Rejected but Unaudited: How LLMs Use the Eliminative Evidence Agentic Systems Discard

Anonymous ACL submission

Abstract

A growing class of agentic systems preserves the reasoning that pipelines normally discard, re-surfacing the hypotheses an agent ruled out on the untested premise that a model will use this eliminative evidence once it is returned. We test that premise behaviorally, treating fault localization, where a decision is checkable against ground truth, as a laboratory for a question prior work has examined only in toy, non-verifiable settings. A decoupled hypothesis-ledger localizer supplies reasoned rejections as a controlled variable, and an editor commits to a fault location under four conditions: positive context alone (POS), plus the genuine rejections (NEG-TRUE), plus a forged rejection of the correct location (NEG-FALSE), and plus a length-comparable control (NEG-SHUF). Across 108 paired instances the model's use of eliminative evidence is correctness-sensitive but asymmetric. Genuine evidence never lowers the chance of a correct commitment and slightly raises it, though the content effect over the control only approaches significance (p=0.0625) and never reverses. The load-bearing finding is the other side: a single forged rejection of the correct location reliably misleads the editor, a harm that is statistically significant and replicates across two models (pooled p=0.0018), while genuine rejections never harm. Since preserved reasoning is rarely guaranteed current, a stale rejection is exactly this forged condition, and the paradigm inherits a failure mode that helps only modestly when the rejection is right and misleads reliably when it is stale. What such systems most need to preserve alongside a rejection is the means to tell whether it remains true.

1 Introduction

Large language models are increasingly deployed not as standalone predictors but as decision-making components embedded within larger autonomous systems. In such systems, agentic pipelines that plan, retrieve, reason, and act over extended horizons, each action is a decision conditioned on context accumulated from earlier stages. A defining property of these pipelines is that they accumulate far more intermediate evidence than they retain: as context windows fill, agents summarize and compact prior context, and recent work argues that this process also discards the reasoning that produced the current state (Kang et al., 2025; Jha et al., 2026; Penaroza, 2026). Recent work has shown the resulting loss of fact-level content to be both substantial and architectural rather than model-specific, persisting across multiple frontier models (Zahn and Chana, 2026). A particularly consequential asymmetry in what survives is that an agent's conclusions are preserved while its *eliminative* reasoning, the hypotheses it considered and ruled out, and why, is among the first things discarded (Jha et al., 2026; Penaroza, 2026).

This observation has motivated a growing body of work that builds machinery to *preserve* discarded reasoning, persisting rejected hypotheses, recording per-evidence deliberation, and re-injecting it across compaction boundaries (Jha et al., 2026; Penaroza, 2026). These systems rest on a shared and largely untested premise: that preserved eliminative evidence, once surfaced back to the model, is information the model will *use*. Whether an LLM, handed an explicit record of what was ruled out and why, actually conditions its decision on that record, or treats it as inert text, is a behavioral question about how these models integrate negative evidence. It is precisely the question on which the value of the entire preservation paradigm depends.

Behavioral studies of how LLMs reason from negative or eliminative information suggest grounds for doubt. Models are known to handle negation poorly (Truong et al., 2023; García-Ferrero et al., 2023), to underperform when reasoning toward incorrect rather than correct options on two-choice commonsense and scientific ques-

tions (Balepur et al., 2024), and to exhibit anchoring patterns analogous to human judgment (Lou and Sun, 2024), with related work documenting confirmation-bias effects under knowledge conflict (Echterhoff et al., 2024). Yet this evidence comes almost entirely from toy, low-stakes paradigms, multiple-choice vignettes, synthetic rule-discovery, hand-curated decision scenarios, where outcomes are not objectively verifiable and the "evidence" carries no causal structure. It remains unknown whether these behavioral tendencies persist in a high-stakes, verifiable setting where decisions have checkable consequences and the eliminative evidence is causally rich.

We close this gap by using a verifiable domain, agentic program repair, where a fault must be localized to a specific function and correctness is checkable against ground truth, as a behavioral laboratory. Our instrument is a localizer that, for each bug, produces a structured ledger of competing hypotheses, most of which it explicitly rules out with causal reasons. Crucially, this ledger is *decoupled*: rejected hypotheses are recorded but do not control the final ranking, so rejection trace becomes a clean experimental variable that we can pass, withhold, or corrupt while holding everything else fixed. We make no claim that this localizer is optimal; it is the apparatus that yields reasoned, decoupled, causally-grounded eliminative evidence, the three properties our probe requires.

On this basis we ask, in a controlled four-condition design, whether an editor model's commitment to a fault location is changed by the eliminative evidence an agentic system would normally discard. We compare positive context alone (POS), positive context augmented with the genuine rejected hypotheses and their reasons (NEG-TRUE), the same augmented with a *forged* rejection of the correct location (NEG-FALSE), and a length-comparable control in which the rejection reasons are replaced by reasons drawn from a different instance (NEG-SHUF). This design isolates not merely whether the model responds to negative evidence, but whether its response tracks the *correctness* of that evidence, the property that determines whether preserving rejections is safe or hazardous at scale.

