AGENT MEMORY CAN ONLY FORGET WHAT ITS DELETION PATH CAN SEE

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing forgetting evaluations can report near-perfect deletion while the forbidden fact remains fully retrievable. Across three widely deployed agent memory systems, surface-form retrieval scoring gives **97–100 %** while direct queries for the same forbidden fact give **0 %**: the fact was rewritten rather than removed, and the old scoring credited the rewrite. What a memory system can forget is bounded by what its *deletion path* — the control plane responsible for memory mutation — can see; existing benchmarks evaluate the recall plane instead. We show this with a controlled ablation that holds the store, model and prompts fixed and moves only the call site. Write-time knowledge helps only when the deletion path can read it: granting the deletion step access to the same annotations, with no new information and no new authority, recovers part of the gap. But a model placed before the request cannot see what the user later asks to *remove*. On the 345 cases that measure forgetting, mutation-time placement raises the pass rate from **71.0 %** to **91.3 %** (+20.3 pt); including a separate 40-case partial-edit diagnostic, the full suite rises from **63.6 %** to **88.8 %**. The same contract takes a third-party store from **0/38** to **38/38** on the affected canonicalization categories. We release FORGETEVAL, a 1385-case benchmark with deterministic scoring and an `Adapter` Protocol for heterogeneous stores. Reliable forgetting depends on the control plane and cannot be inferred from retrieval quality alone.

1 INTRODUCTION

An agent's memory has two planes: a *recall plane* that retrieves stored facts, and a *control plane* that mutates them through supersede, release and purge. This paper's claim is that *where* an LLM sits relative to these two planes determines which forgetting failures an agent can recover from. Whether the pipeline contains one at all does not. The title runs two things together that the rest of this paper takes apart: whether the deletion path can *read* what a write-time model wrote, and whether it can *see* the request it is being asked to serve. The first is buyable; the second is not. The field has saturated the recall plane and left the control plane untested: the dominant memory frameworks (Chhikara et al., 2025; Packer et al., 2023; Letta contributors, 2024; Rasmussen et al., 2025; Gutiérrez et al., 2024; Xu et al., 2025; Cognee contributors, 2024) race to recall harder and never lose a fact, and the benchmarks follow — LongMemEval (Wu et al., 2025) scores answer accuracy, MTEB (Muennighoff et al., 2023) and BEIR (Thakur et al., 2021) are stateless. None probe whether an *external memory store* can be commanded to forget. That question is orthogonal to parametric unlearning (Maini et al., 2024; Jin et al., 2024), which edits weights rather than a store the agent still retrieves from.

In production, that is the failure mode that bites. A password the user rotated three months ago is still suggested. A customer who exercised GDPR Article 17 (European Parliament and Council, 2016) is still in the recommender's candidate pool. A user's job title surfaces in three contradictory versions across sessions. A one-time verification code lives forever next to long-term preferences. Every retrieval succeeded. That is the problem: the store retrieved correctly when the application needed it to retrieve nothing.

One case, which we will return to, makes the shape concrete. The store holds *User lives in Berlin and works at Stripe as a backend engineer*, and the user relocates to Madrid. The city must go; the employer must stay; and both live in one stored row. Deleting the row loses Stripe, keeping it leaves Berlin reachable, and in the ablation this paper runs — one store, one model, one prompt, only the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

call site moving — *every* arm that does not see the relocation request scores **0** of the forty cases shaped like this, and the one that does scores 27. Appendix V maps the production failures above to the attack category that reproduces each, and Appendix E walks through worked cases.

The two planes are not duals. One of the stores we evaluate is mid-field on a conversational-recall benchmark and above our own store there, yet fails every control-plane test we run: recall accuracy says nothing about whether a store can be commanded to forget (§5.4).

A model on the write path does not help. If a deterministic store cannot recognise that `ALICE@ACME.IO` and `alice@acme.io` name one person, the natural response is to put a language model in the pipeline and have it normalise entities as facts arrive. We ran that, holding everything else fixed — same store, same model, same prompt family — and it buys **nothing**: **63.9 %** against a deterministic **63.6 %**, with the very categories normalisation targets left exactly where they were (§5.5). The model is called on every write and files the right canonical forms; the control plane simply cannot reach them.

Write-time knowledge is not useless; it is *inert*. A model placed before the control plane learns what things *are*, and that knowledge transfers only as far as the deletion path can read it — the same annotation is worth nothing until the deletion step is allowed to consult it, and then it recovers most of what was failing, with no new information and no new authority added. But a model placed there never sees what the user asked to *remove*, and no amount of preparation supplies that: *request-conditioned information cannot be produced before the request exists*. The forty cases that turn on removing one clause of a stored memory are **0/40** in every write-time arm of the ablation and **27/40** in the one that sees the request. That is what this paper measures, and it is why comparing whole memory systems — which differ in storage, retrieval and extraction as well as in call site — cannot establish it.

For a practitioner the consequence is narrow and actionable. Putting a language model on the write path to normalise entities, which is what most deployed stores do, does nothing for forgetting on its own. Either the deletion path must be able to read what that model wrote, or the model has to run at mutation time, where it can see the request.

We make three contributions.

(1) A controlled identification of where the model has to sit (§5.5, Table 2). Holding store, model and prompts fixed and moving only the call site, write-time normalisation buys nothing (63.9 vs. 63.6 %) while the same model at mutation time buys $+20.3$ pt on the 345 cases that measure forgetting — capability attaches to the operation the model mediates, not to its presence in the pipeline. The effect is architectural rather than store-specific: an unrelated backend given the same narrow JSON contract replicates it, and it survives an example-free prompt, an NLI re-score of every case and six hook models across five vendors (§5). We situate this against thirteen deployed configurations (Fig. 1), reported as the observational evidence they are: their rows differ in storage and retrieval as well as call site, which is precisely why the controlled ablation is needed.

(2) FORGETEVAL, a 1385-case benchmark (1000 template + 385 adversarial, of which 132 hand-crafted) that decomposes forgetting into 5 structural families and 10 attack categories. Scoring is deterministic substring match over top-k recall, forbidden content checked under direct probing (§5.3), so **no LLM judge sits in the evaluation loop**. The judge is confined to data admission, a binary well-formed/ill-formed judgment on each case, corroborated by 10-annotator agreement on that judgment (Fleiss' $\kappa = 0.958$) and a cross-family judge audit (§3.3).

(3) A six-method `Adapter` Protocol with N/A scoring, so a store lacking `supersede` or `purge` can be compared honestly against one that has them. The Protocol is a *behavioural* contract: a backend composing supersede from `add`+`delete` passes the same tests as one with a native primitive (§4). That is what makes nine heterogeneous stores, in thirteen configurations, commensurable.

2 RELATED WORK

Recall-axis benchmarks. LongMemEval (Wu et al., 2025) scores conversational recall on 500 questions; even its *knowledge-update* category asks whether the latest fact is retrieved, not whether the superseded one is gone. MTEB (Muennighoff et al., 2023) and BEIR (Thakur et al., 2021) are stateless. AMA-Bench (Zhao et al., 2026), MemoryArena (He et al., 2026), and EvoMemBench (Wang et al.,

2026b) score end-to-end agent behaviour on the recall/use axis. None can express "this should now retrieve nothing".

The forgetting axis is being measured at five different layers. Ours is the *memory-store primitive* layer, and the 2026 literature converges on the axis from four others. *Agent-behaviour:* Hu et al. (2026) (ICLR 2026) is the closest prior benchmark — its FactConsolidation task scores single-fact supersession by substring match. We differ in granularity (five primitive families and 10 adversarial categories vs. single-fact supersession), what is scored (the retrieval set, must-not-contain included, vs. a generated answer), and portability (a 6-method Protocol with N/A scoring vs. whatever agent is supplied); our cross-evaluation on all 400 FactConsolidation questions (Appendix O) finds the LLM-hook variants score *identically* to their deterministic backbones at every bucket — a third-party confirmation that the two benchmarks load on different axes. *Aggregate-metric:* Uddin et al. (2026) introduces FAMA, a scalar penalizing obsolete-memory reuse.[1] *Governance:* Ren et al. (2026) evaluate agent-facing active forgetting jointly with multi-principal access control and find that no method achieves utility, access control, and reliable forgetting together; Wang & Zhang (2026) show deletion is incomplete across representations, recovering 12–18 % of nominally deleted multimodal facts from correlated survivors; Ding et al. (2026) survey the governance axes. *Parametric:* Gu et al. (2026a) measure forgetting dynamics in model weights under continual editing. These layers are complementary and none resolves ours: GateMem asks *whether a deployment honours a deletion request end-to-end*, MemLeak asks *what survives a deletion*, ForgetBench asks *what weights retain*; we ask *which architectural placement of the LLM makes the store capable of the deletion in the first place*. Wang (2026) gives a formal write-time correctness contract for the same control-plane operations, complementing our empirical characterization.

Memory frameworks. MEM0 (Chhikara et al., 2025) uses LLM-driven ADD/UPDATE/DELETE routing; its DELETE is contradiction-triggered overwrite, not a user-issued forget primitive. MemGPT (Packer et al., 2023) and Letta (Letta contributors, 2024) paginate between main and archival memory; Zep (Rasmussen et al., 2025) maintains a temporal knowledge graph with edge invalidation; HippoRAG (Gutiérrez et al., 2024) uses PageRank over an extracted KG; A-MEM (Xu et al., 2025) applies Zettelkasten linking. Among these only MEMPALACE (MemPalace contributors, 2026) takes the opposite stance to MEM0: verbatim retention is the feature, and removal is keyed by drawer ID or source file rather than by content.

Forgetting elsewhere. Cognitive psychology (Ebbinghaus, 1885; Bjork, 1972; Anderson et al., 1994) establishes the phenomenology. Machine unlearning (Cao & Yang, 2015; Bourtoule et al., 2021; Jin et al., 2024; Maini et al., 2024; Tian et al., 2024) and model editing (Meng et al., 2022; 2023; Zhong et al., 2023) operate on parameters — Gupta et al. (2024) show mass editing induces catastrophic forgetting of unrelated facts, and Wang et al. (2026a) extend removal jointly to parameters and memory; we require no parameter access. Nearest to us, Tan et al. (2025) evaluates factual vs. reflective memory and Gu et al. (2026b) propose a forgetting *taxonomy* validated on one system; Alqithami (2025) score privacy-aware forgetting *policies* — what a system should retain — where we score whether the store can execute the resulting instruction at all, and Zhang et al. (2023) survey GDPR right-to-be-forgotten approaches. FORGETEVAL occupies a distinct cell: a primitive-level decomposition of the memory-store surface, scored deterministically, with an Adapter Protocol that makes heterogeneous stores comparable without an LLM judge in the evaluation loop.

3 FORGETEVAL

3.1 FIVE FAMILIES

Each family probes one structural property a production memory system must exhibit. **Supersession:** a new fact wins recall; the old fact leaves top-k (failure: confabulation across both). **Decay:** a released fact (TTL, OTP-consumed) stays out of top-k. **Amnesia:** forget every fact about one entity, siblings survive — the hard part is *width control*. **Purge:** hard-delete by identifier (GDPR Article 17);

[1] Their benchmark is named "Memora"; to disambiguate from the contemporaneous retrieval-method paper of the same name (Xia et al., 2026), we cite by title.

semantic similarity is the wrong primitive. **Drift:** a chain of supersedes where only the latest wins and intermediates are unreachable.

Each case is a single dataclass: `setup_facts` (inscribed with distractors), `mutations` (supersede / release / purge calls), `final_query`, `must_contain`, `must_not_contain`. A case passes iff `must_contain` $\subseteq$ top-10 blob *and* `must_not_contain` $\cap$ top-10 blob $= \emptyset$. No LLM judge.

