The Correct Answer Violates

How phrase-list scoring punishes the behavior it exists to reward, and what it did to one leaderboard

Matt Liotta · August 2026

Abstract

Agent-memory benchmarks commonly score responses against author-written phrase lists: a response containing a forbidden phrase is counted as having resurrected a dead fact or leaked restricted data. We show this scoring family systematically penalizes the behavior it exists to reward, and that the effect is large enough to overturn published conclusions.

The decisive test is cheap and requires no inference. We construct, for each of 400 real benchmark queries, the response a *correct* system would give — one that states the required content and then explicitly names the superseded value in order to rule it out ("it is not Friday; that was superseded"). Under the original scoring, **100% of these correct responses are flagged as violations, on every track.** Under corrected scoring, 0%. Separately, on one track a correct answer that merely uses the question's own topic vocabulary is flagged 66.7% of the time, because the forbidden list contains the words the question asks about.

Critically, this is **not** a verbosity effect. Generic filler text of 319 words trips the original scorer only 0.3% of the time. The bias is not correlated with length but with *engagement*: a response that discriminates live state from dead state must name the dead state to do so, and unbounded substring matching cannot distinguish that from a leak. The metric rewards evasion.

We audit StateBench, a conformance benchmark for stateful agents, and find six defects: judges inherited from the system under test, unbounded substring matching, forbidden phrases a correct answer must contain (444 of 4,163 in the release), blindness to negation, one metric computed from another's signal, and a decision extractor reading a bare "no" inside "now". To quantify the impact without confounding it with sampling, we generate each response once and score it twice. On `gpt-5.2-2025-12-11`, the exact published configuration, the resurrection metric falls on **all ten baselines** by 8.8 to 17.6 percentage points, and the between-baseline ordering does not survive. The judge-scored metrics serve as controls and behave as controls should: must-mention holds within 2pp, and decision accuracy moves by at most 6.4pp in no consistent direction.

Re-deriving the full leaderboard (two splits, three runs, 60 units) removes a claim from our own prior work: a reported leakage parity of 24.2% vs 24.1% ("within noise") becomes

14.0% vs 9.5%, a 4.4pp gap at roughly 3.5× the combined standard deviation, replicated on the held-out split. Separating the blended metric also reveals a real result it concealed.

The diagnostic is one afternoon's work for any benchmark: write the ideal answer by hand and score it. We release the harness.

---

1. Introduction

A memory benchmark asks: did the agent mention the address the user moved away from?

The natural implementation is a forbidden-phrase list. The scenario knows the old address was "123 Main St", so a response containing that string resurrects a dead fact. This is cheap, deterministic, and requires no judge. Most agent-memory evaluations use some form of it.

It has a failure mode with an uncomfortable structure. Consider the strongest possible evidence that a system has tracked state correctly:

> *"The meeting is **not** Friday — that was superseded. It moved to Thursday."*

This response demonstrates the exact discrimination the benchmark exists to measure. It also contains the string "Friday". Under substring scoring it is a violation.

Our first hypothesis was that this is a **verbosity** effect: longer answers contain more strings, memory architectures differ in verbosity, so the metric partly ranks answer length. §5.1 reports a controlled experiment that **disconfirms** this. Generic filler text of 319 words trips the scorer 0.3% of the time. Length is not the variable.

The variable is **engagement**. A response that discriminates live state from dead state must *name* the dead state to do so. A response that answers a question about a budget must use the word "budget". Both are correct behaviors, and both are what the phrase list catches. A response that says as little as possible about the scenario is scored best. **The metric rewards evasion.**

We quantify this directly (§6.1): across 400 real benchmark queries, a constructed *correct* answer that explicitly rules out the superseded value is flagged as a violation **100% of the time**, on every track. We then measure what this did to real results (§6.2–6.5), and re-derive the leaderboard.

This is an audit of our own benchmark and a correction to our own published results. The defects are not exotic, and the diagnostic that finds them costs an afternoon.