We find a consistent, asymmetric pattern. Genuine eliminative evidence never harms and slightly aids localization, but its net value over positive context is small and does not reach significance. A forged rejection of the correct location, in contrast, reliably misleads the editor: the harm is statistically significant and replicates across two models (pooled p=0.0018). The model's use of negative evidence is correctness-sensitive, but the upside and downside are not symmetric in strength.

Our contributions are: (i) we reframe a question the agent-systems community treats as an engineering problem, preserving discarded reasoning, as a testable behavioral question about how LLMs integrate eliminative evidence, and we test it in a verifiable rather than toy setting; (ii) we introduce a decoupled hypothesis-ledger localizer as a clean instrument for producing reasoned negative evidence, together with a four-condition probe (including a length-comparable control and a forged-rejection credulity test) that isolates correctness-sensitivity; and (iii) across 108 verifiable instances we characterize the value of the eliminative evidence agentic systems discard, finding it bounded and correctness-contingent, modestly helpful and only when accurate, while a stale or forged rejection reliably and significantly misleads, which bounds what the discarded-reasoning-preservation paradigm can deliver.

2 Related Work

2.1 Discarded Reasoning in Agentic Systems

Our motivation begins with a property of agentic pipelines documented largely in the systems literature: agents accumulate far more intermediate reasoning than they retain. As context windows fill, agents summarize and compact their state, and a recurring observation is that the reasoning behind an agent's actions, including which evidence items were used or rejected and why, is discarded before its significance can be established (Kang et al., 2025; Jha et al., 2026). Penaroza (2026) states the loss directly for retrieval-and-reason agents: at each step the chain of thought, including the agent's evaluation of why each piece of evidence was used or rejected, is discarded, and the next query starts from scratch. Zahn and Chana (2026), studying factual recall under summarization-based compaction, report that this compaction loss is architectural rather than model specific, persisting across multiple frontier models.

A growing body of work responds by building machinery to preserve discarded reasoning across runs. Penaroza (2026) propose persisting per-evidence chain of thought as graph edges so that prior evaluations of an item, including rejec-

tions, can be surfaced in later runs, with the stated goal of reducing variance and driving agent verdicts toward deterministic accuracy across repeated queries. Jha et al. (2026) take a complementary route, preserving graph-structured belief states over a dependency graph and enabling explicit belief revision as new evidence arrives, rather than letting earlier judgments be overwritten by summarization. Related efforts maintain episodic reflection buffers (Shinn et al., 2023) or distil reusable strategies from both successful and failed trajectories (Ouyang et al., 2026). These systems share a premise that our work makes explicit and tests: that preserved eliminative reasoning, once surfaced back to a model, is information the model will use. They argue for the engineering value of preservation across runs; they do not test whether a model behaviorally conditions a single decision on the rejected hypotheses it is shown. That behavioral question is the gap we address, and it is logically prior to the architectures built on top of it.

2.2 How Models Reason from Negative and Eliminative Evidence

A second line of work studies, behaviorally, how language models handle negative information. Models are known to process negation poorly: they underestimate its effect on meaning and struggle with world knowledge stated in negative form (Truong et al., 2023; García-Ferrero et al., 2023). Most directly related to our question, Balepur et al. (2024) introduce process of elimination with chain of thought, asking models to reason toward incorrect rather than correct answers, and find that this strategy consistently underperforms reasoning toward the correct answer across commonsense and scientific benchmarks. This establishes an eliminative reasoning deficit, but in a specific regime: two-choice multiple choice questions where the negative options are constructed distractors lacking causal structure relative to the task. Our setting differs on exactly these axes. The eliminative evidence we study carries causal structure (a function is ruled out because it is downstream of the fault, or is overridden, or only reads a value without using it), the candidates are real program elements rather than constructed distractors, and the correct answer is checkable against ground truth. We therefore test whether the eliminative deficit observed in toy multiple choice settings persists when the negative evidence is causally rich and the task is verifiable.

2.3 Behavioral Evaluation of LLMs as Decision Makers

Our framing treats the model as a decision making component whose commitment is conditioned on the context it is given, connecting to work on cognitive biases in LLMs. Controlled studies report anchoring, confirmation, and framing effects analogous to human judgment (Lou and Sun, 2024; Echterhoff et al., 2024), and Xie et al. (2024) show that when supporting and contradicting external evidence are presented together, models exhibit a confirmation bias and cling to their prior belief, while being receptive to coherent counter evidence when it stands alone. This literature is overwhelmingly conducted in non verifiable, low stakes paradigms: hand curated decision vignettes, synthetic rule discovery, and multiple choice scenarios where the ground truth is stipulated rather than checkable. Recent large-scale evaluation confirms that such behavioral observation remains predominantly conducted on constructed psychologist-authored scenarios rather than verifiable tasks (Knipper et al., 2025). We contribute to this line by moving one such behavioral question, the use of eliminative evidence, into a high stakes verifiable domain, where each decision has a checkable consequence and the evidence is drawn from the system's own deliberation rather than authored by the experimenter.