3.2 TEMPLATE SUITE (1000 CASES)

Each family has four sub-templates the generator cycles through (e.g. supersession has `job`, `theme`, `diet`, `long_form`). Generation is deterministic given seed; no floating-point sources, no time-of-day, no training-set contamination (entity pools are short proper nouns). At seed=42 with 4 distractors per case the suite is 1000 cases. The current work focuses on the English template suite and the 385-case adversarial layer; multilingual template extensions are left for future work.

3.3 ADVERSARIAL LAYER (385 CASES)

The template suite has flat difficulty: every case is the same sub-template plus i.i.d. entity substitution. We complement it with a hand-crafted layer, LLM-assisted in expansion and oracle-validated, across 10 attack categories:

- **substring_trap**: must-not substring embedded in a distractor.
- **prefix_collision**: two identifiers share a long common prefix (`alice@x` vs. `alice.smith@x`).
- **paraphrase_supersession**: new fact lexically distant from old.
- **negation_trap**: negated fact must not be confused with affirmative.
- **temporal_qualifier**: date-stamped supersession chains.
- **shared_attribute**: two entities share an attribute; forgetting one preserves the other's link.
- **compound_fact**: one sentence carries two distinct-topic facts; partial supersede must preserve the other. This tests for a partial-edit primitive, so a system that can express one neither natively nor by composition fails outright (Appendix I).
- **identifier_obfuscation**: same identifier in different surface forms (case, whitespace, separators).
- **cross_lingual_identifier**: same entity under different scripts (e.g. romanized vs. source-script form of a personal name).
- **recursive_supersession**: chain where the latest state matches an earlier-superseded one (Chrome→Brave→Chrome).

We target 40 cases per category, admitted by a protocol that keeps the oracle decoupled from the systems under evaluation: 132 hand-crafted core cases plus 253 LLM-drafted ones. **Stage 1 (structural):** reject malformed JSON, unknown families, and self-substring-traps (a `must_not_contain` substring appearing in a non-targeted setup fact). **Stage 2 (independent LLM-judge):** Qwen-2.5-72B — a different family from the hook LLM — traces the mutations and admits a case only if every `must_contain` string survives, no `must_not_contain` string does, and the final query is unambiguously answerable. A post-hoc label partition (Appendix F.1) is reported for transparency and enters neither admission nor any main table.

Judge precision and its limits. The Qwen judge admits **96/112 (85.7 %)** of the hand-crafted core as it stood at audit time, before the 20 identifier_obfuscation cases that bring it to 132; manual review of all 16 rejections finds *zero bench bugs* (Appendix F.2). The rejections are over-rejections, dominated by (A) *semantic equivalence* — 11 cases where the judge applies literal substring matching to the very categories that test canonicalization — plus 1 multi-row-scope and 4 computational errors. This matters for interpretation: **admission failure shrinks the bench rather than contaminating it**.

Multi-annotator agreement. We collected independent label sets from **10 NLP/CS-trained annotators** on a 100-case stratified sample (10/category, mixing 46 hand-crafted core + 54 LLM-drafted cases; no pre-submission consultation). Fleiss' $\kappa = \mathbf{0.958}$ ("almost perfect"); observed agreement 99.1 % over 1000 labels. The LLM-judge agrees with human majority on **79/100**; the

21 disagreements decompose cleanly: 8 cases where judge said `ill` but humans unanimously `wf` recover the audit's mode (A)/(B)/(C) failures (Appendix F.2), and 13 cases where judge said `wf` but humans `ill` cluster on compound_fact (10/13), revealing a partial `supersede` semantic ambiguity (Appendix I).

Cross-family judge validation. Two further judges from different families — DeepSeek-V3 and Claude Opus 4.7 — re-run admission on the same 100 cases (Appendix F.3). Agreement with the human majority varies sharply by family (Claude 99/100, Qwen 79/100, DeepSeek 58/100), and the disagreements land on the same semantic-abstraction categories in every case: single-LLM admission judging has a reproducible blind spot there, not a Qwen-specific quirk. Since that blind spot is conservative, the admitted set stays clean ($\kappa = 0.958$).

Circularity: what we can and cannot rule out. 253/385 cases are LLM-drafted, admitted by an LLM judge, and the hook is also an LLM — so shared inductive bias is a live threat. We cannot exclude it entirely, but three tests bound it, and each predicts the *opposite* of what we observe if circularity were driving the result. **(i)** On the 132 hand-crafted cases the deterministic floor *drops* (LangGraph 52.3 % vs. 68.4 % LLM-drafted) while the same backend with the hook *rises* (LangGraph+LLM 94.7 vs. 86.6 %): a $+42$-pt lift on human-written cases against $+18$ on LLM-written ones, one backend at both ends. Circularity predicts the reverse asymmetry. **(ii)** The hook's largest gains are in identifier_obfuscation, the category where LLM-drafted candidates were *rejected* almost entirely (0/24 admitted) and the admitted cases are predominantly hand-written. **(iii)** A 77-case external-authored subset (4 contributors at a separate institute, given only the category schema) independently replicates the canonicalization asymmetry on identifier_obfuscation (deterministic 0/8, LLM-hook 8/8 on both backends) with the pass band falling to 26–49 % (Appendix U).

4 Adapter Protocol

A six-method control-plane algebra: three mandatory recall-plane primitives (`reset`, `inscribe`, `recall_texts`) and three optional control-plane mutations (`supersede`, `release`, `purge`) with N/A scoring (a system's N/A pattern characterizes its *control-plane coverage*):

The optional methods raise `NotImplementedError` when a store lacks the primitive, which maps cleanly to N/A on the corresponding family and distinguishes *implemented and failed* from *not provided at all*. Full signatures are in Appendix G; the shipped adapters run 63 to 194 lines each.

The Protocol is a behavioural contract, not a syntactic one. A store exposing only `add` and `delete` can implement `supersede` as "`delete(best_match)` → `add(new)`" and passes as long as the behaviour holds; `release` and `purge` compose the same way (Appendix G). N/A is reserved for stores that *cannot* produce the behaviour under any composition of their API, so a system is never penalized for lacking a *name* it can supply by *behaviour*.

N/A makes coverage visible; it does not excuse it. Headline scores are *strict*: a store that cannot express a deletion has not forgotten anything, and dropping those cases would reward a system for lacking the capability under measurement. Where the two differ we report the evaluable score beside it — the same numerator over only the cases a store can attempt — so the gap is a number rather than something folded into a rate. Four of the five systems in Table 1 implement or compose all six methods and their two scores coincide; MemPalace exposes no content-addressed deletion primitive, so every one of its cases is N/A. The gap opens in the ecosystem evaluations (Appendix R), where A-MEM is 143/385 strict and 143/310 evaluable.

Reference implementation. To give the Adapter Protocol a concrete anchor we additionally ship one of the four primary memory stores benchmarked here as supplementary material ($\sim$700 lines of Python over SQLite + `sqlite-vec` (Garcia, 2024) + FTS5). It implements all six Protocol methods, exposes an optional `llm: Callable[[str], str]` hook for the three mutation-time prompts (SUPERSEDE PLANNER, PURGE MATCH, RELEASE MATCH; full text in Appendix D), and serves as one of the two backends we use in the cross-architecture LLM-hook ablation (Appendix Q). Detailed API surface and formal soft-delete invariants are in Appendix C.

Table 1: FORGETEVAL-Adv: 385 cases across 10 attack categories, cells given as pass/total (%). MEMPALACE scores 0/385 strict and has *no evaluable cases at all* — it exposes no content-addressed deletion primitive — and serves as the axis-flip reference point (§5.4).

Attack category	LETHE	MEM0	LangGraph	MEMPALACE	LETHE+LLM
substring_trap (n=36)	33/36 (92)	30/36 (83)	35/36 (97)	0/36 (0)	36/36 (100)
prefix_collision (n=39)	32/39 (82)	13/39 (33)	27/39 (69)	0/39 (0)	32/39 (82)
paraphrase_supersession (n=38)	31/38 (82)	31/38 (82)	31/38 (82)	0/38 (0)	31/38 (82)
negation_trap (n=40)	38/40 (95)	38/40 (95)	38/40 (95)	0/40 (0)	36/40 (90)
temporal_qualifier (n=37)	37/37 (100)	37/37 (100)	37/37 (100)	0/37 (0)	37/37 (100)
shared_attribute (n=40)	35/40 (88)	37/40 (92)	35/40 (88)	0/40 (0)	40/40 (100)
identifier_obfuscation (n=38)	2/38 (5)	23/38 (61)	2/38 (5)	0/38 (0)	35/38 (92)
cross_lingual_identifier (n=38)	1/38 (3)	7/38 (18)	1/38 (3)	0/38 (0)	32/38 (84)
recursive_supersession (n=39)	36/39 (92)	36/39 (92)	36/39 (92)	0/39 (0)	36/39 (92)
All 385 cases (strict)	**245/385 (63.6)**	**252/385 (65.5)**	**242/385 (62.9)**	**0/385 (0.0)**	**342/385 (88.8)**
95 % Wilson CI	[58.7, 68.3]	[60.6, 70.0]	[57.9, 67.5]	[0.0, 1.0]	[85.3, 91.6]
the 345 that measure forgetting	245/345 (71.0)	252/345 (73.0)	242/345 (70.1)	0/345 (0.0)	315/345 (91.3)
primitive-existence diagnostic: inside the 385-case total, outside the 345-case forgetting row					
compound_fact (n=40)	0/40 (0)	0/40 (0)	0/40 (0)	0/40 (0)	27/40 (68)

compound_fact tests for a *partial edit*: an operation that rewrites one clause of a stored row instead of replacing or removing the whole row. MEMPALACE has one only over knowledge-graph triples, not stored rows. MEM0 and LangGraph have no native one either, but both can compose the behaviour — LangGraph+LLM reaches 27/40 on this column by planning delete-old-then-add-merged over the same `InMemoryStore` (Appendix R) — so their zeros are planner decisions, not missing primitives. LETHE's `supersede` takes such a mode and it still scores 0/40, the same kind of zero: which is why the same store through the same primitive reaches 27/40 once a model sees the request. The category asks which primitives a store has rather than what it can be commanded to forget, so the placement claim below is stated on the 345-case row, where it is not counted.

5 EXPERIMENTS

5.1 SETUP

All four primary adapters use `all-MiniLM-L6-v2` (384-d, ONNX via fastembed (Qdrant team, 2024)) on a single CPU. No GPU, no API calls (except in the LLM-hook and MEM0 `infer=True` configurations), no internet on the recall hot path.

5.2 ADVERSARIAL LAYER (385 CASES)

The 1000-case template suite is saturated: the two lexical stores score 993/1000 and 995/1000, within 0.2 pt of each other, and MEM0 888/1000 (Appendix B). A suite on which the systems under test differ by less than a point cannot locate where they differ, which is what the adversarial layer is for.

Because every system is run on the same case list, all comparisons below are *paired*, and we test them with exact McNemar tests on the per-case verdicts rather than with the marginal Wilson intervals in Table 1 (which ignore the pairing). Full test output, including per-category tests with Holm correction, is released in `data/mcnemar_significance.json`.

