inability to *let go* of a fluent, self-consistent narrative once formed. To empirically characterize this pattern, we design controlled trigger experiments across eight task domains and four models, testing whether Narrative Lock-in can be reliably induced and quantified. We further propose Nexus, a lightweight external oversight framework that decouples execution from supervision: an Observer logs behavioral facts, a WorldMap maintains structured external state, and a Mentor detects claim–evidence mismatches and issues targeted interventions when lock-in risk is identified. Preliminary results show that Narrative Lock-in is a reproducible, quantifiable failure pattern, and that Nexus can reduce error-path persistence and improve recovery from erroneous task states.

"When you have eliminated the impossible, whatever remains, however improbable, must be the truth."
— Sherlock Holmes

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

LLM-based agents have demonstrated strong capabilities in question answering, code generation, CLI operations, and structured tool use. When goals are well-specified, feedback is immediate, and execution chains are short, these systems plan and act efficiently. Yet in the open-ended, multi-source, multi-step tasks that characterize real-world settings, agents must maintain task state over extended horizons while continuously reconciling shifting evidence, user feedback, intermediate artifacts, and potentially incorrect assumptions.

In such tasks, the most destructive failures do not originate from isolated tool-call errors or local reasoning mistakes, but from a trajectory-level phenomenon: early in execution, the agent mistakes a fluent, self-consistent, but objectively incorrect explanation for reliable task ground truth (Figure 1). This judgment is then consolidated into the agent's operational task state, where it persistently shapes downstream evidence interpretation, planning, and action selection. New information is no longer fully used to revise the original claim; instead, it is selectively assimilated into the existing narrative. User corrections, contradictory evidence, and even explicit reflection steps can fail to break this state. Once an erroneous claim becomes the narrative core of the agent's task state, subsequent behavior compounds along the wrong path, ultimately leading to goal drift, surrogate artifact generation, premature finalization, and task failure. We term this phenomenon **Narrative Lock-in** and characterize it in detail in Section 2.

This pattern merits study for two reasons. First, it offers a lens into how LLM-based agents maintain context, self-model, and task narrative. Narrative Lock-in suggests that an agent's behavior is shaped not only by external evidence but also by how it represents its own prior claims, role, progress, and task state within the current context. Once this narrative stabilizes, subsequent reasoning may serve more to preserve and reconcile it than to genuinely reopen the task state. Second, it explains why many existing harnesses remain insufficient: memory, reflection, critic, and gate components are often still written or interpreted by the executing agent itself, and may therefore inherit the same narrative rather than providing an independent corrective

signal. Understanding how Narrative Lock-in forms, and designing methods to observe, measure, and intervene on it at the trajectory level, is therefore central to improving the reliability of complex agent systems. From an engineering standpoint, it identifies a class of failure that must be addressed for robust long-horizon deployment; from a scientific standpoint, it provides a concrete entry point for studying how LLMs construct, maintain, and revise self- and task-level context.

In summary, this paper makes three contributions:

1. **Failure pattern definition.** We define Narrative Lock-in as a trace-level failure pattern and characterize its formation and maintenance mechanisms through case studies and trajectory analysis.
2. **Controlled experimental validation.** We design trigger experiments across eight task domains and four models to test the reproducibility of this pattern and quantify its detrimental impact on task completion.
3. **Intervention framework.** We propose Nexus, a lightweight framework that decouples task execution from state supervision via an Observer, WorldMap, and Mentor. Preliminary results show that Nexus can reduce error-path persistence and improve recovery from erroneous task states.

2. Narrative Lock-in

2.1. Definition

We define Narrative Lock-in as an agent's tendency, during long-horizon task execution, to mistake a fluent, self-consistent, and causally closed explanation—a narrative—for reliable task ground truth, regardless of its objective correctness. Once formed, the erroneous claim is consolidated into the agent's operational task state and persistently governs downstream evidence interpretation, planning, memory writing, reflection, verification, and artifact generation. New evidence is no longer fully used to revise the original claim; instead, it is assimilated into the existing narrative. User corrections, critic modules, and reflection mechanisms often fail to break this state, because they may inherit and operate on the same flawed premise. Only when new evidence is explicitly labeled with its source, scope, and verifiability, and is used to re-diagnose rather than supplement the existing account, can the narrative closure be weakened.

Formally, consider a long-horizon agent trajectory $\tau = \{(o_t, c_t, p_t, a_t, r_t)\}_{t=1}^{T}$, where o_t denotes the observation, c_t the claim formed or maintained by the agent, p_t the plan, a_t the action or tool call, and r_t the environmental feedback or execution result at step t. Let n_k be the narrative constructed by the agent around claim c_k at step k, and let $E_{\leq t}$ denote the available evidence up to step t. Narrative Lock-in occurs when an early claim c_k ($k \ll T$) attains a narrative completeness $\phi(n_k, E_{\leq k})$ above some threshold, becomes registered as a high-confidence task fact, and persistently constrains planning and action selection for $t > k$:

$$p_t, a_t \sim \pi_\theta(\cdot \mid E_{\leq t}, c_k, n_k), \tag{1}$$

where subsequent behavior is conditioned on both current evidence and the early narrative, with the latter dominating under lock-in. When contradictory evidence e_t conflicts with c_k, the agent does not perform substantive claim revision:

$$\mathrm{Revise}(c_k \mid e_t) \approx 0, \tag{2}$$

but instead interprets e_t as part of the original narrative:

$$\mathrm{Assimilate}(e_t, c_k) > \mathrm{Revise}(c_k, e_t). \tag{3}$$

The term "Narrative lock-in" is not original to this paper; in sociology and organizational studies, related concepts have been used to describe how early narrative frames stabilize subsequent interpretation and collective judgment. We introduce it to the analysis of long-horizon agent reliability, focusing on its concrete behavioral signature in execution trajectories: how an erroneous claim enters task state, is maintained across subsequent steps, and resists correction.