2.4 Localization and Repair as the Instrument

We use program repair only as a verifiable instrument, not as a contribution to repair systems. Localize then edit is the dominant paradigm for LLM program repair, where a localizer identifies suspect locations that an editor then modifies (Xia et al., 2024; Chen et al., 2025). Function level localization is a recognized bottleneck, with reported correct function match well below full coverage even among strong systems (Yu et al., 2025), and expanding localization context can degrade rather than aid performance at finer granularities (Sepidband et al., 2026a), which is why we include a token matched control. Prior repair localizers transmit only their chosen locations, and at most a *positive* justification for them, to the editor (Xia et al., 2024); we instead isolate the localizer's *rejected* hypotheses as the experimental variable.

A second wave of localizers has broadened the evidence available at the localizer to editor boundary, and the apparatus we use draws on it. Reddy et al. (2026) couple a SweLoc-

trained bi-encoder with a listwise LLM reranker; Sepidband et al. (2026b) attach bug-specific causal explanations to candidate functions; Liu et al. (2025) construct a causal issue graph in which vertices represent discovered sub-issues and edges encode their causal dependencies, performing iterative abductive reasoning; Wang et al. (2026) augments code-localization agents with non-parametric memory mined from a repository's commit history and linked issues; Yang et al. (2025) links issues, pull requests, and code entities through a repository-aware knowledge graph; and Seddik and Fard (2026) extends the program graph to statement-level nodes connected by intra-procedural definition-use edges. Across this line, every artifact still transmitted to the editor is *positive*: a chosen candidate, a justification for it, an attached context. The hypotheses these systems ruled out, and why, are not represented in their outputs.

We make no claim that our localizer is optimal. It is an apparatus chosen because it yields eliminative evidence with three properties our probe requires: it is reasoned (each rejection carries a causal justification), it is decoupled (rejections are recorded without steering the final ranking, so they can be transmitted, withheld, or corrupted independently), and it is structurally grounded (traversal over code dependencies produces the causal rejection reasons our analysis stratifies on).

3 Methodology

3.1 The Premise as a Testable Claim

A model handed the hypotheses an agent ruled out might condition its decision on what a rejection says is false, which is the behavior the preservation premise needs, or it might merely react to the presence of extra negative text. Only the first justifies preserving rejections. Our experiment separates the two: it asks not whether eliminative evidence moves the editor's decision, but whether that movement tracks whether the evidence is *correct*.

3.2 Design: One Variable, Four Levels

We hold everything about the decision fixed, the bug, the candidate functions, the positive localization context, and vary only the eliminative-evidence block. Every instance is run under all four conditions, so each is its own control, every contrast is paired, and any within-instance change in the committed location is attributable to the manipulation. The four conditions are chosen so that each one removes a competing explanation for an observed effect:

- **POS** (control): positive context only, the leading hypothesis and its causal chain. The floor.
- **NEG-TRUE** (treatment): POS plus the genuine rejected hypotheses and reasons, exactly as recorded.
- **NEG-FALSE** (correctness probe): POS plus a single forged rejection of the *gold* function, with a plausible reason (that the true cause lies upstream). If the model is merely responsive to negative text, NEG-FALSE should look like help; if it tracks correctness, it should hurt. This is the arm that separates the two readings above.
- **NEG-SHUF** (confound control): POS plus rejections from a different instance, matched to NEG-TRUE for approximate length (exact token equality is not enforced). Longer context can degrade performance on its own (Sepidband et al., 2026a), so NEG-SHUF controls for added length and format, leaving content as the main difference from NEG-TRUE.

Read together: NEG-TRUE vs POS asks whether genuine evidence moves the decision; NEG-TRUE vs NEG-SHUF isolates content from length; NEG-FALSE vs POS asks whether the model audits what it is told. We pre-register two strata by localizer certainty (confident vs uncertain); the uncertain stratum, where positive localization is weakest, is where eliminative evidence has the most room to act (definitions and counts in Section 4.1).

3.3 The Instrument and Its Required Properties

The design needs eliminative evidence with three properties, and we choose the localizer only to obtain them. It must be *reasoned* (each rejection carries a causal justification, not a score), *decoupled* (rejections are recorded but do not steer the final ranking, so we can transmit, withhold, or forge them without changing anything else), and *structurally grounded* (the reasons come from code structure ruled out as downstream of the fault, overridden and off the execution path, or reading a value without using it).