Three observations. **(1) Deterministic systems are aggregate-indistinguishable:** the three land within 2.6 points (LETHE 63.6, MEM0 65.5, LangGraph 62.9), and LETHE's 0.7 pt over LangGraph's vanilla `InMemoryStore` is noise by the paired test — 7 discordant cases, 5 to 2, exact $p = 0.45$. *The reference implementation does not beat a batteries-included storage primitive in aggregate*; the differentiation is per-category and per-placement. That 63–66 % band is itself an average over mixed-provenance cases: on the 132 hand-crafted ones the two lexical stores fall $\sim$10 pt (53.0 and 52.3 %) while MEM0 barely moves (64.4 %; Appendix F.4). **(2) Per-category separation:** LETHE's 32/39 on prefix_collision against MEM0's 13/39 is 19 discordant cases, every one of them the same way ($p = 4 \times 10^{-6}$); MEM0's 7/38 on cross_lingual_identifier against LETHE's 1/38 is 6, all in the other direction ($p = 0.03$ raw, 0.25 under Holm across the ten categories — suggestive, not established), and its 23/38 against 2/38 on identifier_obfuscation is 21 ($p = 1 \times 10^{-6}$). The first and third survive Holm; the middle one does not. Lexical-precise purge avoids prefix-collision but cannot bridge script variation; vector-soft delete does the opposite. **(3) The hook lift is one-directional but narrow.** The same adapter goes from **245/385** to **342/385**, $+25.2$ **pt**, and the paired test over those 385 gives $p = 4 \times 10^{-27}$ with **99** of the 101 disagreeing cases going the same way. They are

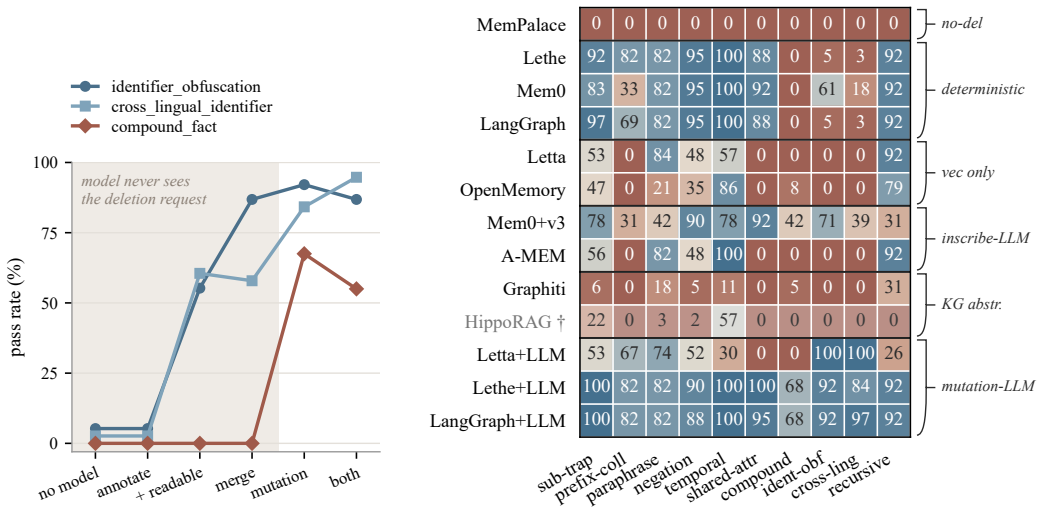


Figure 1: **Left:** the controlled ablation of Table 2, three categories against the six call sites, shaded where the model never sees the deletion request. **Right:** thirteen deployed configurations by category (Appendix J), every cell strict over its own category; HippoRAG (†) is the one row not re-measured.

not spread evenly: **91 of the 99** fall in identifier_obfuscation (+33), cross_lingual_id (+31) and compound_fact (+27), the same three that survive Holm correction, while four categories move by zero. This configuration was executed five times (sd 1.1 pt), and every number we report for it is the run at the mean of those five rather than the best of them; the per-run scores and the two places the same configuration appears under a different execution are in Appendices P and X. Regressions range from 0 to 2 across the five. compound_fact asks whether a partial-edit primitive exists rather than whether a store can be commanded to forget, so the claim is stated without it. On the 345 cases that measure forgetting the same adapter goes **245/345** to **315/345**, +20.3 **pt**, with **72** fixed against 2 broken and $p = 3 \times 10^{-19}$. Of those 72, **64 are the two canonicalization categories**. The same contract on LangGraph's `InMemoryStore`, which shares no code with ours, replicates it: 62.9→89.4 % (+26.5 pt, 107 fixed vs. 5 broken, $p = 5 \times 10^{-26}$), with 96 of the 107 in the *same* three categories (Appendix Q). The two backends land **0.5 pt** apart and concentrate their gains identically. Six models across five vendors and a 141× range in per-run cost all land between 88.6 and 91.4 % (Appendix P); the 11-case spread between them (2.9 pt) tracks neither vendor nor price but whether a model can follow the JSON contract at all — on an earlier revision of the suite Llama-3.1-70B, which cannot, got none of the lift.

The pattern is not an embedder artifact. Swapping the English MiniLM for `paraphrase-multilingual-MiniLM-L12-v2` leaves LETHE's cross-lingual score at 0/16 and moves MEM0's from 8/16 to 7/16: a multilingual *embedder* does not buy canonicalization. The architectural patterns in Table 1 are robust to embedder choice; the LLM's position in the pipeline, not the vector space, is the lever.

5.3 WHAT A NON-SYSTEM SCORES

The requirements below are not housekeeping; without them three deployed systems read as solved. A-MEM, OpenMemory (CaviraOSS Contributors, 2025) and Letta each score 97–100 % on both canonicalization categories when scoring asks whether a forbidden string still *ranks*, and **0/38** on both when it asks whether that string is still *reachable*. They get there by different routes — A-MEM extracts tags with a model on write, Letta and OpenMemory embed the raw passage and let the vector neighbourhood do the work at recall — but in none of the three does the deletion path consult it, so the fact is rewritten rather than removed. Two controls say this is the requirement discriminating and not a harsher scorer — Letta *with* a mutation-time hook holds both at 38/38 under the same scoring, and probing costs the deterministic store **zero cases**: 244/385 with it and without it (Table 3), while it costs *normalize*, which rewrites rather than deletes, 8.9 pt.

Table 2: Controlled placement ablation: one store, one model, one prompt family, and only the call site moves. The last two rows tie — one case apart overall, five on compound_fact, a category carrying 17 cases whose verdict is not unanimous across repeats of a single configuration (Appendix P).

model called at	sees	overall	ident_obf	cross_ling	compound
nowhere	—	63.6 %	2/38	1/38	0/40
write, annotating	data	63.9 %	2/38	1/38	0/40
+ control plane reads it	data	69.6 %	21/38	23/38	0/40
write, merging	data, may edit	74.8 %	33/38	22/38	0/40
mutation	**the request**	**88.8 %**	**35/38**	**32/38**	**27/40**
both	the request	88.6 %	33/38	36/38	22/40

Requirement (1) is the one authors omit. We can measure how naturally, because 77 of our cases were written by four external contributors given the category schema and nothing else (Appendix U). **All 16 of their canonicalization cases omit a positive requirement, and none of their other 61 do.** We expect this wherever a criterion is an absence: **a forgetting case needs a positive requirement, and what enforces one is a degenerate baseline, not review.** We therefore calibrate against three reference points that are not memory systems — *null*, *nuke* and *normalize* (five lines of Unicode hygiene, no model) — and an *oracle* deleting exactly the rows carrying a forbidden string. The oracle is a satisfiability control, not a ceiling: it deletes whole rows, so it fails all 40 compound_fact cases and a store with an edit primitive scores above it.

Running them is what tells you which requirements a forgetting suite has to enforce, and Table 3 (Appendix A) shows three that are easy to omit. **(1) Every case needs a positive requirement.** A case scored only on what must disappear is passed by deleting everything, so each of ours names an entity that must *survive* — a sibling of the same kind, present in the same surface forms as the target. Without this, *null* and *nuke* take 76/76 of the canonicalization cases; with it, 0/76. **(2) Absence must be checked under probing.** Scored against a single query's top-k, a store passes by *not ranking* a row it never deleted — 34 of *normalize*'s 242 passes are of that kind. Each forbidden string is therefore also queried directly, so forgetting means *unreachable* rather than unranked, the distinction Wang & Zhang (2026) draw for multimodal stores. **(3) The request must not contain its own answer.** A cross-lingual purge naming both surface forms of an entity can be satisfied by literal matching, testing nothing; ours name one. The rules and the gate each was held to are in Appendix H.

normalize stays in every table below: Unicode hygiene is a real, nearly free alternative to a model, and the LLM's margin *over* it is the honest form of the placement claim.

5.4 Cross-evaluation on Memora

Does ForgetEval measure a surface existing benchmarks already cover? We run three adapters on all 10 Uddin et al. (2026) personas at the weekly scale (150 questions over $\sim$1,580 sessions), scored deterministically against Memora's released evidence literals (Appendix N). Two findings say no. **Axis flip:** MemPalace scores 0/385 here — no forgetting primitive its retrieval surface can address — but **60/150** on recall-heavy Memora, *above* Lethe's 47/150, so no scalar captures both axes. **Ranking divergence:** Lethe and LangGraph sit within 1 pt on ForgetEval-Adv while LangGraph leads by 20 questions on Memora — the same selectivity trade-off that wins it recall and loses it adversarial deletion (Appendix K), not a retrieval weakness (Lethe scores R@5 = 93.8 % on LongMemEval-S, Appendix T).

5.5 Control-plane placement trade-off

Comparing systems cannot settle a claim about placement. A-MEM, Mem0's router and our hook differ in storage, retrieval, embedder and extraction prompt as well as in when the model runs, so a difference between them identifies "these systems fail differently", not "the placement causes it". We therefore run the ablation that does settle it, and Table 2 states its terms.

Two ingredients, and Table 2 separates them. A write-time model that only *annotates* changes nothing (63.9 % against 63.6), though it runs on every fact and files correct canonical forms: `ALICE@ACME.IO` is stored with `alice@acme.io` beside it and the score does not move. The store says why the obvious reading — that work done before the request cannot become capability — is

not the right one. The deterministic `purge` retrieves lexically, where the annotation helps, and then groups rows by NFKC-normalised *whole text*, where it cannot: annotating a row changes its whole text, so the one thing the annotation is for is the one thing the matcher never consults. The arm was measuring our `purge`, not the call site.

Let the same control plane match against the annotation — a change of roughly forty lines, no new information and no new authority — and the two categories the annotation is *about* go from 2/38 and 1/38 to **21/38 and 23/38**. So write-time knowledge does transfer, exactly as far as the deletion path can reach it. Give the model authority to merge rows instead, collapsing duplicates at write time so that whole-text grouping works again, and identifier_obfuscation reaches **33/38**.

compound_fact is the category where no placement on the write path helps at all. It is 0/40 in all three write-time configurations — annotation, readable annotation, and merge authority — and in the no-model baseline, and 27/40 and 22/40 in the two where the model sees the request. Deciding which *clause* of a row to remove is not a fact about the data, so no amount of preparation makes it reachable; it is a fact about what was asked. Adding annotation on top of the mutation-time hook buys nothing here, and the five-case difference between the two is not a finding either. Five is a net: the combined arm loses nine of the cases the mutation-time arm passes and picks up four it does not, and those nine are drawn from the ones that already flip across repeats of the mutation-time configuration by itself, which are more numerous on this category than on any other (Appendix P).

So capability needs the deletion path to reach the stored knowledge, and to see the request. The first is buyable early, and what it costs is engineering: make the matcher read what the extractor wrote. The second is not buyable early at any price, which is the asymmetry Table 2 exists to isolate.

The cross-lingual limit is a renamed entity, not a third surface form. The hook scores 32/38 on our cross_lingual_identifier cases and **0/8** on the eight external ones, which give an entity three surface forms rather than two. Three matched conditions, differing only in what the third form *is*, locate the cause: **8/8** on two forms, **7/8** when the third is a transliteration variant, and **0/8** when it is a historical exonym (*Peking* for *Beijing*), with the deterministic plane at 0/8 throughout and an oracle at 8/8 on all three. Form count costs one case; what costs eight is that the exonym shares no morphology with the name it replaced, so no rule over the request's surface form reaches it, and the identity is a fact about the world the contract never asks for.