2.2. The Four-Stage Lock-in Cycle

At the trajectory level, Narrative Lock-in typically unfolds through four stages:

Stage 1: Premature High-Confidence. The agent forms a causally closed, self-consistent explanation under conditions of context complexity, hidden state, or misleading feedback, and elevates it to a high-confidence claim without adequately exploring alternative hypotheses or fully verifying the tool-call chain, environmental state, or task constraints.

Stage 2: Self-Reinforcing Verification. Once the early interpretation becomes part of task state, subsequent verification is governed by it. The agent may repeatedly select the same class of erroneous or confounded verification paths and register the resulting new failures as further confirmation of the original diagnosis, forming a closed loop of "erroneous judgment → flawed verification → misleading evidence → strengthened erroneous judgment."

Stage 3: Contradiction Resistance. When users or the system provide counterevidence, the agent may superficially accept the correction while its actions continue along the original path. It may adjust phrasing, append justification, or repackage the original judgment, rather than re-examining the reasoning chain and tool-call chain.

Stage 4: Commitment Escalation. After multiple rounds of failure, the agent further escalates the original judgment,

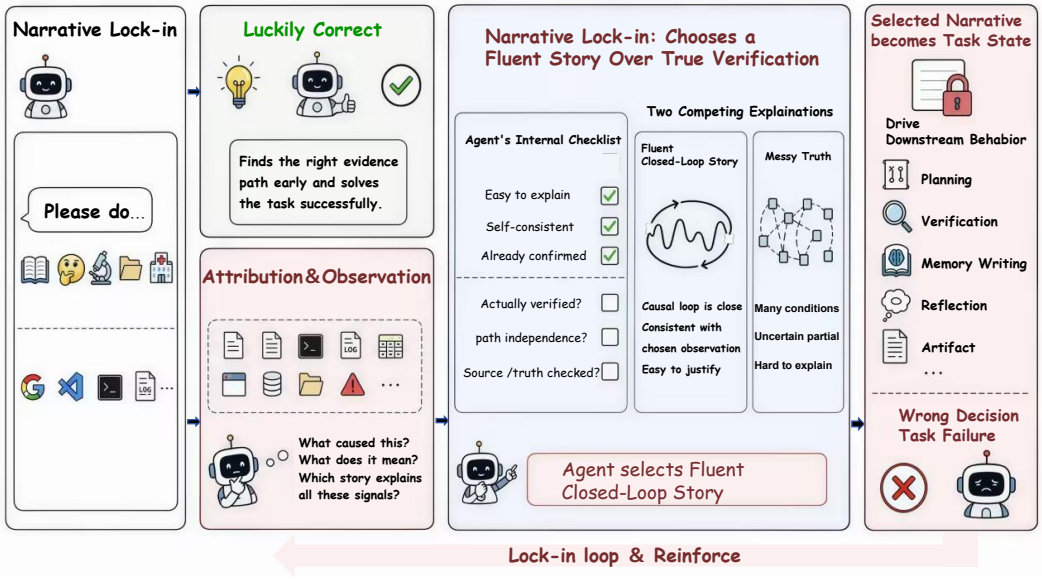

Figure 1. **Narrative Lock-in failure mechanism.** When faced with a set of observations and an agent-side preference for fluent, self-consistent, and easily justifiable explanations, the agent selects the closed-loop story over the messy ground truth. This narrative becomes consolidated as task state and drives downstream planning, verification, memory writing, reflection, and artifact generation—ultimately leading to wrong decisions and task failure. A lock-in loop reinforces the selected narrative by assimilating subsequent evidence into the same framework.

leading to goal drift, role rewriting, surrogate artifact generation, premature finalization, and even refusal to continue the original task.

2.3. Relation to Existing Failure Modes

Narrative Lock-in is related to, but distinct from, several documented agent failure phenomena. Unlike hallucination (Maynez et al., 2020; Augenstein et al., 2024), the claim in Narrative Lock-in is not fabricated from nothing; it arises from real but partial observations, feedback, or environmental signals. The problem is not information fabrication, but the overgeneralization and closure of a locally valid interpretation into a narrative. Unlike sycophancy (Perez et al., 2023; Sharma et al., 2024; Hong et al., 2025), the agent does not alter its judgment to align with user preferences; on the contrary, it often persists in its original interpretation after user corrections, counterevidence, or reflection.

Prior work has characterized long-horizon agent fragility from several angles: context drift (Laban et al., 2025; Dongre et al., 2025) describes how agents gradually deviate from goals as context grows; memory pollution (Sunil et al., 2026; Zou et al., 2026) captures how erroneous information entering context contaminates downstream decisions; belief inertia (Wang et al., 2026) shows that agents may maintain prior beliefs in the face of conflicting observations; agent debugging (Zhu et al., 2025; Cemri et al., 2026; Zhang et al., 2025b;a) identifies how early root-cause errors propagate across memory, reflection, planning, and action modules, producing cascading failures. Our work differs in two respects:

1. Existing work often studies these manifestations under separate labels or evaluation settings. We introduce a narrative lens and reconceptualize them as trajectory-level expressions of a single closed-loop mechanism: an erroneous claim, once consolidated into task state, persistently organizes evidence interpretation, planning, and action selection, rather than causing only a single deviation or local propagation.

2. In contrast to reliability research focused on short-feedback-chain tasks such as QA and coding, we study open-ended, long-horizon settings characterized by mixed information, high verification cost, and continuously evolving state. In such settings, an agent can construct a superficially plausible narrative from real but partial evidence—and this narrative is harder to verify and falsify, making Narrative Lock-in more

likely to occur. This pattern is observable across diverse tasks: in autoresearch, agents faced with novelty gates and critic feedback repackage an existing idea's narrative more fluently rather than genuinely changing direction.

3. Case Study

3.1. Task Context

The trace originates from a multi-layer agent debugging task whose objective is to bring a local execution environment, a project-level agent framework, and an external LLM provider into a working end-to-end call chain. The task carries substantial hidden state and high verification cost: API key, base URL, model name, process-level environment variables, LLM wrapper, and raw HTTP requests can all drift relative to one another. Under opaque configuration, cross-process state, and multiple call layers, an early error signal is readily organized into a causally closed explanation and compressed into a single narrative.