We obtain this from a multi-stage abductive localizer that we use purely as an instrument. It runs five stages per issue:

1. **Pre-filter.** A fusion of BM25, dense bi-encoder retrieval, and an LLM ranker narrows the repository to $\sim$20 candidate files.

2. **Graph.** A structural graph over functions, files, and call/dependency edges is built from those files.

3. **Seeding.** An LLM-extracted symptom set (exception types, error messages, behaviors, named APIs) marks an initial set of suspect nodes.

4. **Traversal.** A priority-guided walk processes a heap of read/expand actions, scoring each candidate by semantic similarity, causal-chain quality, seed proximity, recency, and depth. Every k iterations it pauses and calls an *abductive reasoner*, a single LLM call that returns, per candidate, a causal chain to the symptom and a structured *ledger entry*: a verdict in {LEADING, CONSIDERING, REJECTED} with a one-sentence causal reason (e.g. "only reads the value, does not modify it" or "downstream of the failure; cause is upstream"). The ledger persists across iterations, so each candidate ends with one best-justified verdict. Traversal stops on a ledger gate ($\geq N$ candidates assessed, one marked LEADING), an iteration cap, or a reasoner-call budget.

5. **Ranking.** A separate consistency voter reruns the listwise reranker over the Stage 4 pool at temperature zero and aggregates by mean score weighted by top-k frequency across runs.

The decoupling is what matters for us. The Stage 5 voter consumes only the candidate set and their causal-chain scores; it does *not* read the ledger, so a rejection's presence or absence never changes the ranking by construction. The ledger is a parallel artifact that records reasoned eliminative judgments without steering the verdict, exactly the property our four-arm probe requires, since we can pass, withhold, or forge the rejections with no other part of the output moving. This is the central reason for using a ledger-based localizer over a scoring one.

From each instance's ledger we take the issue text, the leading hypothesis and its causal chain, the genuine rejections with their reasons, and the candidate file's signature skeleton (function and class signatures in source order, gold function unflagged among them); the gold location comes from the reference patch. We make no claim about the localizer's standalone localization quality: it is held fixed across all instances and all four conditions, and its only role is to emit reasoned, decoupled, structurally grounded rejection traces.

3.4 The Editor Decision

The editor sees a minimal, fixed prompt: the issue, the candidate functions (the file skeleton, signatures only, gold function present and unflagged among the rest in source order), and the condition's evidence block. The framing presents editing the localized function and inspecting another as equally normal, and signals nothing about which candidate is correct. The model must commit on one line, `FAULT: <function_name>`. Framing and ordering are constant across all four arms and the template is frozen across batches, so the evidence block is the only thing that varies.

3.5 Measurement and Inference

The outcome is HIT: whether the committed function matches the gold location. We match on normalized keys, stripping class qualifiers and module paths and accepting either `Class.method` or the bare method name, with leading underscores preserved so dunders such as `__call__` match; an instance with several gold functions counts as a hit on any. For NEG-FALSE we also record MOVED-OFF-GOLD: whether the forged rejection pushed the editor off the function it chose under POS, the direct signature of being misled.

Because the decision is our dependent variable, a tolerant parser accepts the `FAULT:` line, a bare function name, or failing those a unique candidate name in the response. A response that hits the output cap without committing is a genuine non-answer, not a miss, and is dropped; an instance enters a contrast only if all four arms produced a valid commit, a succeed-or-fail-together rule that keeps condition-correlated parsing failures from biasing the contrasts.

All contrasts are paired and use the exact McNemar test on the discordant pairs; we report discordant counts both ways, the hit-rate difference with a 95% interval, and the exact two-sided p, within each stratum and pooled where stated. We also report per-arm prompt-token counts, so that any

NEG-SHUF/NEG-TRUE length difference can be accounted for when interpreting the content contrast.

4 Experimental Setup

4.1 Instances

We draw instances from SWE-bench Verified (Jimenez et al., 2024), where each instance pairs a real GitHub issue with a reference patch and an executable test. We use it not as a repair benchmark but as a source of verifiable fault locations: the reference patch defines the gold function(s) to be edited. This is inference-only, so there is no train/development split.

We retain instances whose localizer ledger is well-formed (confidence labels present) and on which both editors returned a valid four-arm commitment (Section 3.5), giving a shared cohort of 108 instances evaluated identically under both models. These split into two pre-registered strata by localizer certainty. In the 79 confident instances the localizer committed to a single leading hypothesis, which serves as the positive anchor. In the 29 uncertain instances it committed to zero or to multiple leading hypotheses; here the anchor is set by a fixed pre-registered rule to the final ranked top-1 candidate. The uncertain stratum is where positive localization is weakest, and therefore where eliminative evidence has the most room to change the decision. Instances span ten repositories and are dominated by *django* (58.3%), a consequence of random sampling from a django-heavy ledger pool rather than a design choice, so we treat cross-repository generality as provisional. SWE-bench Verified is distributed for research use under the MIT license, and all model inference complied with the provider API terms.