A third-party store reproduces the ablation. The thirteen deployed configurations are profiled in Appendices J and R. One comparison among them is not observational. Letta scores **0/38 and 0/38** on the two canonicalization categories, and Letta with the same contract wired in at mutation time scores **38/38 and 38/38** — same store, same embedder, same suite, same scorer. Our ablation moves the same two categories from 2/38 to 35/38 and from 1/38 to 32/38 on a store we wrote; Letta moves both from 0/38 to 38/38 on one we did not.

The other rows explain why this was not already known: the three stores of §5.3 that read as solved all annotate at write time and none consults that work when deleting. Read *across* systems the right panel stays observational — its rows differ in storage and retrieval as well as call site, and canonicalization is not exclusive to mutation time, five rows clearing 30 % with two running no model there. compound_fact separates by call site, less sharply here than in the ablation: the controlled arms that cannot see the request are at 0/40 exactly, while deployed systems reach 42 % (MEM0 with `infer=True`, Appendix S). Letta+LLM sees the request, cannot express the edit, and stays at zero, which is why seeing it is necessary and not sufficient.

Cost of the hook. Only 99 of the 445 mutations need the model, and a three-signal rule over the request and retrieved rows captures **77.4 % of the lift for 31 % of the calls** without one, at a cost of three regressions. The hook adds about 1.0 s/case and leaves recall untouched (Appendices L and M).

What we would not claim from this. Four limits. compound_fact asks whether a partial-edit primitive exists, so the claim is stated without it: 245/345 to 315/345, +20.3 pt, which deleting the category outright leaves standing. Mutation-time placement buys surface-form canonicalization, not cross-lingual identity. The pass band reflects who wrote the cases, not a ceiling: 63.6 % here against 33.8 % externally, where the lift falls to +11.7 and +16.9 pt; and the substring scorer is not exact — 15 of 50 audited failures are its artifacts and an NLI re-score moves the lift by three cases in a hundred (Appendix I).

REPRODUCIBILITY STATEMENT

All experiments in this paper are reproducible from the released artifact. The four primary adapters (LETHE, MEM0, LangGraph `InMemoryStore`, MEMPALACE) run on a single CPU with no API key and no GPU; Appendix Y gives the seeds, the pinned third-party versions, and per-experiment wall-clock costs, and Appendix G gives the adapter sources. The FORGETEVAL case definitions, the six-method Adapter Protocol, the per-case verdicts behind every 385-case run (`data/*_ckpt.jsonl`), the LLM prompts (Appendix D), and the analysis scripts that regenerate every figure and the eighteen tables built from run data are released under MIT. Extended-ecosystem results (Letta, Graphiti, OpenMemory, HippoRAG, A-MEM) and the mutation-time-hook results additionally require external services or an OpenAI-compatible endpoint; the exact model identifiers, service versions, and container images used are in Table 14 and Appendix P.

ETHICS STATEMENT

Data. FORGETEVAL contains no personal data. All cases use fictional identifiers (Alice/Bob/Carol, Acme/Globex, addresses at `example.com`); the case-authoring brief given to external contributors explicitly forbids real names, institutions, phone numbers, and e-mail addresses, and admitted cases are screened for compliance.

Human contributors. Two groups of people contributed: 10 annotators who independently re-labeled a 100-case admission sample (§3.3), and external authors who wrote adversarial cases (Appendix U). Both groups authored or labeled synthetic test material; no personal data about the contributors was collected or analyzed, and no human-subjects data appears in the benchmark. The activity is therefore not human-subjects research and is exempt from IRB review. Contributors are unpaid volunteers who consented to MIT release of their cases and are acknowledged by preferred name or anonymously, at their choice.

Dual use. This work measures whether memory systems can be commanded to forget. Its findings identify configurations in which deletion requests are *not* honored, which is information an adversary could in principle use to select a leakier deployment. We judge the disclosure benefit to be decisively larger: the same finding is what lets a deployer subject to GDPR Article 17 (European Parliament and Council, 2016) choose a configuration that actually deletes, and every system we evaluate is already publicly available.

AI USE STATEMENT

Large language models are *objects of study* in this work — as mutation-time hooks (§5.5), inscribe-time extractors, data-admission judges (§3.3), and the NLI re-scorer (Appendix X). That usage is part of the experimental design and is documented with exact model identifiers, prompts (Appendix D), and per-case outputs in the released artifact.

Separately, the authors used LLM-based coding assistants during implementation (adapter scaffolding, experiment runners, plotting code) and for language editing of this manuscript. Assistant-generated code was reviewed and tested before any result it produced was used, and `scripts/verify_all.py` regenerates every table and figure from `data/` and fails if one differs from what is committed. Every number reported in this paper was produced by executing the released scripts against real memory systems; no experimental result was generated, estimated, or filled in by a language model. The authors take full responsibility for all content and claims in this paper.

REFERENCES

Saad Alqithami. Forgetful but faithful: A cognitive memory architecture and benchmark for privacy-aware generative agents, 2025. URL `https://arxiv.org/abs/2512.12856`.

Michael C. Anderson, Robert A. Bjork, and Elizabeth L. Bjork. Remembering can cause forgetting: Retrieval dynamics in long-term memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20(5):1063–1087, 1994.

Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. Ed25519: high-speed high-security signatures. `https://ed25519.cr.yp.to/`, 2011. Now RFC 8032.

Robert A. Bjork. Theoretical implications of directed forgetting. In Arthur W. Melton and Edwin Martin (eds.), *Coding Processes in Human Memory*, pp. 217–235. Winston, Washington, DC, 1972.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *IEEE Symposium on Security and Privacy*, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *IEEE Symposium on Security and Privacy*, 2015.

CaviraOSS Contributors. OpenMemory: Self-hosted long-term AI memory engine. `https://github.com/CaviraOSS/OpenMemory`, 2025.

Prateek Chhikara et al. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025. URL `https://arxiv.org/abs/2504.19413`. Also appearing at ECAI 2025.

Cognee contributors. Cognee: Memory layer for ai agents. `https://github.com/topoteretes/cognee`, 2024.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *SIGIR*, pp. 758–759, 2009.

Tianyu Ding, Aditya Nannapaneni, Bingfan Liu, and Ling Zhang. Always-on agents: A survey of persistent memory, state, and governance in LLM agents, 2026.

Hermann Ebbinghaus. *Über das Gedächtnis: Untersuchungen zur experimentellen Psychologie*. Duncker & Humblot, 1885.

European Parliament and Council. Regulation (eu) 2016/679 of the european parliament and of the council (general data protection regulation). Official Journal of the European Union, L 119, 1–88, 2016. See in particular Article 17, "Right to erasure".

Alex Garcia. sqlite-vec: A vector search sqlite extension. `https://github.com/asg017/sqlite-vec`, 2024.

Ruxi Gu, Zhenliang Zhang, and Wei Wang. ForgetBench: Benchmarking forgetting dynamics of long-term parametric memory in language models, 2026a.

Yingjie Gu, Wenjian Xiong, Liqiang Wang, Pengcheng Ren, Chao Li, Xiaojing Zhang, Yijuan Guo, Qi Sun, Jingyao Ma, and Shidang Shi. FSFM: A biologically-inspired framework for selective forgetting of agent memory, 2026b. URL `https://arxiv.org/abs/2604.20300`.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL `https://arxiv.org/abs/2405.14831`.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks, 2026. URL `https://arxiv.org/abs/2602.16313`.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *International Conference on Learning Representations (ICLR)*, 2026. URL `https://arxiv.org/abs/2507.05257`.

Zhuoran Jin, Pengfei Cao, Chenhao Wang, Zhitao He, Hongbang Yuan, Jiachun Li, Yubo Chen, Kang Liu, and Jun Zhao. RWKU: Benchmarking real-world knowledge unlearning for large language models. In *NeurIPS*, 2024. URL `https://arxiv.org/abs/2406.10890`.

Letta contributors. Letta: Stateful agents framework (successor to memgpt). https://github.com/letta-ai/letta, 2024.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. TOFU: A task of fictitious unlearning for LLMs. *arXiv preprint arXiv:2401.06121*, 2024. URL https://arxiv.org/abs/2401.06121.

MemPalace contributors. Mempalace: An open-source ai memory system. https://github.com/mempalace/mempalace, 2026.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *NeurIPS*, 2022.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *ICLR*, 2023. MEMIT: batch knowledge editing in LLM weights.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. In *EACL*, 2023.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Qdrant team. fastembed: Fast, accurate, lightweight embeddings via onnx runtime. https://github.com/qdrant/fastembed, 2024.

Preston Rasmussen et al. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025. URL https://arxiv.org/abs/2501.13956.

Vikas Reddy and Sumanth Reddy Challaram. Reliable post-retrieval assembly for agent memory: Separating evidence extraction from policy execution, 2026. URL https://arxiv.org/abs/2606.01435.

Zhe Ren, Yibo Yang, Yimeng Chen, Zijun Zhao, Benshuo Fu, Zhihao Shu, Bingjie Zhang, Yangyang Xu, Dandan Guo, and Shuicheng Yan. GateMem: Benchmarking memory governance in multi-principal shared-memory agents, 2026.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2025. URL https://aclanthology.org/2025.findings-acl.989/.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS Datasets and Benchmarks*, 2021. arXiv:2104.08663. 18 zero-shot IR datasets.

Bozhong Tian, Xiaozhuan Liang, Siyuan Cheng, Qingbin Liu, Mengru Wang, Dianbo Sui, Xi Chen, Huajun Chen, and Ningyu Zhang. To forget or not? towards practical knowledge unlearning for large language models. In *Findings of the Association for Computational Linguistics (EMNLP Findings)*, 2024. URL https://arxiv.org/abs/2407.01920.

Md Nayem Uddin, Kumar Shubham, Eduardo Blanco, Chitta Baral, and Gengyu Wang. From recall to forgetting: Benchmarking long-term memory for personalized agents. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2026. URL https://arxiv.org/abs/2604.20006.

Bin Wang, Fan Wang, Pingping Wang, Jinyu Cong, Yang Yu, Yilong Yin, Zhongyi Han, and Benzheng Wei. Agentic unlearning: When LLM agent meets machine unlearning, 2026a. URL https://arxiv.org/abs/2602.17692.

Kuan Wang and Chao Zhang. MemLeak: Diagnosing information leaks in multimodal agent memory, 2026.

Yuyao Wang, Zhongjian Zhang, Mo Chi, Kaichi Yu, Yuhan Li, Miao Peng, Bing Tong, Chen Zhang, Yan Zhou, and Jia Li. EvoMemBench: Benchmarking agent memory from a self-evolving perspective, 2026b. URL https://arxiv.org/abs/2605.18421.

Ziming Wang. TOKI: A bitemporal operator algebra for contradiction resolution in LLM-agent persistent memory, 2026.

Di Wu et al. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *ICLR*, 2025. URL https://arxiv.org/abs/2410.10813.

Menglin Xia, Xuchao Zhang, Shantanu Dixit, Paramaguru Harimurugan, Rujia Wang, Victor Ruhle, Robert Sim, Chetan Bansal, and Saravan Rajmohan. Memora: A harmonic memory representation balancing abstraction and specificity. *arXiv preprint arXiv:2602.03315*, 2026. URL https://arxiv.org/abs/2602.03315.

Wujiang Xu et al. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025. URL https://arxiv.org/abs/2502.12110.

Zep AI. Graphiti: Temporal knowledge graphs for memory. https://github.com/getzep/graphiti, 2025. Open-source successor to the deprecated Zep CE.