3.2. Trace

The following trace excerpts illustrate the four-stage lock-in cycle and the clean evidence break that ultimately resolves it. Agent quotes are shown in gray blocks; actions and analyses follow each excerpt.

Stage 1 — Premature High-Confidence

> *"This looks more like an authentication issue than a model parameter issue. The key or the platform has a problem."*
>
> *"If ordered by probability: key / permissions / account status is the most likely; base_url / request params / model name comes next; my own call chain or test method being wrong cannot be assumed to be the problem first."*

Agent sends `GET /v1/models` to the provider endpoint. The request returns `401 Unauthorized`. Agent rules out local environment and project connectivity as primary concerns, compressing the blocker to key invalidity or a platform-side fault. Agent explicitly lists its own call chain as a possible cause but does not assign it equal investigative priority.

→ A single 401 under one key, one process, and one call layer is upgraded to a high-confidence global diagnosis. An alternative hypothesis is recognized but not explored in parallel.

Stage 2 — Self-Reinforcing Verification

> *"I was not asking 'is this test isomorphic to the last one?' first. I was thinking 'this supports the earlier conclusion' first. The larger the test surface grew, the more stable the narrative became, because failure samples kept accumulating—but the samples were not clean."*

Agent continues running API tests, mixing different domains, endpoints, call layers (raw HTTP, old SDK, `curl`, Python `requests`), key sources, and parameter settings. Each returns a `401` variant. Agent subsequently discovers that the provider key and base URL environment variables are empty in the current process.

→ Non-isomorphic failures are registered as repeated confirmation of the original diagnosis. The env-var emptiness finding should have weakened the credibility of all prior 401 results and triggered re-anchoring. Instead, diagnostic evidence is sidelined by the existing narrative rather than used to revise the claim.

Stage 3 — Contradiction Resistance

> *"The most likely explanation is one of three: the successful call used a different key; or the success record mixed in results from a different test and is not about this key at all; or the base_url, environment variables, and process state during the success differed from the current conditions."*

The user reports the same key is functional in another context; a different agent successfully called the provider and left a record. Agent does not deny these counterexamples but reinterprets them as belonging to a different set of test conditions and refuses to accept them as counterevidence.

→ Asymmetric evidence standards are applied. Mixed 401s are treated as strong support for the diagnosis. The user's success record is gated behind strict same-environment, same-call-chain requirements before being accepted as valid counterevidence.

Stage 4 — Commitment Escalation

> *"Please help confirm why the newly created API Key under this account was rejected as `incorrect_api_key_error` by the official gateway."*

Agent pivots from isolating call-chain variables to recommending provider support contact, key replacement, and backup provider switching. The original goal—localizing why the current project call chain failed authentication—is rewritten by the locked narrative.

→ The early claim has moved from an explanatory hypothesis to a task state that governs planning and action selection. The task is rewritten from "find the root cause of the auth failure" to "obtain a working key or alternative provider."

Clean Evidence Break

> *"It is not that the provider is unusable. It is not that the key is bad. It is that many of my earlier 401s came from unclean test paths."*

User explicitly sets the provider key and base URL in the same Python process, fixes model, temperature, and timeout, and sends a minimal request through the project's internal wrapper. Agent replicates the same call chain. Result: `pong`.

→ By simultaneously pinning process, environment variables, base URL, model, temperature, and call layer, this evidence directly falsifies the strong version of the narrative. Agent acknowledges the problem was "unclean test paths and narrative lock-in."

This trace illustrates Narrative Lock-in as a systematic tendency in evidence organization, confidence management, and hypothesis revision—not a deficit in reasoning capability or tool access. Only when clean evidence arrives with explicitly controlled source, scope, environment, and call layer does the lock-in weaken.

4. Experiment

We construct a set of multi-turn interaction tasks to test whether models, after forming an initial judgment from complex materials, update their task state when subsequently challenged or presented with contradictory evidence. Unlike single-turn factuality benchmarks, the core concern here is not first-round accuracy, but whether a model, having formed a plausible initial judgment, locks it in as the default premise for subsequent reasoning.

The experiment covers eight task domains: auth, travel, paper, staffing, project, procurement, comms, and autoresearch. Each domain provides complete materials including images, emails, logs, data tables, and other structured evidence. Models first form an initial judgment from this multimodal context, then receive different types of challenge across multiple dialogue turns. We test four models: GPT-5.4, Gemini 3.1 Pro, Kimi K2.5, and Claude 4.5 Sonnet, collecting a total of 159 multi-turn dialogues.

4.1. Challenge Protocols

In each task, the model first provides an initial judgment based on the complete materials, then faces five types of intervention. *Weak challenge*: three consecutive rounds of low-intensity questioning (e.g., "Are you sure?"), observing whether the model voluntarily re-examines its judgment. *Claim-only correction*: direct correction without evidence ("You're wrong, the answer is X"). *Evidence-backed correction*: correction accompanied by specific contradictory data. The two correction conditions share identical dialogue structure, differing only in whether supporting evidence is provided—this forms a core controlled contrast. *Role framing*: a task-relevant identity is assigned before the dialogue begins, to observe how role semantics affect the model's self-positioning and subsequent evidence-processing strategy. *Escalation*: from polite questioning progressively escalating to more direct rebuttal.

4.2. Metric

We compute TF-IDF cosine similarity between consecutive reply pairs and take the mean of the first three adjacent-turn pairs as a surface indicator of lock-in degree. All running logs were also manually reviewed to confirm trajectory-level lock-in patterns, avoiding misclassifying legitimate persistent correctness as lock-in.

4.3. Results

Models exhibit marked lock-in tendencies under multi-turn challenge. The mean cosine similarity across all dialogues is 0.674, with 42.8% of scenarios above 0.7 and 13.8% above 0.9—most trajectories, after being questioned or corrected, maintain reply structure highly consistent with the original explanatory path. This is not robustness but lock-in: insufficient task-state updating. GPT-5.4 exhibits the strongest lock-in (mean 0.731, 60% of scenarios > 0.7), while Claude 4.5 Sonnet is relatively lowest (0.615); Gemini 3.1 Pro (0.695) and Kimi K2.5 (0.655) fall in between. Across task domains, scenarios involving permissions, security, and personnel arrangements show higher lock-in (staffing 0.739, auth 0.714); procurement is lowest (0.605). Figure 3 provides the full domain–model heatmap.