4.2 Models

Localization ledgers were generated with Gemini 2.5 Pro (Gemini Team, Google, 2025), which is also the primary editor; we add Llama 3.3 70B Instruct (Dubey et al., 2024) as a second editor to test whether the effect depends on the model producing visible deliberation. Gemini 2.5 Pro is a closed model whose parameter count is undisclosed; Llama 3.3 has 70B parameters. The localizer and editor stages are independent: the editor never sees the localizer's ranking, only the condition-specific evidence block, so the editor is the manipulated component and the localizer is held fixed across all conditions and instances.

4.3 Inference Configuration

All inference ran through hosted model APIs (editor queries via OpenRouter), with no local GPU compute. The editor ran at temperature 0 with a 16,000-token output budget, sized to avoid truncating reasoning before commitment; capped non-committing responses are excluded under the succeed-or-fail-together rule (Section 3.5). We perform no hyperparameter search, and at temperature 0 each instance is a single deterministic decode, so reported variation is across instances, not re-samples. Total editor cost was approximately $60, the full pre-allocated budget; cohort size was fixed by exhausting it, not by inspecting outcomes.

5 Results and Analysis

We evaluate two editors, Gemini 2.5 Pro and Llama 3.3 70B Instruct, on the same 108 instances—those on which both editors returned a valid four-arm commitment (Section 3.5), giving the cleanest possible paired cross-model comparison. The cohort size was fixed in advance by a pre-allocated inference budget, not extended after inspecting outcomes. All contrasts use the exact McNemar test; we report discordant counts in both directions (b: first arm hits where second misses; c: the reverse), the paired hit-rate difference with a 95% Newcombe interval, and the exact two-sided p. Marginal hit rates carry Clopper–Pearson intervals.

5.1 Primary Model: a Correctness-Sensitive Asymmetry

On Gemini the arm ordering matches the hypothesis and never reverses (Table 1): NEG-TRUE (37.0%) > POS (34.3%) > NEG-SHUF (32.4%) > NEG-FALSE (27.8%).

The significant effect is the credulity harm. A single forged rejection of the gold function lowers the hit rate by 6.5 points (NEG-FALSE vs POS, b=0, c=7, p=0.0156, 95% CI $[-11.3, -1.8]$); the interval excludes zero and all seven discordant pairs run in the harmful direction. On each, the editor was directly observed committing to a different function than under POS (MOVED-OFF-GOLD): the forgery does not add noise, it relocates the decision.

The benefit of genuine evidence is one-sided but marginal. NEG-TRUE never loses a discordant pair, to POS (b=3, c=0) or to the length-comparable control NEG-SHUF (b=5, c=0), so across 108 instances no genuine rejection ever moved the editor

Arm	Hit % [95% CI]	Tok.
NEG-TRUE	37.0 [27.9, 46.9]	1884
POS	34.3 [25.4, 44.0]	1448
NEG-SHUF	32.4 [23.7, 42.1]	1788
NEG-FALSE	27.8 [19.6, 37.2]	1468
Contrast	**Δ [95% CI]**	(b, c, p)
NT vs NS	++4.6 [+0.6,+8.7]	$(5, 0, .063)$
NT vs POS	++2.8 [-0.4,+6.0]	$(3, 0, .25)$
POS vs NS	++1.9 [-1.9,+5.6]	$(3, 1, .63)$
NF vs POS	−-6.5 [-11.3,-1.8]	$(0, 7, .016)$

Table 1: Gemini cohort (n=108). Hit and Δ in %, with 95% intervals (Clopper–Pearson for arms, Newcombe for differences); paired contrasts use the exact McNemar test. Arms: NT = NEG-TRUE, NS = NEG-SHUF, NF = NEG-FALSE. Every discordant pair favors NEG-TRUE (c=0) and opposes NEG-FALSE (b=0); the credulity contrast (highlighted) is significant and no contrast reverses.

off a correct commitment. The effect is nonetheless small: neither contrast clears conventional significance (p=0.25 and p=0.0625). The content contrast against NEG-SHUF is the stronger of the two, and its Newcombe interval $[+0.6, +8.7]$ sits just above zero while the exact McNemar p stays marginally above .05; we read it as suggestive, not conclusive. NEG-SHUF is on average shorter than NEG-TRUE (1788 vs 1884 tokens); since longer context tends to degrade rather than aid performance on these tasks (Sepidband et al., 2026a), this difference works against NEG-TRUE, so the content advantage is conservative rather than a length artifact.