Dawen Zhang, Pamela Finckenberg-Broman, Thong Hoang, Shidong Pan, Zhenchang Xing, Mark Staples, and Xiwei Xu. Right to be forgotten in the era of large language models: Implications, challenges, and solutions. *arXiv preprint arXiv:2307.03941*, 2023. URL https://arxiv.org/abs/2307.03941.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, Yuandong Tian, and Jishen Zhao. AMA-Bench: Evaluating long-horizon memory for agentic applications, 2026. URL https://arxiv.org/abs/2602.22769.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *EMNLP*, 2023.

A Calibration against non-systems

The four reference points of §5.3 under each combination of requirements.

B Template suite results

The flat-difficulty suite of §5, at seed=42 with four distractors per case.

C Reference implementation API surface

The reference implementation we ship as supplementary material stores each memory row with a single scalar $\mathrm{depth} \in \mathbb{R} \cup \{\mathsf{void}\}$ that collapses its forgetting state into one column ($+\infty$ pinned; 1.0 surface; $(0, 1)$ sinking; 0 submerged but logged; void erased); every state transition writes one row to an append-only event table, enabling time-travel queries and Ed25519-signed purge receipts (Bernstein et al., 2011). Compound-fact cases use an `edit` primitive that updates row text and re-indexes the vector + FTS5 entry without changing depth; the LangGraph cross-architecture ablation (Appendix Q) demonstrates the same effect via `delete-old + add-merged` on a backend without a native edit primitive, so this is an ergonomic convenience rather than a uniquely necessary architectural feature. Soft-delete invariants (one-way purge, event-log determinism, monotone decay) follow from textbook SQL semantics on the single-scalar + audit-log schema; we omit the proofs from this paper and ship them with the supplementary source.

Table 3: Pass rate (%) of four reference points, none of which performs selective forgetting, under successively tighter requirements. Without all three, a returns-nothing adapter takes 100 % of both canonicalization categories and a delete-everything adapter reaches 60.0 %, within three points of the band real systems occupy. **The oracle is invariant across all four columns**, so each requirement raises the floor without raising the ceiling. Both measured systems are unchanged by the survivor requirement — 244 to 244 and 348 to 348 — so all of their movement is the one-form purge request: a case back for the deterministic store, six off the hooked one — though those are separate executions, and six is inside that configuration's spread (sd 4.2, Appendix P). The hooked + probing cell would cost a sixth 445-call run; the deterministic one is free, so it is measured.

Reference point	**none**	**+ survivor**	**+ probing**	**+ one-form request**
null (returns nothing)	24.9	5.2	5.2	5.2
nuke (deletes everything)	60.0	40.3	40.3	40.3
normalize (5 lines, no LLM)	62.3	62.9	54.0	48.6
oracle (deletes exactly the target)	85.5	85.5	85.5	85.5
the two canonicalization categories only (% of 76)				
null (returns nothing)	100	0	0	0
nuke (deletes everything)	100	0	0	0
normalize (5 lines, no LLM)	93	96	59	32
oracle (deletes exactly the target)	100	100	100	100
measured systems, same adapter, identical hook				
deterministic (no hook)	63.4	63.4	63.4	63.6
+ mutation-time hook	90.4	90.4	—	88.8

Table 4: Template suite, seed=42, distractors=4. 95 % Wilson intervals in brackets. LETHE and LangGraph `InMemoryStore` saturate to within 0.2 pt of each other; MEM0 collapses on amnesia and purge. This suite ships per-family totals (`data/full_1000.json`), not per-case verdicts.

System	**Pass**	**Wilson 95%**
LETHE v1	993/1000 (99.30%)	[98.56, 99.66]
LangGraph `InMemoryStore`	995/1000 (99.50%)	[98.83, 99.79]
MEM0 v2.0.18	888/1000 (88.80%)	[86.69, 90.61]
MEMPALACE	0/1000 (0.00%)	[0.00, 0.38]

D LLM PROMPTS

The three prompts the mutation-time hook issues, verbatim. Each returns JSON; a response that does not parse falls through to the deterministic path, which is what makes a model that cannot follow the contract score at the deterministic baseline rather than below it. Temperature is 0 throughout.

`supersede`. Constant `LLM_PROMPT_SUPERSEDE` in `bench/forgeteval/adapter.py`.

```
\
You are deciding how to apply a memory supersession.

EXISTING_MEMORY: {old_text}
SUPERSEDE_QUERY: {query}
NEW_FACT: {new_text}

**Default is ATOMIC** -- replace the whole memory with NEW_FACT.
Choose ATOMIC when EXISTING_MEMORY's topic matches SUPERSEDE_QUERY
(even if paraphrased, dated, or recursive).

Choose PARTIAL when EXISTING_MEMORY combines two **distinct-topic**
facts (different attributes about the same subject, like
location-vs-employer or marital-status-vs-employer), and
SUPERSEDE_QUERY targets only one attribute. In partial mode,
preserve the unaffected attribute and splice NEW_FACT into the
addressed one.

Examples
--------

EXISTING_MEMORY: User lives in Berlin and works at Stripe as a backend engineer.
```

```
SUPERSEDE_QUERY: user city of residence
NEW_FACT: User relocated to Madrid and continues working remotely.
-> {{"mode": "partial", "merged_text": "User relocated to Madrid and works at Stripe as a backend engineer."}}

EXISTING_MEMORY: User is married to Jamie and works at Google.
SUPERSEDE_QUERY: user employer Google
NEW_FACT: User joined Anthropic as a research engineer.
-> {{"mode": "partial", "merged_text": "User is married to Jamie and joined Anthropic as a research engineer."}}

EXISTING_MEMORY: User does NOT work at Anthropic and has never interviewed there.
SUPERSEDE_QUERY: user Anthropic employment status
NEW_FACT: User actually joined Anthropic last quarter.
-> {{"mode": "atomic"}} (both clauses are about Anthropic-employment -- atomic supersedes them together)

EXISTING_MEMORY: User joined Google in 2020.
SUPERSEDE_QUERY: user current employer
NEW_FACT: User joined Meta in 2022.
-> {{"mode": "atomic"}} (single-topic supersession)

Format
------
Reply with exactly one JSON object and nothing else:
  {{"mode": "atomic"}}
or
  {{"mode": "partial", "merged_text": "<the merged sentence>"}}

Do not add facts not present in either source.
```

purge_match. Constant LLM_PROMPT_PURGE_MATCH in bench/forgeteval/adapter.py.

```
\
You are grouping memory items that describe the same underlying
identifier. Surface-form variations (case, whitespace, quoting,
leading @, optional separators in phone numbers / UUIDs / SSNs /
credit cards) all count as the same identifier.

TARGET_IDENTIFIER: {target}

CANDIDATES (one per line, indexed from 0):
{candidates}

Return exactly one JSON object listing the candidate indices whose
text describes the SAME identifier as the target. Distinct
identifiers that happen to share a prefix (e.g. alice@acme.io vs
alice.smith@acme.io, 12345 vs 123456) MUST NOT be grouped.

  {{"matching_indices": [<int>, ...]}}
```

release_match. Constant LLM_PROMPT_RELEASE_MATCH in bench/forgeteval/adapter.py.

```
\
You are deciding which memory items should be released (soft-deleted)
based on a natural-language release request.

RELEASE_REQUEST: {request}

CANDIDATES (one per line, indexed from 0):
{candidates}

Return exactly one JSON object listing the candidate indices whose
content should be released according to the request. Include a
candidate if and only if it (a) describes the entity or topic the
request targets, OR (b) mentions that entity even as part of a
compound statement. Do NOT include candidates that only share an
attribute (e.g. same city, same job) with the target -- those are
sibling facts, not target facts.

  {{"matching_indices": [<int>, ...]}}
```

E WORKED CASE EXAMPLES

One case per attack category, verbatim from the released suite, with the verdict each of the four Adapter Protocol systems returns on it. ✓ means the survivor was still reachable and every forbidden

string was not, under the probe scoring of §5.3. The pattern to read is not the aggregate but which *kind* of request the deterministic plane loses: it holds on the lexically unambiguous ones and fails where the request names a form the stored row does not carry. The cross_lingual_identifier example is one of the eight Latin-script cases in that category, since the others use scripts this template cannot typeset; all 38 are in the release. Generated by `scripts/worked_cases.py`.

substring_trap — `adv_substring_trap_01`.

- User currently lives in Vienna.
- The office cafeteria serves Vienna sausage on Tuesdays.
- Hallway plants were repotted last week.
- Conference room reservation system is offline.

Request: `supersede("user residence city", "User moved to Oslo last month.")`. Query: "Where does the user live?".
Must survive: `Oslo`. Must be unreachable: `lives in Vienna`, `currently lives`.

LETHE	MEM0	LangGraph	LETHE+LLM
✓	✓	✓	✓

prefix_collision — `adv_prefix_collision_01`.

- Customer alice@acme.io subscribed in 2024.
- Customer alice.smith@acme.io subscribed in 2023.
- Quarterly all-hands was rescheduled.

Request: `purge("alice@acme.io")`. Query: "Which alice customers are still active?".
Must survive: `alice.smith@acme.io`. Must be unreachable: `alice@acme.io subscribed`.

LETHE	MEM0	LangGraph	LETHE+LLM
✓	×	✓	✓

paraphrase_supersession — `adv_paraphrase_supersession_01`.

- User works at Stripe as a senior backend engineer.
- User commutes by bike to the SF office.

Request: `supersede("user employer", "User quit the payments industry and joined an AI safety lab in San Francisco.")`. Query: "What does the user do for work now?".
Must survive: `AI safety`, `lab`. Must be unreachable: `Stripe`, `backend engineer`.

LETHE	MEM0	LangGraph	LETHE+LLM
✓	✓	✓	✓

negation_trap — `adv_negation_trap_01`.

- User does NOT work at Anthropic and has never interviewed there.
- User's coffee preference is espresso.

Request: `supersede("user Anthropic employment status", "User actually joined Anthropic last quarter.")`. Query: "Does the user work at Anthropic?".
Must survive: `joined Anthropic`. Must be unreachable: `does NOT work at Anthropic`, `never interviewed`.

LETHE	MEM0	LangGraph	LETHE+LLM
✓	✓	✓	✓

temporal_qualifier — `adv_temporal_qualifier_01`.

- User joined Google in March 2020 as a software engineer.
- User's hobby is competitive chess on weekends.

Request: `supersede("user current employer Google", "User moved to Meta in June 2022.")`. Query: "Where does the user currently work?".
Must survive: `Meta`, `June 2022`. Must be unreachable: `joined Google in March 2020`.

LETHE	MEM0	LangGraph	LETHE+LLM
✓	✓	✓	✓

shared_attribute — `adv_shared_attribute_01`.

- Dana lives in Berlin and works at Stripe.
- Eve lives in Berlin and works at Notion.
- Frank lives in Tokyo and works at Spotify.

Request: `release("everything about Dana")`. Query: "Who lives in Berlin?".
Must survive: `Eve`, `Berlin`. Must be unreachable: `Dana lives in Berlin`.

LETHE	MEM0	LangGraph	LETHE+LLM
✓	✓	✓	✓

compound_fact — `adv_compound_fact_01`.

- User lives in Berlin and works at Stripe as a backend engineer.
- User uses an ergonomic mechanical keyboard.

Request: `supersede("user city of residence", "User relocated to Madrid and continues working remotely.")`. Query: "Where does the user work?".
Must survive: `Stripe`. Must be unreachable: `lives in Berlin`.

LETHE	MEM0	LangGraph	LETHE+LLM
×	×	×	✓

identifier_obfuscation — `adv_identifier_obfuscation_01`.

- Customer ALICE@ACME.IO subscribed to Plan A.
- Customer alice@acme.io is the same person (case-normalized).
- Office lunches are catered on Wednesdays.
- Record for morgan.reyes@acme.io is active on the account.
- Same record, alternate form MORGAN.REYES@ACME.IO, was verified.