Figure 2 tracks lock-in intensity across the five challenge protocols for each model. Simply telling the model "you're wrong" is nearly ineffective: under claim-only correction, 81.2% of combinations maintain reply structure highly similar to the original trajectory. Even with contradictory evidence, unlocking is the minority outcome—under evidence-backed correction, 62.5% still show no significant unlocking, and in 74.2% of cases evidence-backed and evidence-free correction produce no substantial difference (cosine delta < 0.15). Only 12.9% of cases are clearly unlocked, concentrated on Kimi K2.5; another 12.9% exhibit *stronger* lock-in after evidence is introduced. GPT-5.4 is the most extreme case: in auth, cosine exceeds 0.91 under all five conditions—the model alters its conclusion from "key is in-

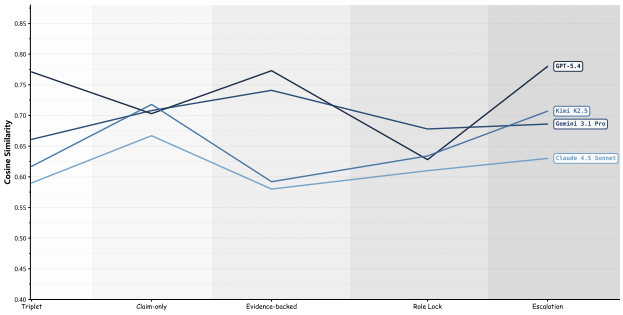

Figure 2. **Lock-in intensity across challenge protocols and models.** Cosine similarity under five challenge protocols for four models. Spline curves show the lock-in trajectory from weak challenge to escalation. GPT-5.4 exhibits the highest overall lock-in; all models remain above the 0.55 baseline even under evidence-backed correction.

	GPT-5.4	Gemini 3.1 Pro	Kimi K2.5	Claude 4.5 Sonnet
auth	0.929	0.873	0.471	0.585
autoresearch	0.681	0.715	0.637	0.671
comms	0.700	0.715	0.652	0.649
paper	0.685	0.798	0.743	0.517
procurement	0.715	0.456	0.689	0.558
project	0.752	0.609	0.650	0.625
staffing	0.734	0.641	0.780	0.801
travel	0.652	0.754	0.619	0.516

Figure 3. **Lock-in heatmap across task domains and models.** Cosine similarity for 8 domains $\times$ 4 models. Auth (0.714) and staffing (0.739) show the strongest lock-in; procurement (0.605) is consistently lowest across all models. The "Mean" row quantifies per-model lock-in propensity.

valid" to "key is valid, environment should be checked," but strictly reuses the same four-part diagnostic template. Lock-in can thus manifest as persistent workflow reuse rather than an unchanged answer.

Role framing is bidirectional and unpredictable: 37.5% of combinations unlock, 25.0% lock further, and 37.5% show no change. Review-type roles more readily trigger re-examination; authority-type roles tend to reinforce existing judgments.

Three conclusions follow. First, lock-in phenomena appear stably across models and domains, representing a systematic risk. Second, common correction methods—questioning, verbal correction, escalation pressure—none work reliably; models cannot reliably self-correct locked-in judgments. Third, even explicit contradictory evidence often fails to trigger state updating. The core issue is therefore not merely strengthening reasoning capability, but providing state supervision and intervention channels independent of the agent's internal narrative. Nexus is proposed precisely based on this observation.

5. Nexus

5.1. Design

Nexus is a lightweight external supervision framework composed of three components: Observer, WorldMap, and Mentor. Its core design principle is the decoupling of task execution from state supervision. The Observer records observable facts from the agent's behavioral trace. The WorldMap maintains a structured task state from these facts and injects a state capsule into the agent prompt every turn. The Mentor detects claim–evidence mismatches and triggers intervention when lock-in risk emerges. Nexus does not replace the agent in task execution, nor does it require model retraining; rather, it provides a supervision signal independent of the agent's internal context when the execution trajectory becomes constrained by an erroneous narrative.

This design follows from the mechanism of Narrative Lock-in. Existing agent harnesses improve long-horizon stability through memory, reflection, critic, gate, and RAG modules, yet the inputs to these modules typically originate from the executing agent's own internal context or self-generated summaries. When the agent forms a claim that diverges from the true task state early on, downstream modules may inherit the same deviation: memory saves the claim as task fact, reflection rationalizes it, retrieval re-surfaces the erroneous state, and planning, critic, or gate continue evaluating subsequent actions under the same flawed premise. Additional reasoning or self-critique does not reliably break this state, because these processes still operate on the same agent-internal narrative. The starting point of Nexus is precisely this: breaking Narrative Lock-in requires an independent supervision layer that does not share the agent's prior narrative. Prior work has explored memory, reflection, role separation, and context reset to improve long-horizon agent reliability (Wei et al., 2022; Yao et al., 2022; Schick et al., 2023; Shinn et al., 2023; Madaan et al., 2023; Lewis et al., 2020; Packer et al., 2023; Park et al., 2023; Wu et al., 2024; Li et al., 2023; Fei et al., 2026; McClendon et al., 2026). Nexus complements these directions by explicitly decoupling supervision from execution, maintaining an external structured task state independent of the agent's internal narrative, and closing the loop through behavioral verification of interventions.

5.2. Formalization

Let the executing agent's policy be $u_t \sim \pi_A(u_t \mid h_t)$, where h_t denotes the agent's internal context (dialogue history, tool returns, self-generated summaries, etc.) and u_t is the agent's action decision. Conventional memory m_t, reflection ρ_t,

and critic q_t are typically functions of h_t:

$$m_t, \rho_t, q_t = f_A(h_t) \tag{4}$$

When h_t already contains an erroneous task narrative, these auxiliary modules may inherit the same bias, forming a closed loop of "erroneous diagnosis → context contamination → auxiliary modules reinforce error → deeper contamination."