---

2. Where this sits in the memory-evaluation literature

Agent memory is not short of benchmarks, and the scoring functions they use are public. That is enough to say something concrete about exposure without re-running anyone's experiments.

Scoring families and their sensitivity. The mechanism we identify — a scorer that cannot distinguish *naming a value to reject it* from *asserting it* — is a property of surface-matching metrics generally, not of one implementation.

Scoring family	Can it distinguish "not X" from "X"?	Used by
Forbidden-phrase / must-not-mention	**No** — substring containment is negation-blind	StateBench
Required-phrase / must-mention	N/A (positive) — but rewards incidental inclusion	StateBench and others
Token-level F1 with stemming	**No** — "not Friday" and "Friday" share the token	LoCoMo
BLEU-*n* / ROUGE-*n* at low *n*	**No** at n=1; partially at higher n	LoCoMo
LLM-as-judge, binary	In principle yes; verbosity and self-enhancement biases documented separately	LongMemEval, Mem0, most recent work

LoCoMo (Maharana et al.) reports token-level F1 with Porter stemming, BLEU-1 and ROUGE-L over 1,986 QA pairs. Unigram-level overlap metrics cannot represent negation: a reference answer "Thursday" scored against "not Friday, it moved to Thursday" and against "Friday" differs only in precision, not in whether the model got the state right. The failure is less catastrophic than a forbidden-phrase list (F1 does not flip a correct answer to a violation) but it is the same blindness.

LongMemEval (Wu et al., ICLR 2025) covers 500 questions across five abilities including *knowledge updates* and *abstention* — the two abilities most exposed to this class of error, since both require a system to signal that something is *not* the case. Its use of judge-based scoring rather than surface matching is the right structural choice; whether the judges handle negation and abstention reliably is a separate question we do not measure here.

Adjacent work on evaluation validity. MemDelta (Wang, 2026) makes a complementary argument from a different direction: that "reported gains often mix changes in the memory method with changes in the language model, embedding model, or retrieval pipeline, making it unclear what is actually being measured." Varying one component at a time on LongMemEval-S, it finds that swapping only the embedding model shifts accuracy by 6.2pp and that one variable flips the conclusion about whether a memory system beats retrieval. Its recommendations — fix embedding models across comparisons, stratify by model family, report write-path cost — are about *pipeline* confounds; ours are about *scoring* confounds. Neither subsumes the other, and a benchmark can be exposed to both independently. Taken together they suggest that published rankings in this area should be treated as provisional pending controlled replication.

What we do not claim. We audited one benchmark. We have not scored anyone else's systems, and the magnitudes here are specific to StateBench's phrase lists. What we claim generalizes is the *dia-*

gnostic: §8 item 1 costs an afternoon and would surface this class of defect anywhere it exists. We would be interested to be told it does not.

---

3. The benchmark under audit

StateBench (Parslee, 2025) evaluates whether an agent maintains correct state over time, across thirteen tracks: supersession, scope permission, enterprise privacy, hallucination resistance, authority hierarchy, repair propagation, and others. Each test case is a timeline of events terminating in a query with ground truth: a correct decision, phrases that must appear (`must_mention`), and phrases that must not (`must_not_mention`).

Four metrics are reported:

- **Decision Accuracy** — correct decision, via deterministic extraction with an LLM fallback.
- **SFRR** (Superseded Fact Resurrection Rate) — how often invalidated facts reappear.
- **Must Mention Rate** — required information appears.
- **MNM Violation Rate** — forbidden information appears.

Two prior papers report results on it (Liotta 2025, 2026). This audit was undertaken as a prerequisite for a third, which required resolving effect sizes the existing harness could not.

---

4. Six defects

4.1 The judge was inherited from the system under test

The evaluation harness selected the judge provider from the provider of the model being evaluated. OpenAI arms were graded by `gpt-4o-mini`; Anthropic arms by `claude-3-haiku`. Each model family was graded by a judge from its own family.

Every cross-model comparison in the published tables therefore mixes two different graders. The papers state the judge is GPT-4o-mini throughout, which was the intent; the code did something else.