Request: `purge("alice@acme.io")`. Query: "List every record on file, including morgan.reyes@acme.io.".
Must survive: `morgan.reyes@acme.io`. Must be unreachable: `ALICE@ACME.IO subscribed`, `alice@acme.io is the same`.

LETHE	MEM0	LangGraph	LETHE+LLM
×	✓	×	✓

cross_lingual_identifier — `adv_cross_lingual_identifier_07`.

- Customer José García placed order #42.
- Same customer as Jose Garcia (no accents) has Plan B.
- Office supply order was approved.

- Customer Renée Dubois opened a separate account in April.
- The same customer, written Renee Dubois, is on the Premium tier.

Request: `purge("Jose Garcia")`. Query: "List every customer on file, including Renee Dubois.". Must survive: `Renee Dubois`. Must be unreachable: `José García placed`, `Jose Garcia (no accents)`.

LETHE	MEM0	LangGraph	LETHE+LLM
×	×	×	✓

recursive_supersession — `adv_recursive_supersession_01`.

- User has been using Chrome as primary browser since 2020.

Request: `supersede("user primary browser Chrome", "User switched to Brave for privacy reasons in 2024.")`, `supersede("user primary browser Brave", "User switched back to Chrome in 2025 after Brave broke an extension.")`. Query: "What is the user's current browser?".
Must survive: `Chrome`. Must be unreachable: `switched to Brave`.

LETHE	MEM0	LangGraph	LETHE+LLM
✓	✓	✓	✓

F IS THE BENCHMARK TRUSTWORTHY?

Four checks, each answering a different way the suite could be circular or self-serving: how the cases partition against the two systems we develop, whether the admission judge over-rejects, whether it is an artefact of one model family, and whether the hand-crafted core behaves like the LLM-drafted complement.

F.1 POST-HOC LABEL PARTITION (§5 STAGE-3)

The Stage-3 labels assigned by running each admitted case through LETHE (deterministic) and LETHE+LLM (DeepSeek-V4-Flash) are **post-hoc analytical partitions** of the bench against the two systems we develop, not an independent difficulty annotation. By construction LETHE passes every easy case and fails every llm_lift one — a definitional re-statement, not a measurement. The labels were assigned by an earlier execution of these two configurations, so re-running moves exactly two cases across that boundary: the deterministic store now passes one llm_lift case and the hook passes one unsolvable case. We report the partition only to make two non-tautological points transparent: (i) how many cases are out of reach of either LETHE variant (unsolvable); and (ii) how a third-party system (MEM0) fares on cuts defined by our reference adapter. The 132 hand-crafted core cases carry the synthetic label manual and are not partitioned by the dual-system check.

Table 5: Population distribution by Stage-3 label, with per-system pass rates on each partition, recomputed from the designated runs so the column totals are the headline numbers of Table 1. llm_lift cases pass LETHE+LLM and fail LETHE in the labelling run; unsolvable cases fail both LETHE variants in it. Re-running moves one case out of each of those two definitions. The *manual* partition (hand-crafted core) is not labeled by the dual-system check. Generated by `scripts/partition_table.py`.

Label	N	LETHE	MEM0	LETHE+LLM
manual (core)	132	70 (53%)	85 (64%)	123 (93%)
easy	174	174 (100%)	161 (93%)	173 (99%)
llm_lift	55	1 (2%)	6 (11%)	45 (82%)
unsolvable	24	0 (0%)	0 (0%)	1 (4%)
Total	385	245 (64%)	252 (65%)	342 (89%)

Two observations follow from the partition. **First, labels predict outcome:** LETHE passes exactly the easy (and manual subset it passes by hand-crafted design) and fails llm_lift + unsolvable by

construction. **Second, unsolvable is a small (6.2 %) reservation:** 24 cases the deterministic store and MEM0 both fail outright and the hook clears once, surfacing residual failure modes worth future work — combinations of compound_fact + paraphrase that defeat even the LLM hook.

F.2 JUDGE AUDIT ON HAND-CRAFTED CORE

The Qwen-2.5-72B admission judge admitted 96 of the 112 hand-crafted core cases (85.7%) and rejected 16. The audit predates the 20 identifier_obfuscation cases added later, which is why its denominator is 112 and not the 132 of §3.3. Manual review of every rejection found *zero genuine bench bugs*: every rejected case is well-formed, falling into three characterizable failure modes. We list each by mode below.

Mode (A): Semantic equivalence (11 cases). The judge applies literal substring matching to mutation arguments and concludes that the surviving non-targeted surface form should have been preserved — but the category exists specifically to test whether the system canonicalizes surface forms. This mode hits two categories almost exclusively: identifier_obfuscation (cases 02, 08, 09, 12, 14, 16; case-, separator-, plus-tag-, and prefix-form variants) and cross_lingual_identifier (cases 03, 06, 07, 12, 13; Korean↔romanized, Chinese↔English, Spanish accented↔unaccented, Hebrew↔transliteration).

Mode (B): Multi-row mutation scope (1 case). `adv_shared_attribute_04` uses `release('everything about Hannah on the team')`, which by design fans out to every row mentioning Hannah; the judge interprets release as targeting one best-matching row and flags the substring leak that disappears once the release semantics are honored.

Mode (C): Judge computational error (4 cases). Four cases trip the judge on its own literal-substring rule:

- `adv_substring_trap_02`: judge claims the purged row's content remains part of the substring check; under the protocol it does not (the row is removed).
- `adv_prefix_collision_07`: judge claims `'case_id 12345 was filed'` is a substring of `'case_id 123456 was filed'` — arithmetically false (the digit '6' between '12345' and ' was' breaks the substring match).
- `adv_prefix_collision_16`: judge mis-identifies which row the `purge('phone +1-555-0100 office line')` mutation targets.
- `adv_recursive_supersession_02`: judge confuses intermediate state of a 3-step supersede chain with the final state.

The full per-case data (judge verdict, judge reason, reviewer verdict, failure-mode tag, one-line note) is released in `data/manual_review_v04.json` in the supplementary materials.

Two methodological conclusions follow. First, the independent LLM-judge admission protocol is *useful* but not *sufficient*: on the 7 mechanically-decidable categories (substring_trap, prefix_collision, paraphrase, negation, temporal, shared_attribute, recursive_supersession) judge precision is 92–100%; on the 3 semantic-abstraction categories (identifier_obfuscation, cross_lingual_identifier, compound_fact) precision drops to roughly 65–75%, dominated by mode (A) failures. Second, single-judge designs are not the endpoint — a category-aware admission protocol or an ensemble of category-specific judges would close the mode (A) gap. We leave this for future work and report the audit as a positive finding rather than a hidden limitation.

F.3 CROSS-FAMILY JUDGE VALIDATION

Admission by a single LLM family is a circularity risk in its own right, so we re-judge the same 100-case IAA-stratified sample (§3.3) under three LLM judges from three different model families and compare to the 10-annotator human-majority label:

Three readings. (1) Anthropic-family judge matches humans nearly perfectly (99/100, one disagreement on `adv_substring_trap_29` where Claude marks ill / humans wf). This is the strongest LLM–human agreement we measured and validates the admission protocol is reproducible across

Table 6: Three-judge cross-family agreement on the 100-case IAA sample. Human-maj = majority vote across 10 NLP/CS-trained annotators (Fleiss' $\kappa = 0.958$). Claude aligns with human majority on 99/100.

Judge	vs. Human	vs. Qwen	vs. DS-V3	WF/Ill
Claude 4.7 (Anthropic)	**99**	80	59	88/12
Qwen-2.5-72B (Alibaba)	79	—	73	92/8
DeepSeek-V3 (DeepSeek)	58	73	—	71/29
Human-majority (10 annotators)	—	79	58	87/13

Cells are agreement counts out of 100 cases.

Table 7: Pass rate on the hand-crafted core (HC, 132 cases) vs. the LLM-drafted complement (253 cases), for all twelve measured configurations. Δ = HC rate − LLM-drafted rate (negative = HC is harder for this system). This split uses the strict denominator throughout — a case whose primitive a store does not expose counts as a failure — so every denominator is 132/253 and each row's two cells sum to that system's strict total.

System	hand-crafted	LLM-drafted	Δ
MEMPALACE	0/132 (0.0)	0/253 (0.0)	+0.0
LETHE	70/132 (53.0)	175/253 (69.2)	−16.1
MEM0	85/132 (64.4)	167/253 (66.0)	−1.6
LangGraph	69/132 (52.3)	173/253 (68.4)	−16.1
LETHE+LLM	123/132 (93.2)	219/253 (86.6)	+6.6
LangGraph+LLM	125/132 (94.7)	219/253 (86.6)	+8.1
MEM0+v3	87/132 (65.9)	142/253 (56.1)	+9.8
A-MEM	34/132 (25.8)	109/253 (43.1)	−17.3
Letta	32/132 (24.2)	95/253 (37.5)	−13.3
Letta+LLM	88/132 (66.7)	103/253 (40.7)	+26.0
OpenMemory	27/132 (20.5)	78/253 (30.8)	−10.4
Graphiti	9/132 (6.8)	20/253 (7.9)	−1.1

families. **(2) Per-judge WF rate brackets the human rate**: Qwen 92/8 (over-permissive), Human 87/13, Claude 88/12, DeepSeek 71/29 (most strict). The Claude WF rate is within 1 case of the human-majority rate; the original Qwen admission rate is 5 cases higher (Qwen accepted 5 cases that humans and Claude reject, all in compound_fact). **(3) Of Qwen's 21 human-disagreements, 10 are compound_fact** cases where Qwen accepted but Claude+humans rejected. This is the partial-supersede semantic ambiguity flagged in Appendix I: when a supersede replaces an entire row, the non-superseded half of a compound fact is destroyed — humans and Claude (when tracing literally) reject; Qwen (when reasoning semantically) accepts. Per-case verdicts and the Python script are in the release at `iaa/third_judge_claude_summary.json` and `scripts/third_judge_agreement.py`.

Three judges, 55/100 unanimous. Claude, Qwen, and DeepSeek all label 55/100 cases identically (all WF; zero cases are unanimously ill across the three LLMs and the human majority). No case has all three LLMs disagreeing with humans. The 45 cases with at least one judge–judge disagreement cluster on compound_fact, prefix_collision, and paraphrase_supersession — exactly the semantic-abstraction categories that drive the 21 Qwen–human disagreements, so the disagreement pattern is a known LLM systematic limitation rather than a content disagreement about specific cases.

F.4 HAND-CRAFTED VS. LLM-DRAFTED SUBSET BREAKDOWN

FORGETEVAL-Adv is 132 hand-crafted core cases (authored by the research team, no LLM involvement) plus 253 LLM-drafted oracle-validated cases. We split the per-case verdicts from §5 along this provenance line to test whether the headline patterns (deterministic pass band, mutation-time-hook lift, inscribe-vs-mutation placement asymmetry) are properties of the LLM-drafted complement or hold on the hand-crafted core alone.

Three readings. (i) Hand-crafted is harder for deterministic stores. The ∼16-pt gap (LETHE 53.0 / LangGraph 52.3 % HC against 69.2 / 68.4 % LLM-drafted) means the LLM-drafted complement, despite oracle admission via Qwen judge, ended up systematically easier than what the human authors wrote. We report this as a property of the bench rather than fix it post-hoc: rewriting the LLM-drafted cases to harder targets would re-introduce the very circularity this analysis addresses.