5.2 Where the Effect Lives

The pattern is stable across the two pre-registered localizer-certainty strata. It holds on the confident stratum (n=79: NEG-TRUE 38.0 / POS 35.4 / NEG-SHUF 32.9 / NEG-FALSE 27.8) and on the uncertain stratum (n=29: 34.5 / 31.0 / 31.0 / 27.6), with the same ordering and the same NEG-FALSE floor in both. Two instances make the asymmetry concrete. On `django-11551`, POS, NEG-SHUF, and NEG-FALSE all commit to the wrong location and NEG-TRUE alone recovers the gold function: genuine evidence redirecting the editor where nothing else does. On `django-14608`, POS correctly identifies `full_clean` and a single forged rejection of it under NEG-FALSE pushes the editor onto `non_form_errors`: the harm in miniature.

5.3 Second Model and Pooled Credulity

Llama answers our neutral commit prompt tersely, with little intermediate reasoning, and in this regime the three reasoning-neutral arms sit within a single point (POS 33.3%, NEG-TRUE and NEG-SHUF 32.4%); no pairwise contrast among them exceeds one discordant instance in either direction (p=1.0 throughout). Genuine and shuffled eliminative evidence leave the decision unchanged.

The one manipulation that moves Llama is the forged rejection, and it moves it harmfully: NEG-FALSE vs POS gives b=1, c=6 (Δ=−4.6, 95% CI $[-9.5, +0.1]$, p=0.125), a directional replication of the Gemini harm that does not reach significance on Llama alone. The two models dissociate cleanly. The benefit of genuine evidence is contingent on the model deliberating, present if modest on Gemini, absent on the non-deliberating Llama, where genuine content yields the most Gemini-only-correct instances and no movement of its own, while the harm of a forged rejection appears in both and reverses into a gain only once (`django-15525`).

Pooling the credulity contrast over 216 paired observations (the same 108 instances under both models) sharpens it (Table 2): b=1, c=13 (p=0.0018), a −5.6-point paired difference (95% CI $[-8.9, -2.2]$). Of the fourteen observations where the forgery changes the answer, thirteen (92.9%, 95% CI $[66.1\%, 99.8\%]$) run in the harmful direction, the interval excluding the 50% null. Three instances, `django-11433`, `django-14315`, and `django-16100`, are misled identically under both models: a model-independent credulity signature, and the single strongest piece of evidence that the asymmetry is a property of the evidence, not the editor.

6 Discussion

We set out to test a premise a growing class of agentic systems takes on faith: that preserved eliminative evidence, re-surfaced to a model, is information the model will use (Jha et al., 2026; Penaroza, 2026). Tested behaviorally in a verifiable setting, it holds only in a narrow, asymmetric form. A confident false rejection misleading a model is unsurprising on its own; yet models also cling to their priors under knowledge conflict (Xie et al., 2024), so the direction was not settled in advance. The contribution is that this harm is reliable and model-independent while the symmetric benefit is

Model	Hit rate (%)			
	N-TRUE	POS	N-SHUF	N-FALSE
Gemini (n=108)	37.0	34.3	32.4	27.8
Llama (n=108)	32.4	33.3	32.4	28.7

N-FALSE vs POS (credulity), exact McNemar	
Gemini	b=0, c=7, p=.016
Llama	b=1, c=6, p=.125
Pooled	b=1, c=13, p=.0018 (216 obs.)

Table 2: Cross-model comparison on the shared 108 instances. On Llama the genuine-evidence arms are statistically tied with POS; only forged evidence moves the answer, and only harmfully. The harm replicates in both models and is highly significant when pooled.

neither, and that it is exactly the regime a preservation system enters when a stored rejection goes stale.

Use of negative evidence tracks its correctness, not its presence. A genuine rejection never moved the editor off a correct commitment (c=0 throughout), while a forged rejection moved it off on 13 of 14 discordant pairs across both models. The decision is conditioned on *what the rejection says is false*, and is only as reliable as the rejection: the same mechanism that makes true rejections safe makes false ones dangerous.

The downside is load-bearing, because the upside is contingent. The benefit of genuine evidence is small, never significant, and present only when the model deliberates; it vanishes on Llama, whose reasoning-neutral arms are indistinguishable from POS. The harm carries none of these conditions: it is significant on Gemini, replicates on Llama, survives pooling (p=0.0018), and appears even on the model that ignores genuine evidence. A preservation system cannot count on the upside but inherits the downside regardless.

A stale rejection is the NEG-FALSE condition. A rejection sound when recorded can be falsified by later state, so a system re-injecting it presents a confidently stated rejection that is now wrong, which is operationally our NEG-FALSE signal. The paradigm does not merely fail to deliver a large benefit; at every compaction boundary where a preserved rejection has gone stale, it manufactures exactly the input that reliably misleads, and the model, in this regime, does not audit it.

Preserve provenance, not just the verdict. The fix is not to discard rejections but to preserve the means to re-check them. A bare "X was ruled out" is the form that misleads; "X was ruled out under state S for reason R" is auditable. What these systems most need to preserve alongside a rejection is the means to tell whether it is still true.