This matters most for the headline cross-model claim in Liotta (2026) — that one model family outperforms another under the same architecture — which is a comparison across two graders.

Fix: the judge is resolved globally, never from the system under test, and its identity is recorded with every result.

4.2 Unbounded substring matching

Phrase matching used plain containment. Therefore:

Forbidden phrase	Spuriously matches
`"by"`	"nearby", "standby"
`"HP"`	"PHP"
`"100"`	"1000", "2100"
`"$150"`	"$1500"
`"10%"`	"110%"

Fix: boundary-aware matching, with the boundary applied per edge according to whether that edge is a word character — so `"$45"` still does not match inside `"$450"`, and `"10%"` does not match inside `"110%"`. A naive `\b` on both sides is wrong for phrases that begin or end with punctuation.

4.3 Forbidden phrases that cannot discriminate

Some forbidden phrases carry no information about correctness. Three classes:

1. **No content.** `"$"` appears as a forbidden phrase in 5 test-split queries, and `"#"` in one. Any response quoting any dollar amount violates.
2. **Ordinary English.** `"by"`, `"to"`, `"from"`, `"using"` match nearly any sentence.
3. **The vocabulary of the correct answer.** This is the damaging one.

On the `hallucination_resistance` track, the correct behavior is to *ask for the missing detail*. One template's query is:

> *"What's our agreed timeline and budget for this CRM implementation?"*

and its forbidden list is, in full: `"$"`, `"week"`, `"month"`, `"quarter"`, `"deadline"`, `"budget"`, `"by"`, `"before"`, `"approved"`, `"allocated"`. A correct response — "no timeline or budget was discussed" — **violates by construction**, on `budget` alone. The track penalized the behavior it existed to test.

Across the v1.0 release — 1,410 unique timelines, deduplicated by timeline id because `full.jsonl` is the union of the splits plus 143 held-out items — **444 of 4,163 forbidden phrases (10.7%)** fail an admissibility test. They are heavily concentrated: 340 are on `hallucination_resistance`, where they account for **40.1%** of that track's entire forbidden list, and 69 more (25.6%) are on `supersession`. Six tracks have none. `experiments/phrase_admissibility_audit.py` regenerates this.

Fix: an admissibility gate. A phrase is admissible only if a correct response could not plausibly contain it. Inadmissible phrases are excluded from scoring *and* from the denominator — keeping them in the denominator deflates the violation rate as surely as scoring them inflates it — and are recorded so audits can see what was dropped. Templates emitting them were rewritten to name fabricated specifics ("we agreed", "$50,000", "6 weeks") rather than topic vocabulary.

4.4 Negation was invisible

Consider a response to a supersession scenario:

> *"The meeting is **not** Friday; it moved to Thursday."*

This contains "Friday", so it scored as resurrecting the superseded value. But distinguishing dead state from live state, explicitly, is precisely the capability SFRR exists to reward. The metric scored its own success condition as failure.

Fix: a forbidden phrase appearing *only* under a negation cue is recorded but not counted. The rule is deliberately conservative — a single un-negated occurrence anywhere still counts as a violation — so it cannot launder a genuine leak.

4.5 SFRR was not measuring resurrection

SFRR was set from *any* `must_not_mention` violation:

```
result.resurrected_superseded = len(result.must_not_mention_violations) > 0
```

But forbidden phrases across tracks detect different failures. On `enterprise_privacy` they detect restricted-data leaks; on `hallucination_resistance` they detect fabrications. Neither is resurrection. SFRR and MNM were two normalizations (per-query and per-phrase) of one undifferentiated signal.

Fix: phrases carry a failure class, inferred from the track or set explicitly. SFRR counts only superseded-state violations; leakage and fabrication are reported as separate rates. §5.3 shows this separation is not cosmetic — it surfaces a result the blend concealed.

4.6 A bare "no" inside "now"

The decision extractor matched yes/no signals as bare substrings. `"no"` is a signal, so `"now"`, `"know"`, `"noted"`, and `"nothing"` all set it. Verified:

> *"Right now, the budget is $150,000 and the vendor is approved." → extracted **"no"***

Worse, because extraction *succeeded*, the LLM fallback — which exists to rescue exactly these cases — never ran. Blast radius on the release is bounded (20 of 251 test-split queries are binary), but on binary-heavy experiment suites it is not.