(ii) The sign of Δ tracks where the deletion decision is made. Every configuration that decides deletion by deterministic matching finds the hand-crafted core harder (Δ from -1.1 to -17.3, seven rows, including A-MEM and Graphiti, which both run a model at write time but not on the deletion path); every configuration that routes the decision through a model finds it easier ($+6.6$ to $+26.0$, four rows, including MEM0+v3, whose model sits at write time but rewrites the rows a deletion would later have to match). The twelfth row, MEMPALACE, passes nothing on either subset, so its Δ is 0 by construction; among the eleven that score at all there is no exception in either direction. A shared-inductive-bias account — the LLM-drafted cases are easy for LLM-using systems because they share a drafter's habits — predicts exactly the opposite sign, and the two mutation-time hooks also beat their own full-suite headlines on the hand-crafted core (93.2 against 88.8, 94.7 against 89.4).

(iii) The placement asymmetry survives on hand-crafted cases alone. Restricting the controlled ablation of §5.5 to the 132 hand-crafted cases: no model **70/132**, annotation **69/132**, readable annotation **101/132**, merge authority **111/132**, mutation time **123/132**. Annotation alone moves the hand-crafted core by a single case, and in the wrong direction — the same nothing it is worth on the full suite — while the ordering of the remaining arms is unchanged. The asymmetry is therefore not an artifact of the LLM-drafted complement.

G ADAPTER SOURCES

The six-method contract in full:

```
class Adapter(Protocol):
    name: str
    def reset(self) -> None: ...
    def inscribe(self, text: str) -> int|str: ...
    def recall_texts(self, q: str, k: int) -> list[str]: ...
    # optional -- NotImplementedError -> N/A
    def supersede(self, old_q: str, new: str) -> None: ...
    def release(self, q: str) -> int: ...
    def purge(self, q: str) -> int: ...
```

Reference Python source for all four primary adapters (63–194 lines each) is in the released artifact.

H CASE REQUIREMENTS AND THE GATE EACH WAS HELD TO

Each requirement in §5.3 is one rule applied to every affected case. Nothing was authored per case and nothing was chosen after seeing a system's output, so no rule can have been fitted to a result. We state them as a specification rather than as a history because they are what another forgetting suite would have to enforce, whatever its cases look like.

Requirement 1: every case names a survivor. For identifier_obfuscation, the sibling is derived from the target by identifier type – 24 rules keyed on e-mail, card, IP, URL, handle, person, project, date, account, SSN, phone, ZIP, token, time, ISBN, path, colour, SKU, licence, ticket, VIN, BIC, snake handle, plain id – and all 38 cases matched a rule. For cross_lingual_identifier, no rule over the target string can produce a plausible sibling (a sibling for a Han name has to be a different Han name, not a transformation of that one), so the siblings are one authored pair per script family, applied uniformly to every case in that family. The external subset needed two further pairs, one for hyphenated codenames and one for Latin-script place names, on the same discipline.

Requirement 2: the request names one surface form. Every whitespace token containing a non-ASCII letter is dropped from the purge query, then dangling connectors are trimmed. One rule, 38 of 38 cases.

Requirement 3: forbidden content is checked for reach. Each forbidden string is checked against the union of the answer sets for the final query and one probe per forbidden string, the probe being that string itself. No store introspection and no judge.

The gate. A rule that only makes a suite harder is not a requirement, it is a difficulty knob. Each was required to drive *null* and *nuke* to zero on the affected categories while leaving an oracle that deletes exactly the forbidden rows still passing them, and to cost a store that genuinely deletes nothing:

the deterministic reference scores 244/385 with and without probing. The oracle is invariant at 85.5 % across all four columns of Table 3, so the suite became harder without becoming unsatisfiable. `scripts/validate_repair.py` and `scripts/repair_cross_lingual_queries.py` run the gate.

I EXTENDED LIMITATIONS

Four caveats bear on the tables, the same four §5.5 names. A 50-case audit finds 30 % of the reference store's deterministic failures are substring-scorer artifacts, confined to prefix_collision and clause-level supersession (Appendix X). compound_fact tests for a partial-edit primitive rather than for forgetting, so the placement claim is stated on the 345 cases that exclude it. The hand-crafted layer is English and cross_lingual_identifier samples nine scripts unevenly (Latin 12, Greek 7, Han 5, down to one case each for Devanagari and Thai), so per-script figures are well-powered only on the first three, and what mutation-time placement buys is surface-form canonicalization rather than cross-lingual identity (Appendix U). And the deterministic pass band is a property of who wrote the cases rather than a ceiling: the same store scores 63.6 % here and 33.8 % on the externally authored subset, where the lift falls to $+11.7$ and $+16.9$ pt. Coverage, per-category power and LLM-quality sensitivity follow below.

Substring-scorer blind spots. A 50-case stratified audit (Appendix X) finds 30 % of LETHE's deterministic failures are scorer artifacts, confined to prefix_collision and clause-level supersession; the canonicalization categories carrying the placement finding contain zero. A full NLI re-score confirms it: deterministic LETHE rises only 63.6→66.2 %, and on those categories the judge is if anything stricter than the scorer (-1 case each), so the central result is scorer-invariant. A naive `DELETE-LIKE` SQL baseline scores 23 pt below LETHE (Appendix W): the BM25-precise purge, not the depth scalar, is what avoids substring over-deletion.

English emphasis and a thin script tail. The hand-crafted layer is English, and cross_lingual_identifier samples 9 script families very unevenly (Latin 12, Greek 7, Han 5, down to Devanagari and Thai at 1 each). Neither hooked system is perfect here: LETHE+LLM scores 32/38 and LangGraph+LLM 37/38. Their misses do not concentrate in the thin tail, which is what a script-difficulty reading would predict. LETHE+LLM loses one case each in Latin, Greek, Hangul, Cyrillic and Hebrew — the first two are its best-powered families — along with its single Devanagari case; LangGraph+LLM's one miss is in Hebrew. Only Latin/Greek/Han are well powered ($n = 24$); five of the nine families hold 1–3 cases each, and those slices cannot distinguish among hook systems in either direction. Deeper per-script coverage is future work.

compound_fact is a primitive-existence test, not a forgetting-capability test. This category tests whether a system can perform partial supersession (update one clause of a two-fact row). LETHE exposes this via its `supersede` in edit mode (`surrender(mode="edit")` in the store's own API). MEMPALACE exposes one only over knowledge-graph triples, not stored rows. MEM0 and LangGraph have no native one either, but the Protocol asks what a store can be made to do rather than what its API names: LangGraph+LLM composes the operation as delete-old-then-add-merged over the same `InMemoryStore` and reaches 27/40 (Appendix Q). Their deterministic zeros are planner decisions, not absent primitives, and we keep the category as a diagnostic. It should therefore be read out of the headline, and we report the lift both ways: excluding it, LETHE goes 245/345 to 315/345, $+20.3$ **pt** against the full-suite $+25.2$, and **64 of those 72 fixes** are identifier_obfuscation and cross_lingual_identifier. The narrower figure is the one the placement claim rests on: what mutation-time placement buys is canonicalization, and the partial-edit primitive is a separate axis that happens to co-vary with it here. It also accounts for 10 of the 21 human–judge disagreements in our IAA sample (Appendix F.3).

Memory-system coverage, per-category statistical power, and the hook's sensitivity to LLM quality are taken up in turn below.

Memory-system coverage. We benchmark four primary open systems with shipped Adapter Protocol implementations (LETHE, MEM0, LangGraph `InMemoryStore`, MEMPALACE) and report nine further systems — A-MEM, Letta/MemGPT, Graphiti, OpenMemory, HippoRAG, and the LLM-hooked variants — under the same contract. Per-system engagement notes, strict- vs. evaluable- denominator scores, the two MEM0 `infer` configurations, and the systems we could not run (Cognee (Cognee

contributors, 2024), Zep CE (Rasmussen et al., 2025), and hosted-only services) are in Appendix R.1. Any backend supporting `add`/`delete` composition can join (§4).

LLM dependency in 3 attack categories. Because the lift is concentrated in compound_fact, identifier_obfuscation and cross_lingual_identifier (§5), a fully offline deployment has no recourse in exactly those three beyond its own edit and canonicalization primitives.

Bench size. At n=36–40 per category the per-category contrasts are well-powered, but only 3 of 10 categories survive Holm correction for the hook comparison (§5) — the lift is real but statistically *localized*, and per-category claims outside those three should be read as descriptive. identifier_obfuscation was N=18 before Qwen over-rejection was redressed and is now N=38 with additional hand-crafted cases under the same protocol.

LLM-quality sensitivity. The hook needs a competent JSON-follower and degrades below that bar (§5). Bench-construction provenance and the evidence that would update each claim are given above.

Bench construction. The 132 hand-crafted cases were written by the research team and re-labeled by 10+ external annotators (§3.3); the 253 expansion cases are LLM-drafted and oracle-validated. Formal cool-off self-IAA and broader external panels are future work.

What would update our claims. Architecture-agnosticism rests on three backends and six LLMs, replicated on a 77-case external-authored subset (Appendix U); a store that broke 70 % deterministically on a comparably hard suite, or a backend where the same contract failed to transfer, would be a substantive update. The cross_lingual_identifier category is the one we would most expect to move: it is repaired rather than redesigned (§5.3), and a Unicode normaliser still reaches 7/38 on it.

J Cross-system deployment profile

Figure 1 is in §5.5; this section gives the per-system commentary behind it.

What the deployments add. The thirteen rows carry what the ablation cannot: architecture *reorganises* the failure surface rather than raising or lowering it. Mem0+v3's link-and-keep router reaches 71 and 39 % on the two canonicalization columns, the highest of any row that runs no model at mutation time, and pays for it where the deterministic stores are strong — 31 % on prefix_collision and on recursive_supersession against 82 and 92. Letta+LLM inverts the trade, topping both canonicalization columns at 100 % while sitting at zero on compound_fact, which it has no primitive to express. HippoRAG's KG extraction scores 0/38 on both surface-variant categories, which is the same zero the annotate-on-write stores reach under reachability scoring: canonicalization is not automatic in any regime. We report these rows as deployment evidence that motivated the controlled experiment and is explained by it, not as support for a claim about call sites.

K Recall–forgetting trade-off

Plotting the recall axis (Memora-weekly pass rate) against the forgetting axis (Table 1) places each architecture in two dimensions (Figure 2). With three backends in five configurations we read this as a sample of the landscape, not a population-level Pareto frontier.

Three readings. **(1)** Lethe and LangGraph are non-dominated in opposite corners (0.7 pt on forgetting, 13.3 pt on recall), while MemPalace is dominated by LangGraph on both axes: its 40.0 % recall clears Lethe's 31.3 but not LangGraph's 44.7, and it buys that with zero forgetting. **(2)** The hook moves both backends *vertically* — it is applied at mutation time, so recall is untouched. **(3)** Once both backends have the hook, the forgetting axis stops separating them: 89.4 against 88.8 %, a gap inside the 1.1 pt run-to-run spread of either. What still separates them is recall, by 13.3 pt. A deployment choosing between them should therefore choose on recall, because the hook has already equalised the other axis — which is the practical form of the architecture-agnosticism result.

L The escalation rule

Three signals, each tied to a known failure of the deterministic plane and each computed from the request and the rows already retrieved (no extra model call): **cross-script**, a non-ASCII letter

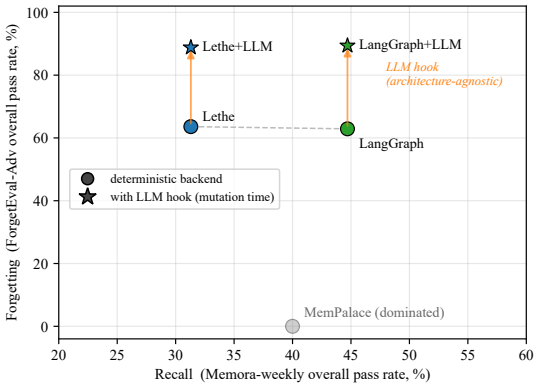


Figure 2: Recall–forgetting trade-off across three backends in five configurations. LETHE (forgetting corner) and LangGraph (recall corner) are non-dominated; MEMPALACE is dominated on both axes. The mutation-time hook lifts both non-dominated backends vertically — it buys forgetting without spending recall.

making NFKC-equality unable to decide identity; **variant-family**, more rows carrying the request's identifier — compared with punctuation stripped — than the NFKC-equal group about to be deleted; **compound-row**, a `supersede` whose target holds multiple clauses, where whole-row replacement must drop the one the request did not restate.