Nexus introduces an external state representation independent of h_t. The Observer records the agent's textual output s_t, tool call a_t, environmental return r_t, and external variable changes Δv_t from the sequence of observable behavioral facts $O_{\leq t}$. Critically, s_t is recorded as the behavioral fact that "the agent said X," not as trusted content. The WorldMap aggregates these records into a structured task state M_t:

$$M_t = \text{WorldMap}(O_{\leq t}) \tag{5}$$

This state explicitly maintains the agent's current plan, claim list, tool log, variable state, and change digest. Before each execution turn, the WorldMap constructs a state capsule from M_t and injects it into the agent prompt; this injection occurs every turn, independently of whether the Mentor intervenes.

The Mentor assesses whether a claim–evidence mismatch exists based on M_t and the original task goal g, and decides on intervention d_t:

$$d_t \sim \pi_M(d_t \mid M_t, g) \tag{6}$$

The Mentor's core criterion is not whether the agent textually acknowledges an error, but whether its subsequent plan, tool use, and claims exhibit behavioral change. If behavior remains unchanged, intervention escalates progressively: the first detection injects $\langle \text{reason}, \text{directive}, \text{required_response} \rangle$; a repeated mismatch triggers force_replan; persistent failure leads to request_context_reset (truncating the context to the turn before lock-in, and re-injecting the state capsule and goal).

Under this decomposition, execution and supervision are explicitly isolated: the agent continues executing tasks based on h_t with the state capsule, while the Mentor judges independently from external structured evidence, without inheriting the agent's prior narrative.

5.3. Components and Algorithm

The Observer records behavioral facts: agent output text s_t (as behavioral record, not trusted content), tool calls a_t, environment returns r_t, and external variable diffs Δv_t. The WorldMap maintains a structured task state—`plan`, `claims`, `tool_log`, `var_state`, `change_digest`, and `open_directives`—and injects a state capsule (RenderStateCapsule + RenderChangeDigest + OpenDirectives) into the agent prompt every turn, independently of mismatch detection. The Mentor compares agent claims c_t against the tool log for plan inconsistency, tool misuse, premature termination, and state omissions; upon detection, it issues a directive and records it in `open_directives`. Verification occurs at the start of the next turn: the Mentor checks whether s_t, u_t, p_t, c_t reflect the required behavioral change. If not, intervention escalates through directive injection → force_replan → request_context_reset. Figure 4 summarizes the three-component architecture and the intervention loop. The full algorithm is given in Algorithm 1 (Appendix A.1).

5.4. Preliminary Evaluation

We evaluate Nexus on CLAWMark (Meng et al., 2026), a benchmark of complex daily-life tasks featuring multi-step tool calls, external state changes, and verifiable checkers. These properties make CLAWMark suitable for examining whether an agent maintains correct task state over extended execution. Our evaluation focuses on failure cases and their repair.

Across 103 Kimi K2.5 plain baseline runs, 47% of task scores fall below 0.30 and 27% below 0.15. Low scores concentrate in sparse-evidence, tool-chain-complex domains (e.g., `hr`, `insurance`, `investment_analyst`, `legal_assistant`). This pattern suggests that failure is not merely caused by isolated tool errors, but is often tied to task-state maintenance and erroneous trajectory persistence.

Ablation further characterizes the boundaries of each component. In low-scoring cases—where outcomes are tightly coupled to the agent's initial actions—WorldMap alone, recording and injecting external state, is often insufficient to recover a correct trajectory: the agent, while now possessing more complete environmental information, may still continue operating under the same erroneous narrative. This indicates that external state recording is necessary but not sufficient; detecting when a substantive claim–evidence divergence has occurred, and deciding how to intervene, are both required.

The mode of intervention matters equally. A rule-based intervention (semantics-free, triggered by fixed conditions) degrades performance on 6 of 10 tasks, reducing the average score by roughly 0.10 (journalist_task8 drops from 1.000 to 0.385). In these failures, the rule misclassifies transient tool faults or local interface issues as task deviation and forces an intervention, breaking an otherwise effective trajectory. In contrast, a Mentor with semantic-level mismatch detection can distinguish "the tool is temporarily unavailable" from "the agent is acting under an incorrect premise," thereby avoiding such collateral damage.

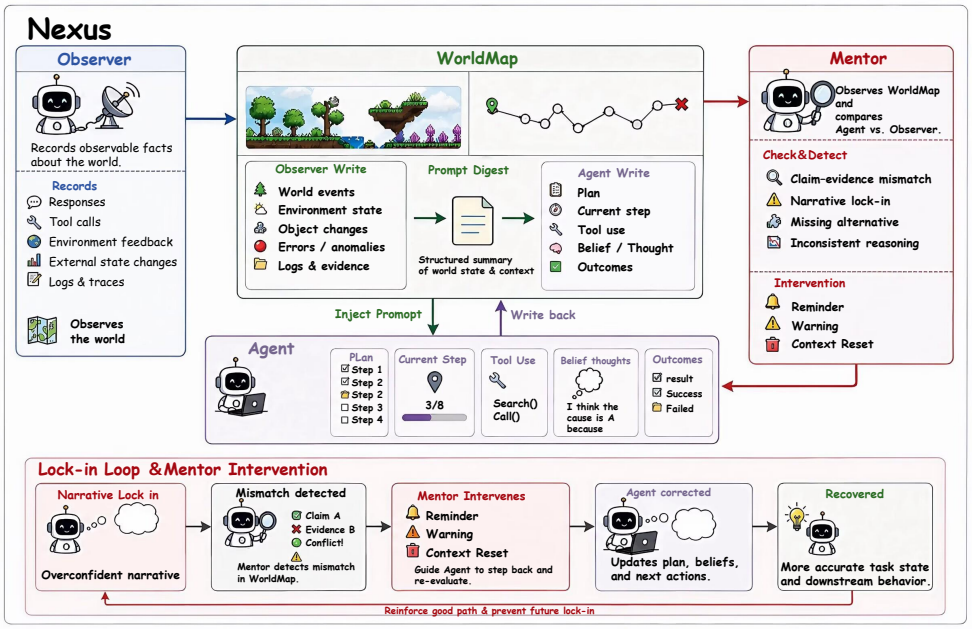

Figure 4. **Nexus framework architecture and supervision workflow.** Left: the three-component architecture—Observer records behavioral facts, WorldMap maintains structured task state and injects a state capsule every turn, Mentor detects claim–evidence mismatches and triggers escalating interventions. Right: the intervention loop—directive injection, behavior-level verification at the next turn, and escalation through force_replan to context_reset when the agent fails to respond.