Method. The setup is reusable: a verifiable domain can serve as a behavioral laboratory for questions about negative-evidence integration studied so far mostly in non-verifiable settings (Balepur et al., 2024; Xie et al., 2024). We frame our findings as behavioral observations about a decision, not claims about model internals.

7 Limitations

Our cohort is modest (n=108; 79 confident, 29 uncertain), and the benefit, though perfectly one-sided (c=0 throughout), does not reach significance (best p=0.0625); we therefore rest the benefit claim on its consistency and the correctness asymmetry, not on magnitude. The credulity harm is significant and replicates across both editors, but the benefit is established on the deliberating model alone. The editor sees function signatures, not bodies, so our credulity finding is behavioral and scoped to this evidence regime, not a claim about internals. We note, however, that under POS the editor frequently reaches the gold location from this same signature-level information, so a forged rejection overturns commitments the model was otherwise able to make correctly; the harm is not mere deference under total ignorance.

8 Conclusion

We asked whether LLMs use the eliminative evidence larger systems discard, and whether that use tracks its correctness. Used as a behavioral laboratory, a verifiable repair setting shows genuine evidence to be consistently but modestly helpful and never harmful, while a forged rejection reliably and significantly misleads, across both a deliberating and a non-deliberating model. The finding is not a large benefit but a clean asymmetry: the model conditions on negative evidence without auditing it in the setting we test, so the downside is reliable while the upside is not. This bounds the discarded-reasoning-preservation premise and suggests that what such systems most need to preserve alongside a rejection is the means to tell whether it is still true.

References

Nishant Balepur, Shramay Palta, and Rachel Rudinger. 2024. It's not easy being wrong: Large language models struggle with process of elimination reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10143–10166, Bangkok, Thailand. Association for Computational Linguistics.

Zhaoling Chen, Robert Tang, Gangda Deng, Fang Wu, Jialong Wu, Zhiwei Jiang, Viktor Prasanna, Arman Cohan, and Xingyao Wang. 2025. LocAgent: Graph-guided LLM agents for code localization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8697–8727, Vienna, Austria. Association for Computational Linguistics.

Abhimanyu Dubey and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jessica Maria Echterhoff, Yao Liu, Abeer Alessa, Julian McAuley, and Zexue He. 2024. Cognitive bias in decision-making with LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12640–12653, Miami, Florida, USA. Association for Computational Linguistics.

Iker García-Ferrero, Begoña Altuna, Javier Alvez, Itziar Gonzalez-Dios, and German Rigau. 2023. This is not a dataset: A large negation benchmark to challenge large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8596–8615, Singapore. Association for Computational Linguistics.

Gemini Team, Google. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Saurabh Jha, Rohan Arora, Bhavya, Noah Zheutlin, Paulina Toro Isaza, Laura Shwartz, Yu Deng, Daby Sow, Ruchi Mahindru, and Ruchir Puri. 2026. Think locally, explain globally: Graph-guided LLM investigations via local reasoning and belief propagation. *arXiv preprint arXiv:2601.17915*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. 2025. Acon: Optimizing context compression for long-horizon llm agents. *Preprint*, arXiv:2510.00615.

R. Alexander Knipper, Charles S. Knipper, Kaiqi Zhang, Valerie Sims, Clint Bowers, and Santu Karmaker. 2025. The bias is in the details: An assessment of cognitive bias in llms. *Preprint*, arXiv:2509.22856.

Wei Liu, Chao Peng, Pengfei Gao, Aofan Liu, Wei Zhang, Haiyan Zhao, and Zhi Jin. 2025. GraphLocator: Graph-guided causal reasoning for issue localization. *Preprint*, arXiv:2512.22469.

Jiaxu Lou and Yifan Sun. 2024. Anchoring bias in large language models: An experimental study. *Preprint*, arXiv:2412.06593.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. 2026. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*.

Matthew Penaroza. 2026. Reasoning graphs: Deterministic agent accuracy through evidence-centric chain-of-thought feedback. *arXiv preprint arXiv:2604.07595v1*.

Revanth Gangi Reddy, Tarun Suresh, JaeHyeok Doo, Ye Liu, Xuan Phi Nguyen, Yingbo Zhou, Semih Yavuz, Caiming Xiong, Heng Ji, and Shafiq Joty. 2026. Swerank: Software issue localization with code ranking. *Preprint*, arXiv:2505.07849.

Shahd Seddik and Fatemeh Fard. 2026. ARISE: A repository-level graph representation and toolset for agentic fault localization and program repair. *Preprint*, arXiv:2605.03117.

Melika Sepidband, Hung Viet Pham, and Hadi Hemmati. 2026a. On the role of fault localization context for llm-based program repair. *Preprint*, arXiv:2604.05481.