Fix: boundary matching on all signal terms, including the first-position tiebreak.

---

5. Method: paired scoring

To measure the impact of the corrections, comparing two fresh evaluation runs would be wrong: the difference would confound the scoring change with model sampling noise, and — because the published tables use a model we can no longer assume is byte-identical — with model drift.

Instead: **generate each response once, score it twice.** The legacy scorers are reimplemented alongside the corrected ones and applied to the same response set.

This has three properties worth stating:

1. For the deterministic metrics (SFRR, MNM) the delta is **exact** — no sampling noise at all, because both scorings are pure functions of the same string.
2. For metrics that invoke the LLM judge (decision accuracy, must-mention) the delta carries judge sampling noise in both passes, and we mark those figures accordingly.
3. It halves the inference cost of the audit.

The legacy scorers live only in the audit harness, not behind a flag in the library. Production should carry one scoring semantics, and the legacy one is the incorrect one.

---

6. Results

6.1 The correct answer violates

The decisive experiment uses no model at all. For 400 real queries drawn from the release, we construct the response a correct system would give, directly from the ground truth, and score it under both scorings. Two variants:

- **plain** — states the required (`must_mention`) content and echoes the question's vocabulary.
- **negating** — the same, plus an explicit rejection of the superseded value: *"Note that it is not X — that value was superseded."*

Ground truth is exact: neither variant resurrects anything, so every violation is a false positive.

Constructed correct answer	Legacy	Corrected
states required content + question vocabulary	4.5%	3.5%
… and explicitly rules out the superseded value	**100.0%**	**0.0%**

The second row is the finding, and it holds on **every one of the twelve tracks**. A system that demonstrates state discrimination in the most direct way available — naming the dead value to reject it — is scored as having resurrected it, always.

The per-track breakdown isolates the second defect:

Track	plain (legacy)	negating (legacy)	negating (corrected)
`hallucination_resistance`	**66.7%**	100.0%	0.0%
`supersession_detection`	6.7%	100.0%	0.0%
`scope_leak`	6.7%	100.0%	0.0%
`authority_hierarchy`	6.7%	100.0%	0.0%
`supersession`	4.5%	100.0%	0.0%
`scope_permission`	3.8%	100.0%	0.0%
(six others)	0.0%	100.0%	0.0%

On `hallucination_resistance`, two thirds of correct answers are flagged **without any negation at all** — merely for using the vocabulary the question asks about (§4.3).

The control that rules out verbosity. We sweep generic business filler — text containing no scenario entity, amount, or date — against the same phrase lists:

Decoy length (words)	Legacy FPR	Corrected FPR
10	0.0%	0.0%
20	0.3%	0.0%
40	0.0%	0.0%
80	0.3%	0.0%
160	0.7%	0.0%
319	0.3%	0.0%

Every length is reported; the sequence is non-monotonic and never exceeds 0.7%, which is the point — a 32× increase in length does not produce a trend.

Length alone does essentially nothing. Our initial verbosity hypothesis was wrong, and we report it as disconfirmed. The scorer is not sensitive to how *much* a response says; it is sensitive to whether the response engages with the scenario's own content.

Reading the two numbers together. The 100% is a *conditional* rate: if a system explicitly rejects the dead value, it is always flagged. The 8.8–17.6pp population effect in §6.2 is smaller because real systems do not always phrase their answers that way. The conditional rate bounds the failure mode; the population rate measures how often it is triggered in practice. Both matter: the first says the metric is broken for a specific correct behavior, the second says that behavior is common enough to move leaderboards.

6.2 The instrument delta

`gpt-5.2-2025-12-11`, judge `gpt-4o-mini` — the exact published configuration — over 104 stratified dev timelines (n≈125 queries per baseline).