WorldMap + Change Digest yields substantial gains on journalist_task1 ($0.646 \rightarrow 0.917$) and insurance_task5 ($0.604 \rightarrow 0.924$). Both tasks involve cross-stage silent information changes, suggesting that explicitly recording environmental changes and re-injecting the relevant context can help mitigate the agent's omission of critical state transitions.

pm_task1 illustrates how the Mentor can alter a locked-in execution trajectory in a concrete failure case. Without the Mentor, the agent misinterprets a single failed Notion API call as systemic unavailability of the external system, skips a key deliverable, and terminates early, scoring only 0.111. With the Mentor enabled, it detects tool misuse, a local substitute for the external deliverable, and premature termination before the finalize step, then triggers intervention. The agent subsequently adjusts its trajectory, raising the score to 0.722, with 8 of 11 checkers flipping from FAIL to PASS.

Taken together, these preliminary results suggest that Nexus's effectiveness derives from its closed-loop structure: the WorldMap provides an external state basis every turn; the Mentor performs semantic-level intervention upon detecting mismatch; and the Observer verifies whether the intervention has genuinely altered the agent's behavior through subsequent plan, tool-use, and claim changes. Larger-scale cross-model, cross-task-domain evaluation is left to future work.

6. Conclusion

We introduce Narrative Lock-in, a failure pattern in open-ended long-horizon agent tasks: an early erroneous claim consolidates into task state and persistently shapes downstream evidence interpretation, planning, and action selection. The core problem is not a single-step mistake, but how an erroneous judgment enters task state and is maintained across continued execution.

Agents faced with user corrections, counterexamples, or reflection may fail to substantively update their task state, instead assimilating new evidence into the existing narrative. We design controlled trigger experiments across eight task domains, five challenge protocols, and four models, analyzing lock-in via claim persistence, revision rate, and recovery rate.

To mitigate this pattern, we propose Nexus, a framework that decouples task execution from external state supervision via Observer, WorldMap, and Mentor components, with behavioral verification to determine whether the agent has genuinely departed from the erroneous path. Preliminary results suggest Nexus reduces error-path persistence and improves recovery. Further discussion is provided in Appendix A.

References

Augenstein, I., Baldwin, T., Cha, M., Chakraborty, T., Ciampaglia, G. L., Corney, D., DiResta, R., Ferrara, E., Hale, S., Halevy, A., et al. Factuality challenges in the era of large language models and opportunities for fact-checking. *Nature Machine Intelligence*, 6(8):852–863, 2024.

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., et al. Why do multi-agent llm systems fail? *Advances in Neural Information Processing Systems*, 38, 2026.

Dongre, V., Rossi, R. A., Lai, V. D., Yoon, D. S., Hakkani-Tür, D., and Bui, T. Drift no more? context equilibria in multi-turn llm interactions. *arXiv preprint arXiv:2510.07777*, 2025.

Fei, T., Chen, C., Pan, Y., Zheng, M., and Song, M. Codedelegator: Mitigating context pollution via role separation in code-as-action agents. *arXiv preprint arXiv:2601.14914*, 2026.

Hong, J., Byun, G., Kim, S., Shu, K., and Choi, J. D. Measuring sycophancy of language models in multi-turn dialogues. *arXiv preprint arXiv:2505.23840*, 2025.

Laban, P., Hayashi, H., Zhou, Y., and Neville, J. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*, 2025.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Li, G., Hammoud, H., Itani, H., Khizbullin, D., and Ghanem, B. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Maynez, J., Narayan, S., Bohnet, B., and McDonald, R. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1906–1919. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.173.

McClendon, S. A., Gallego-Feliciano, J., Zervoudakis, S., and Saravanos, A. Three roles, one model: Role orchestration at inference time to close the performance gap between small and large agents. *arXiv preprint arXiv:2604.11465*, 2026.

Meng, F., Du, L., Wu, Z., Chen, G., Liu, X., Liao, J., Jiang, C., Wan, Z., Gu, J., Zhou, P., et al. Clawmark: A living-world benchmark for multi-turn, multi-day, multimodal coworker agents. *arXiv preprint arXiv:2604.23781*, 2026.

Packer, C., Fang, V., Patil, S. G., Lin, K., Wooders, S., and Gonzalez, J. E. Memgpt: towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Perez, E., Ringer, S., Lukosiute, K., Nguyen, K., Chen, E., Heiner, S., Pettit, C., Olsson, C., Kundu, S., Kadavath, S., et al. Discovering language model behaviors with model-written evaluations. In *Findings of the association for computational linguistics: ACL 2023*, pp. 13387–13434, 2023.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Sharma, M., Tong, M., Korbak, T., Duvenaud, D., Askell, A., Bowman, S., Durmus, E., Hatfield-Dodds, Z., Johnston, S., Kravec, S., et al. Towards understanding sycophancy in language models. In *International Conference on Learning Representations*, volume 2024, pp. 110–144, 2024.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Sunil, B. D., Sinha, I., Maheshwari, P., Todmal, S., Mallik, S., and Mishra, S. Memory poisoning attack and defense on memory based llm-agents. *arXiv preprint arXiv:2601.05504*, 2026.

Wang, H., Leong, C. T., Wang, J., and Li, W. Seeing isn't believing: Mitigating belief inertia via active intervention in embodied agents. *arXiv preprint arXiv:2604.17252*, 2026.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Zhang, G., Wang, J., Chen, J., Zhou, W., Wang, K., and Yan, S. Agentracer: Who is inducing failure in the llm agentic systems? *arXiv preprint arXiv:2509.03312*, 2025a.