Melika Sepidband, Hamed Taherkhani, Hung Viet Pham, and Hadi Hemmati. 2026b. RGFL: Reasoning guided fault localization for automated program repair using large language models. *Preprint*, arXiv:2601.18044.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Thinh Hung Truong, Timothy Baldwin, Karin Verspoor, and Trevor Cohn. 2023. Language models are not naysayers: An analysis of language models on negation benchmarks. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 101–114, Toronto, Canada. Association for Computational Linguistics.

Boshi Wang, Weijian Xu, Yunsheng Li, Mei Gao, Yujia Xie, Huan Sun, and Dongdong Chen. 2026. Improving code localization with repository memory. *Preprint*, arXiv:2510.01003.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. 2024. Agentless: Demystifying llm-based software engineering agents. *Preprint*, arXiv:2407.01489.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*.

Boyang Yang, Jiadong Ren, Shunfu Jin, Yang Liu, Feng Liu, Bach Le, and Haoye Tian. 2025. Enhancing repository-level software repair via repository-aware knowledge graphs. *Preprint*, arXiv:2503.21710.

Zhongming Yu, Hejia Zhang, Yujie Zhao, Hanxian Huang, Matrix Yao, Ke Ding, and Jishen Zhao. 2025. Orcaloca: An LLM agent framework for software issue localization. In *Forty-second International Conference on Machine Learning*.

Oliver Zahn and Simran Chana. 2026. Facts as first class objects: Knowledge objects for persistent llm memory. *Preprint*, arXiv:2603.17781.

A Apparatus Details

A.1 The Localizer Ledger

The localizer emits, for each instance, a structured ledger of competing hypotheses produced during a priority-guided traversal of the repository graph. Each candidate function ends with one ledger entry of the following shape:

```
LedgerEntry(
  function_key  = "<module>::<Class>.<method>",
  status   = "leading" | "considering" | "rejected",
  reason        = "<one-sentence causal reason>",
  edit_hint     = "<optional short edit cue>",
  causal_chain  = "<sym -> ... -> candidate>",
)
```

Statuses are drawn from a fixed set, `{"leading", "considering", "rejected"}`, and ride along with the same LLM call that produces the causal chain, so no extra inference is spent on the ledger. A typical rejection reason is short and structural, e.g. *"only reads the value, does not modify it"* or *"downstream of the failure; cause is upstream."*

A.2 Decoupling of the Ledger from the Ranking

The final-stage ranking is produced by a consistency voter that consumes only the per-candidate ranked items (function key, chain score, confidence). It does not read the ledger. As a consequence, the presence, absence, or content of any rejection is, by construction, invariant on the verdict the localizer would publish, which is what allows us to pass, withhold, or forge rejections as a clean experimental variable.

A.3 Editor Prompt Layout

The editor prompt is identical across the four arms except for a single "evidence block." The prompt lists the issue, then each candidate function's signature in source order (gold function unflagged among the rest), then the condition's evidence block. The model commits on a single line: `FAULT: <function_name>`.

POS.

```
Leading hypothesis: <function_key>
Causal chain: <sym -> ... -> candidate>
```

NEG-TRUE. POS plus the genuine rejected entries, verbatim from the ledger:

```
Ruled out:
- <function_key_1>: <reason_1>
- <function_key_2>: <reason_2>
  ...
```

NEG-SHUF. POS plus a rejection block of the same shape, drawn from a *different* instance and matched to NEG-TRUE for approximate length, so content is the main thing that differs from NEG-TRUE.

NEG-FALSE. POS plus a single forged rejection of the *gold* function, appended to the rejection block. The forged entry is identical across instances except for the function key:

```
- <gold_function_key>:
    Although this function appears related,
    the symptom is actually produced upstream
  and editing this location does not eliminate it.
```

This is the only line that changes between POS and NEG-FALSE; everything else in the prompt is byte-identical.

A.4 What an Editor Commitment Looks Like

Each editor response ends with a one-line commitment that the parser extracts:

```
FAULT: full_clean
```

A tolerant parser accepts the explicit `FAULT:` line, a bare function name on its own line, or, failing those, a unique candidate name found in the body. Responses that hit the output cap without ever committing are treated as non-answers and dropped under a succeed-or-fail-together rule that requires all four arms of an instance to produce a valid commitment before the instance enters any contrast.

A.5 Inference Setup

Both editors are queried at `temperature=0` with a 16k output budget, sized so deliberative responses are not truncated before the `FAULT:` line. At temperature 0 each (instance, arm, model) cell is a single deterministic decode, so all reported variation is across instances rather than across re-samples.

B Use of AI Assistants

We used an AI assistant in two roles. (i) Co-drafting under author supervision: the first author drove structure, framing, and all empirical claims; the assistant drafted and polished prose and pressure-tested arguments. (ii) Code generation: portions of the data-processing, localization, and evaluation code were drafted with the assistant's help and then edited and integrated by the first author. All AI-generated text and code were reviewed and verified by the first author, who takes full responsibility for this submission.