Configuration check. Legacy decision accuracies track the published dev table: `state_based` 88.0% (published 89.1%), `transcript_replay` 78.4% (81.2%), `no_memory` 26.4% (23.8%). The configuration reproduces.

Baseline	SFRR legacy → corrected	Δ
`state_based_no_supersession`	26.4% → 8.8%	**−17.6pp**
`transcript_replay`	24.8% → 8.0%	−16.8pp
`rag_transcript`	28.8% → 12.8%	−16.0pp
`state_based`	24.8% → 8.8%	−16.0pp
`rolling_summary`	25.6% → 10.4%	−15.2pp
`transcript_latest_wins`	19.2% → 6.4%	−12.8pp
`fact_extraction`	28.0% → 16.0%	−12.0pp
`fact_extraction_with_supersession`	22.4% → 12.0%	−10.4pp
`no_memory`	11.2% → 1.6%	−9.6pp
`memgine`	22.4% → 13.6%	**−8.8pp**

SFRR falls on every baseline, without exception, by 8.8 to 17.6pp. Roughly half to three-quarters of published SFRR was not resurrection.

Decision accuracy moves by at most 6.4pp (`transcript_replay`) with no consistent direction — four baselines fall, six rise — consistent with judge sampling noise at this n. Must-mention is tighter, within 2pp on every baseline, as expected for a metric the decision extractor does not feed. **Only the phrase-list metrics move systematically**, which is what the defect analysis predicts.

6.3 The ordering does not survive

The corrections are not a uniform rescaling. Because the artifact share differs per baseline, the between-baseline ordering changes:

SFRR rank (best)	Legacy	Corrected
1	`no_memory`	`no_memory`
2	`transcript_latest_wins`	`transcript_latest_wins`
3	`fact_extraction_with_supersession`	`transcript_replay`
4	**`memgine`**	`state_based_no_supersession`
5	`state_based`	`state_based`
…		
9	`fact_extraction`	**`memgine`**

`memgine` moves from 4th to 9th of ten; `transcript_replay` moves from 6th to 3rd.

The mechanism is engagement, not length. Per §6.1, the artifact fires when a response names scenario content — most sharply when it names the superseded value in order to reject it. Baselines differ in how much they engage: a system given rich context discusses the situation, including what changed; a system given little context answers narrowly. `memgine`'s curated context yields terse, direct answers that rarely restate the dead value, so little of its legacy SFRR was artifact — most of it was real resurrection. The context-rich baselines discussed the supersession and were charged for it.

An unsettling confirmation appears in the prior work itself. Liotta (2026) §7.3 explains one model's higher SFRR as "[it] include[s] more contextual information in responses, mentioning superseded facts **even when they correctly reason about current values**" (emphasis ours). That sentence describes the artifact precisely — it even names the condition, *correct reasoning* — and then attributes it to the model rather than to the metric.

6.4 The corrected leaderboard, and a claim that does not survive

Re-deriving in full — both splits, three runs, 60 evaluation units:

Baseline (dev)	Decision Acc	SFRR	Leakage	Must Mention
`memgine`	**96.8% ± 0.0%**	14.0% ± 0.2%	**1.9%**	81.0% ± 0.1%
`state_based_no_supersession`	91.5% ± 0.6%	**9.5% ± 1.2%**	5.4%	80.8% ± 0.5%
`state_based`	87.6% ± 0.5%	12.9% ± 0.9%	7.4%	79.3% ± 0.2%
`fact_extraction_with_supersession`	84.0% ± 1.1%	13.4% ± 0.8%	4.6%	65.2% ± 1.5%
`rolling_summary`	83.9% ± 0.7%	6.9% ± 0.6%	5.1%	67.1% ± 1.2%
`rag_transcript`	83.5% ± 1.0%	10.8% ± 0.5%	4.8%	71.0% ± 0.3%
`transcript_replay`	83.2% ± 0.2%	7.9% ± 0.8%	4.6%	70.6% ± 0.5%
`fact_extraction`	77.6% ± 2.0%	14.1% ± 1.1%	4.4%	64.3% ± 0.7%
`transcript_latest_wins`	69.5% ± 1.3%	7.4% ± 0.5%	2.3%	42.7% ± 1.1%
`no_memory`	26.3% ± 1.1%	4.7% ± 0.5%	2.3%	9.3% ± 0.4%

What survives. The accuracy result is confirmed and slightly strengthened: `memgine` leads on both splits (96.8% dev, 94.2% test), by a wide margin. Nothing in the correction touches it.