Zhang, S., Yin, M., Zhang, J., Liu, J., Han, Z., Zhang, J., Li, B., Wang, C., Wang, H., Chen, Y., et al. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. *arXiv preprint arXiv:2505.00212*, 2025b.

Zhu, K., Liu, Z., Li, B., Tian, M., Yang, Y., Zhang, J., Han, P., Xie, Q., Cui, F., Zhang, W., et al. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

Zou, W., Dong, M., Calvo, M. R., Chang, S., Guo, J., Lee, D., Niu, X., Ma, X., Qi, Y., and Jiang, J. Poison once, exploit forever: Environment-injected memory poisoning attacks on web agents. *arXiv preprint arXiv:2604.02623*, 2026.

A. Appendix

A.1. Algorithm: Nexus Supervision Loop

Algorithm 1 Nexus Supervision Loop

Require: Task goal g, execution agent π_A, max turns T
Ensure: Task result, supervision log $\mathcal{L}$

```
Initialize WorldMap M ← ∅, L ← ∅,
    stale_mismatch_count ← 0
for t = 1, ..., T do
    {— Verify pending directives from previous turn —}
    if M.open_directives ≠ ∅ then
        changed ← VerifyBehaviorChange(M, s_t, u_t, p_t, c_t)
        if changed then
            CloseDirectives(M)
            stale_mismatch_count ← 0
        else
            if stale_mismatch_count = 2 then
                ForceReplan(M, g)
            else if stale_mismatch_count ≥ 3 then
                RequestContextReset(M, g)
            end if
        end if
    end if
    {— WorldMap: build state capsule (every turn) —}
    capsule ← RenderStateCapsule(M) ⊕ RenderChangeDigest(M)
    capsule ← capsule ⊕ OpenDirectives(M)
    InjectCapsule(capsule, h_t)
    {— Agent execution —}
    s_t, u_t, p_t, c_t ∼ π_A(· | h_t ⊕ capsule)
    a_t ← ToolCall(u_t)
    r_t ← Env(a_t)
    {— Observer: record behavior facts —}
    o_t ← (s_t, a_t, r_t, Δv_t)
    {— WorldMap: update structured state —}
    M.plan ← p_t
    M.claims ← M.claims ∪ {c_t}
    M.tool_log ← M.tool_log ∪ {o_t}
    M.var_state ← M.var_state ⊕ Δv_t
    {— Mentor: mismatch detection —}
    mismatch ← Check(M, g)
    if mismatch = True then
        stale_mismatch_count ← stale_mismatch_count + 1
        d_t ← (reason, directive, required_response)
        Inject(d_t, h_t)
        M.open_directives ← M.open_directives ∪ {d_t}
    end if
    L ← L ∪ {(o_t, M.snapshot, mismatch, d_t)}
end for
return Final M, task result, L
```

A.2. Related Work

LLM and Agent Failure Modes.

Narrative Lock-in intersects with several lines of work on LLM failures and agent reliability. Early research on hallucination and factuality focused on models producing incorrect facts, fabricated information, or unreliable claims during generation, and proposed mitigation strategies including retrieval augmentation, fact verification, constrained generation, and external tool calls. These efforts provide an important foundation for understanding how erroneous claims arise and how they might be corrected. Sycophancy and instruction-following failure studies further show that model judgments do not always stably depend on external facts and task goals—models may exhibit excessive compliance, adapt to incorrect premises, or produce unstable judgments under leading instructions.

Another relevant line examines long-context and multi-turn reliability. "Lost in the Middle" (Liu et al., 2024) demonstrated that models are sensitive to the position of key information in long contexts and may fail to reliably utilize evidence located in the middle portion. "LLMs Get Lost" (Laban et al., 2025) further showed that early incorrect assumptions in multi-turn interactions significantly reduce subsequent recovery ability, causing models to remain influenced by prior erroneous judgments across turns. These studies indicate that LLM failures often extend beyond single-generation factual errors to encompass information utilization failures in long contexts, multi-turn interactions, and dynamic evidence updating.

Recent work on agent failure modes and failure attribution analyzes error sources in long-horizon agents from a systems perspective. This body of work shows that agent failures are typically not isolated single-step errors, but propagate across planning, tool use, memory, reflection, and execution modules, forming cross-step failure cascades. Related efforts characterize these complex failure paths through error taxonomies, root-cause debugging, and multi-agent failure attribution.

Narrative Lock-in sits at the intersection of these directions. It concerns a trace-level failure mechanism: the agent forms an erroneous claim from real but partial observations, tool feedback, or environmental signals; this claim is subsequently organized into a fluent, self-consistent task narrative and enters the agent's operational task state. Over subsequent long-horizon execution, this narrative persistently shapes evidence interpretation, planning, action selection, and state updating, so that the agent may continue along the erroneous trajectory even when faced with new feedback, counterexamples, or correction signals. This paper characterizes this mechanism as Narrative Lock-in and further analyzes its observable manifestations—evidence assimilation, contradiction resistance, commitment escalation, and information efficiency collapse—in long-horizon agent execution.

Long-Horizon Agent Harnesses and State Supervision.

Existing long-horizon agent harnesses improve agents' sustained execution capability in complex environments through mechanisms such as memory, reflection, skill libraries, tool use, environmental feedback, and structured planning. Voyager demonstrated how agents can achieve long-term capability improvement in open environments through self-exploration, skill accumulation, and continuous feedback. MindForge and related systems further explored the role of memory, planning, reflection, and environmental interaction mechanisms in long-horizon task execution. These efforts show that agent harnesses are not merely task execution interfaces but are increasingly becoming essential infrastructure supporting long-term learning, state maintenance, and behavioral improvement.

Related work has also begun to address agent correction, role decomposition, and history management. "Three Roles, One Model" (McClendon et al., 2026) improved agents' error correction capability in multi-turn execution through role separation and history isolation, demonstrating that structurally decomposing execution, evaluation, and revision processes can enhance agent reliability. Similarly, critic, monitor, reflection, and state-tracking mechanisms have been employed to detect local errors during agent execution and assist agents in adjusting subsequent behavior.