What does not. Liotta (2026) §7.1 claims the system "breaks this correlation — improving accuracy substantially without increasing leakage," supported by SFRR 24.2% ± 1.3 versus 24.1% ± 0.8 — a +0.1pp difference called "within noise." Under corrected scoring the same comparison is **14.0% ± 0.2 versus 9.5% ± 1.2: a 4.4pp gap at roughly 3.5× the combined standard deviation**, replicated on the held-out test split (12.7% vs 9.2%).

The parity was an artifact of unequal artifact shares between the two arms. The claim does not hold.

What the blend concealed. Splitting SFRR into resurrection, leakage, and fabrication is not bookkeeping. `memgine` has the **lowest leakage of any baseline** at 1.9%. The two nearest, `no_memory` and `transcript_latest_wins` at 2.3%, achieve that trivially by carrying almost no context; among state-aware baselines the range is 4.4–7.4%. Engine-level access control produces exactly this signature, and the blended metric could not express it. Correcting the instrument cost the system one claim and gave it another.

6.5 The bias is not static: model capability interacts with it

Repeating the corrected evaluation on a current-generation model (`gpt-5.6-sol`, same scoring, same judge) shows SFRR falling a further 0.8–7.5pp on nine of ten baselines; `transcript_latest_wins` is the exception and rises 1.7pp. More consequentially for anyone reading an older leaderboard: the accuracy premium of the best architecture over a simpler one collapses from 9.1pp to 0.7pp on dev, and from 7.3pp to 0.1pp on test.

Leaderboards computed on one model generation do not transfer to the next, even holding the instrument fixed. This compounds the measurement problem: a biased metric evaluated on an obsolete model is two removes from the claim it is used to support.

---

7. What went wrong, structurally

The defects share a shape. Each is a **cheap proxy substituted for an expensive judgment**, where the proxy's failure mode correlates with something the evaluation is also varying.

- Substring containment proxies for "mentioned this fact" — and correlates with length.
- Track membership proxies for "which failure class" — and collapses three failures into one rate.
- Signal-word presence proxies for "made this decision" — and collides with common words.
- The provider under test proxies for "an available judge" — and correlates with the model family being graded.

None required sophistication to find. What they required was *looking* — specifically, reading actual model responses next to their scores rather than reading aggregate rates. Every defect in §3 was found by printing one response and its score side by side.

The organizational failure is that the metrics were never adversarially tested against their own success condition. It is worth asking of any benchmark metric: **what does a perfectly correct response look like, and does it score well?** On `hallucination_resistance`, it did not, and could not.

---

8. A checklist

For benchmark designers using phrase-list scoring:

1. **Write the ideal response by hand and score it.** If it does not achieve a perfect score, the ground truth is wrong. Do this per track.
2. **Match on word boundaries**, with per-edge handling for phrases starting or ending in punctuation.

3. **Audit forbidden phrases for discriminative power.** Any phrase a correct answer could contain is not a violation signal. Exclude it from scoring *and* the denominator.
4. **Handle negation.** A phrase under explicit negation is usually evidence of the capability under test, not a violation of it.
5. **One metric, one failure.** If two metrics are computed from the same signal, they are one metric.
6. **Pin the judge globally.** Never derive it from the system under test. Record it with every result.
7. **Test the rejecting answer specifically.** Not just the ideal answer — the ideal answer *phrased as an explicit rejection* ("it is not X, that was superseded"). This is the single highest-yield probe: it is where the metric and the capability diverge most, and it caught the total failure in §6.1 that a plain ideal answer did not.
8. **Version the scoring.** Results are (system, model, instrument) triples. Record all three, and treat an instrument change as invalidating comparisons across it.

Items 1 and 7 are the cheapest and highest-yield: together they cost an afternoon and would have caught every defect in §4 except the inherited judge, which item 6 catches by inspection.

---

9. Limitations

Single-benchmark measurement. We measured one benchmark. §2 argues from published metric definitions that the negation blindness is a property of surface-matching scorers generally — token-F1 and BLEU-1 cannot represent "not X" either — but that is an argument from metric algebra, not a measurement of anyone else's results. We have not scored other systems and make no claim about their magnitudes.

The correct-answer probe is constructed, not sampled. The 100% figure in §6.1 is a conditional: *if* a system phrases its answer as an explicit rejection, it is always flagged. We constructed those answers from ground truth rather than observing them from a model, so the number bounds the failure mode rather than estimating its population frequency. The 8.8–17.6pp figure in §6.2 is the population effect and is much smaller, because real systems phrase answers that way only some of the time.

Our first hypothesis was wrong. We predicted a verbosity effect and designed a decoy experiment to confirm it; it disconfirmed it (§6.1). We report this because the corrected mechanism — engagement rather than length — has different implications for which benchmarks are exposed, and a reader replicating our reasoning should not repeat the error.

The corrected leaderboard covers OpenAI models only. The Anthropic arm of the published tables remains uncorrected, so the cross-model claims in the prior work are flagged as confounded (§3.1) but not re-derived.

The negation heuristic (§3.4) is lexical and will miss constructions its cue list does not cover. It is conservative in the safe direction — it under-credits rather than over-credits — but it is not a semantic entailment check.

The admissibility gate (§4.3) uses a hand-built stop list plus a length rule. It is a heuristic standing in for "could a correct answer contain this," which is ultimately a judgment about the scenario. We chose to err toward rejecting phrases, which costs sensitivity.

Judge-mediated metrics carry sampling noise we do not fully characterize; we ran three seeds for the leaderboard and one for the paired delta.

---

10. Conclusion

We audited our own benchmark, found that its headline safety metric was substantially measuring response length, and corrected it. The correction removes a claim from our prior work — a reported parity in leakage was an artifact — and adds one the previous metric had concealed.

The general lesson is not that phrase lists are unusable. It is that a cheap proxy is acceptable only when its failure mode is uncorrelated with the capability under test — and here it was *anti*-correlated. The clearest signal a system can emit that it has tracked state correctly is to name the dead value and rule it out; that is exactly the signal the scorer read as failure.

The condition is checkable in an afternoon: write the ideal answer, write it again as an explicit rejection, and score both.

We release the paired-scoring harness so that other benchmarks can measure their own exposure without re-running generation.

References

Maharana, A., et al. (2024). *Evaluating Very Long-Term Conversational Memory of LLM Agents* (LoCoMo).

Wang, K. (2026). *MemDelta: Controlled Baselines and Hidden Confounds in Agent Memory Evaluation.* arXiv:2606.29914.

Wu, D., et al. (2025). *LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory.* ICLR 2025. arXiv:2410.10813.

Liotta, M. (2025). *Beyond Conversation: A State-Based Context Architecture for Enterprise AI Agents.*

Liotta, M. (2026). *Memgine: A Deterministic Memory Engine for Stateful AI Agents.*

Parslee. (2025). *StateBench: A Conformance Test for Stateful AI Systems.*

Appendix A: Reproduction

```
# Paired scoring: generate once, score under both semantics
uv run python experiments/with_car_secret.py -- \
    python experiments/instrument_delta.py \
    --model gpt-5.2-2025-12-11 --judge-model gpt-4o-mini --limit 104

# Corrected leaderboard (resumable; 60 units)
uv run python experiments/with_car_secret.py -- \
    python experiments/rederive_v11.py \
    --model gpt-5.2-2025-12-11 --splits dev test --runs 3
```

Per-unit results: `experiments/results/v11_rederived/`, `experiments/results/g52_*.json`.
Regression tests for every fix in §3: `tests/test_instrument.py`.