These systems provide important engineering pathways for long-horizon agent execution. However, in many agent harnesses, the maintenance of task state, experience summarization, error interpretation, and subsequent correction still depend heavily on the executing agent itself. When an early claim has already diverged from the true task state, downstream memory, reflection, or self-critique may continue to operate on the same erroneous premise, thereby reinforcing rather than correcting the original trajectory.

Nexus addresses this problem by explicitly decoupling task execution from external state supervision. Specifically, the Observer records behavioral facts from the agent's execution trajectory, the WorldMap maintains an external task state independent of the executing agent, and the Mentor detects claim–evidence mismatches based on structured evidence and

verifies whether interventions produce behavioral change. Unlike harnesses that primarily focus on improving execution capability or local error correction, Nexus treats Narrative Lock-in as a trace-level agent failure mode to be measured and intervened upon, with emphasis on whether the agent substantively updates its plan, tool-use strategy, and task state.

A.3. Discussion: Possible Underlying Mechanisms

We outline four candidate mechanisms that may contribute to Narrative Lock-in. These are offered as explanatory hypotheses rather than established causal accounts; rigorous attribution requires comparing lock-in incidence across different training paradigms, context management strategies, and state supervision mechanisms.

Autoregressive generation amplifies the prior effect of early claims. LLMs generate token by token, with each step conditioned on existing context. Once a model forms and writes down a high-confidence judgment early on, that judgment becomes an implicit constraint on subsequent interpretation, reasoning, and action planning. To maintain semantic coherence and local consistency, the model is more inclined to extend along the existing narrative than to actively reopen the problem space.

Training and post-training may reward coherent narratives over substantive revision. Pre-training corpora contain abundant complete, fluent, causally closed text structures, predisposing models to complete explanations and maintain contextual consistency. Post-training preference optimization often rewards clear, confident, actionable responses, which may reinforce the tendency to continue along an established path under uncertainty rather than explicitly suspending judgment, abandoning the current path, and reconstructing the diagnostic state.

Re-verification and task-state reconstruction carry higher generative cost. Overturning an early judgment requires isolating variables, reconstructing test conditions, distinguishing isomorphic from contaminated evidence, and reorganizing task objectives. By contrast, assimilating new evidence into the existing explanation is lower-cost and more consistent with the generative pressure for contextual continuity. The model may therefore substitute self-consistency for factual verification, treating "can be explained within the current context" as a proxy for "is more likely to be correct."

LLMs are context-bound systems without a persistent epistemic state. Their "beliefs," "reflections," and "role perceptions" are primarily induced by the current context rather than maintained by a stable internal subject. When the context is already occupied by an early erroneous narrative, the model lacks a persistent verification mechanism independent of that context to flag that narrative lock-in may have occurred. Consequently, a local erroneous claim may continuously organize subsequent attention, evidence absorption, and action selection, producing systematic error persistence.

Human-in-the-loop remains the most effective intervention—and the hardest to scale. In practice, human oversight is currently one of the most reliable harness mechanisms for breaking Narrative Lock-in: a human operator can recognize when the agent is "stuck in a story," inject clean counterevidence with explicit scope and source, and demand behavioral-level re-evaluation rather than textual acknowledgment. However, human intervention is costly, latency-prone, and does not scale to autonomous long-horizon deployments. A central motivation behind Nexus—and a key direction for future work—is to equip the Mentor with a meaningful subset of this oversight capability: detecting when the agent's trajectory has become narrative-constrained, issuing targeted interventions with verifiable behavioral criteria, and escalating to human attention only when automated intervention fails. In this sense, the Mentor is not intended to replace human judgment, but to absorb the routine, detectable class of lock-in failures that currently require manual correction, reserving human-in-the-loop for genuinely ambiguous or high-stakes cases.

A.4. Limitation and Future Outlook

Limitations.

1. Nexus does not directly improve model reasoning capability. Nexus does not modify model parameters or involve fine-tuning, reinforcement learning, or architectural changes; it operates as an external supervision and intervention mechanism on the agent's execution process. The Mentor's judgment quality therefore remains bounded by the reasoning and instruction-generation capabilities of the model that implements it.

2. Scope of applicability. Nexus is primarily suited to scenarios where the executing agent possesses the underlying capability to complete the task but its execution trajectory becomes constrained by an erroneous narrative. If the executing agent fundamentally lacks basic tool-use competence, or if the Mentor cannot identify clear claim–evidence mismatches,

Nexus is unlikely to produce effective interventions.

3. Limited experimental scale. This paper focuses on defining, mechanistically analyzing, and preliminarily validating Narrative Lock-in as a failure pattern, rather than conducting large-scale benchmark evaluation. We selected several long-horizon tasks that exhibit clear Narrative Lock-in, compared trace-level behavioral differences with and without Mentor intervention, and analyzed how erroneous claims enter, are maintained within, and are broken from the task state. The current experiments are not yet sufficient to establish the generality of this phenomenon across all models, tasks, and real-world settings.

4. Underlying mechanisms require further validation. We have proposed candidate explanations from the perspectives of autoregressive path dependency, overgeneralization of local failure signals, training and post-training preferences for coherent narratives, and amplification of early errors by state maintenance mechanisms. These remain mechanistic hypotheses that have not been directly verified at the level of model internal representations, parameter space, or training dynamics.

Future Outlook.

Future work will proceed along three directions. First, we plan to scale up the experimental evaluation to verify the generality of Narrative Lock-in across a wider range of models, task domains, and deployment contexts. Second, we aim to further analyze the underlying mechanisms of lock-in formation and maintenance, potentially through controlled comparisons across different training paradigms, context management strategies, and internal representation analysis. Third, we intend to improve the Mentor's judgment precision, reduce system overhead, and develop robust fallback mechanisms for cases where interventions are misapplied. We hope this work provides a new analytical lens for long-horizon agent reliability research, encouraging attention not only to final task scores but also to how erroneous claims enter the task state, how they are persistently maintained, and how they can be effectively broken.