Against 63.6 % deterministic with no calls and 88.8 % always-calling with 445, the router scores **83.1 % on 140 calls**: each call it makes is worth 0.54 cases where an always-calling hook averages 0.22. Routing captures **77.4 % of the lift for 31 % of the calls** (+19.5 pt of the hook's +25.2, 78 fixed against 3 broken), and dropped onto LangGraph's `InMemoryStore` unchanged — it reads only `recall_texts`, which every backend implements — it captures **84.3 %** (+22.3 pt of a +26.5 lift, 86 fixed, **0 broken**) at 146 calls. The rule is a property of the control plane rather than of one store, but it is not equally good on both, and the difference is instructive.

Where a cheap rule fails is legible. All three regressions are negation_trap supersessions the hook handles correctly, so the loss is the router declining to escalate rather than the model erring. Its misses concentrate too: of the 30 cases the hook fixes and the router skips, 11 are identifier_obfuscation and 2 are cross_lingual_identifier — cross-script variation is detectable from the request without a model, same-script variation is where the headroom remains. An oracle escalating exactly the mutations the hook helps on avoids its two regressions and so scores $245 + 99 = 344/385$, two above always-calling, for 99 of the 445 calls (22 %).

M LATENCY MEASUREMENTS

Every figure here is wall/cases from the run files, the same source as Table 27. Wall clock covers setup, store construction, the cases and teardown, so each is the cost of reproducing that row rather than an isolated per-case cost.

The MEM0 figure includes a per-test cold start (Qdrant index and LLM-client construction); a pooled setup would close part of that gap, so read it as out-of-the-box adapter cost. The hook runs at mutation time only and leaves the recall path untouched.

N MEMORA CROSS-EVALUATION, PER-PERSONA DETAIL

All three systems score $\leq 6\,\%$ on Memora's recommending task, which requires user-preference modelling outside the forgetting axis; the aggregate totals above are dominated by Remember and Reason. Table 10 aggregates the same runs by Memora task.

Table 8: Per-case cost on the 385-case suite, single CPU, no GPU. The mutation-time hook costs about ten times the deterministic store it wraps, and still an order of magnitude less than A-MEM, whose cost is write-time work no deletion path consults.

System	wall (s)	cases	per case
LangGraph `InMemoryStore`	35	385	90 ms
LETHE	37	385	96 ms
MEMPALACE	247	385	641 ms
MEM0 `infer=False`	1533	385	4.0 s
LETHE+LLM	394	385	1.0 s
LangGraph+LLM	453	385	1.2 s
A-MEM	10068	385	26.2 s

Table 9: Three FORGETEVAL adapters on Memora-weekly (10 personas, 150 questions). MEMPALACE flips axis between the two benchmarks: 0/385 on FORGETEVAL-Adv, 60/150 here.

System	Remember	Reason	Recomm.	Total
LETHE	17/50	30/50	0/50	47/150 (31%)
LangGraph	**32/50**	**32/50**	3/50	**67/150 (45%)**
MEMPALACE	28/50	**32/50**	0/50	60/150 (40%)

The translation from Memora's data to our Adapter Protocol is released in `scripts/eval_on_memora.py`; the per-question verdicts (system, persona, task, question_id, pass/fail, memory_recall_rate, forgetting_rate) are in `data/memora_xeval_all_personas.json`.

O FACTCONSOLIDATION CROSS-EVALUATION (MEMORYAGENTBENCH)

To probe an independent third-party recall surface that is the natural dual to the Memora cross-evaluation (§5.4), we run six systems — four in-house adapters plus the two LLM-hook variants — on the *FactConsolidation* task of MemoryAgentBench (Hu et al., 2026) (ICLR 2026), an MQUAKE-derived single- and multi-hop fact-supersession benchmark. We use four context-length buckets from the HF release `ai-hyz/MemoryAgentBench/Conflict_Resolution`: sh/mh $\times$ 6K/32K, the full 100 questions per bucket ($=$400 questions total). We skip the 64K and 262K buckets ($>$270K-character contexts blow the in-memory index budget on single-CPU adapters). Each fact is inscribed in source order; questions are scored by case-insensitive substring of the gold answer in the top-10 retrieval.

Three readings. (1) Single-hop saturates at 100 % across all six systems and both context lengths. Both the original and the counterfactual edit appear in the haystack; top-10 substring scoring finds the gold answer for either ordering. This is the prediction of our §2 complementarity claim: FactConsolidation tests "can the system retrieve the edit?" (a recall-plane question) not "can the system command the original to disappear?" (a control-plane question). **(2) Multi-hop drops sharply (28–37 % on 6K, 17–19 % on 32K).** Multi-hop requires reasoning across chained facts that lexical retrieval alone cannot perform; the remaining 60–80 % need an LLM reasoning step over the retrieved chain, which is outside the forgetting axis (a perfect supersede primitive would not help). MEM0's slight 9 pt lead on mh_6k (37 vs. 28) and 2 pt on mh_32k reflects its BM25 + entity multi-signal scoring helping chain navigation in lexical surface form. **(3) LLM-hook variants are *numerically identical* to their deterministic backbones on every bucket.** This is the axis-flip prediction: the mutation-time hook is a control-plane lever (fires on supersede/release/purge); FactConsolidation exercises only the recall plane (inscribe-then-retrieve), so the hook never fires and the result is the deterministic baseline. The same systems differ by 25.2 and 26.5 pt on FORGETEVAL-Adv (Table 1) where control-plane mutations are invoked. Independent third-party data shows the two planes are distinct levers. Concurrent work Reddy & Challaram (2026) reach the same conclusion from the opposite direction: on this FactConsolidation surface, replacing the LLM-judgment answer pipeline with candidate extraction plus a deterministic `max(serial)` *improves* single-hop accuracy by 10.8 pt, because freshness consolidation is a recall-plane assembly problem the mutation-time hook is not meant to solve — consistent with our finding that deterministic stores retain the lexical/temporal

Table 10: By-task aggregate (50 questions per task). Recommending collapses uniformly because no adapter implements user-preference modeling, which is outside the forgetting axis as we define it.

Task	LETHE	LangGraph	MEMPALACE
Remembering	17/50 (34%)	32/50 (64%)	28/50 (56%)
Reasoning	30/50 (60%)	32/50 (64%)	32/50 (64%)
Recommending	0/50 (0%)	3/50 (6%)	0/50 (0%)

Table 11: MemoryAgentBench Conflict_Resolution (`ai-hyz/MemoryAgentBench`), 100 questions per bucket, top-10 substring scoring. Six systems: four in-house adapters plus two LLM-hook variants. "sh"=single-hop, "mh"=multi-hop chained reasoning. The LLM-hook variants (LETHE+LLM, LangGraph+LLM) score *identically* to their deterministic backbones — as expected, since FactConsolidation is pure recall (no supersede/release/purge primitives invoked), the mutation-time hook never fires. This is the axis-flip third-party check (§5.4). Per-bucket totals in `data/factconsolidation_full.json`.

System	sh_6k	mh_6k	sh_32k	mh_32k
LETHE	100/100	28/100	100/100	17/100
LangGraph	100/100	28/100	100/100	17/100
MEMPALACE	100/100	28/100	100/100	17/100
MEM0 (`infer=False`)	100/100	**37/100**	100/100	**19/100**
LETHE+LLM	100/100	28/100	100/100	17/100
LangGraph+LLM	100/100	28/100	100/100	17/100

categories at $\geq 95\,\%$ while the hook is reserved for the intent-aware deletion categories deterministic scoring cannot reach.

P CROSS-LLM ABLATION ON THE HOOK

We run the same hook prompts across **six LLMs spanning five vendors and two price tiers** on the 385-case suite, LETHE backend fixed so the only variable is the LLM behind the mutation-time planner: GPT-5.5, Claude-Opus-4.8, Gemini-3.1-pro, DeepSeek-V4-Pro, DeepSeek-V4-Flash and Qwen-3.6, all via OpenRouter at submission time. Three further models were run only on an earlier revision of the suite and are not comparable to these: DeepSeek-V3 and Qwen-2.5-72B via SiliconFlow, and Llama-3.1-70B, which fails the JSON contract in either revision. Within this table the V4 Pro/Flash pair isolates price tier at fixed vendor and generation; no generational pair survives on the current suite, so we draw no within-family generational comparison.

Table 12: Cross-LLM hook ablation, LETHE backend fixed. Left against a deterministic 63.6 %. Six models across five vendors span a $141\times$ range in per-run cost and land within 11 cases (2.9 pt) of each other. Cost is the run's own metered spend.

Hook LLM	Vendor	Pass rate	$/run
(none)	—	63.6	0
DeepSeek-V4-Flash	DeepSeek	**88.8**	0.016
DeepSeek-V4-Pro	DeepSeek	**90.1**	0.185
GPT-5.5	OpenAI	**90.4**	2.117
Claude-Opus-4.8	Anthropic	**88.6**	1.610
Gemini-3.1-pro	Google	**90.1**	2.227
Qwen-3.6	Alibaba	**91.4**	1.805

† Llama-3.1-70B's response format does not parse into our strict JSON contract; it falls back to the deterministic baseline. We treat this as a known integration gap, not a lift result.

Three readings. **(1) Every model that follows the contract gets most of the lift.** Six models across five vendors score 88.6–91.4 %, a $+25$ to $+28$ pt lift over the 63.6 % deterministic baseline, across per-run costs differing by $141\times$ ($0.016 for V4-Flash against $2.23 for Gemini-3.1-pro). The

11 cases (2.9 pt) between them are real but small next to that lift, and they are not ordered by vendor or by price. **(2) Price tier buys little.** DeepSeek's Flash and Pro isolate price at fixed vendor and generation: the pricier model leads by 1.3 pt (90.1 against 88.8), an order of magnitude below the $+25$ pt the hook itself is worth, and within the run-to-run spread of either. Across vendors the ordering is not by price at all: the third most expensive model scores highest (Qwen-3.6, \$1.80, 91.4) and the fourth scores lowest (Opus-4.8, \$1.61, 88.6), 2.9 pt apart at 89 % of the price. The hook asks for one small structured decision per mutation, not for reasoning depth, and paying for depth does not help it. **(3) What it does need is a model that follows the contract.** Llama-3.1-70B cannot parse it and receives none of the lift; every model that can parse it lands in the same 2.9-point band.

(4) Per-category personality is small. The six agree to within a few cases on most categories; the only material spread is compound_fact, which is also the least stable category across repeats (17 of the 38 non-unanimous cases below), so we read it as a wide band rather than a ranking.

Qwen-2.5-72B appears only as an admission judge (§3.3) and was never run as a hook LLM on this suite, so the table supports no within-family generational comparison against Qwen-3.6.

Prompt-robustness ablation. To test whether the lift is a fragile prompt-engineering artifact, we swap the `supersede` prompt for two degraded variants on a fixed model and backend: **zero-shot** (the shipped prompt with its four examples deleted) and **reworded** (a terse paraphrase, no examples, different phrasing).

All three arms run on DeepSeek-V3, one model held fixed across the three prompts; they are comparable to each other and not to the V4-Flash headline elsewhere in the paper. They score **331, 331 and 340** of 385 against the same deterministic 245 — a $+86$ to $+95$ lift, with the shipped prompt the weaker of the two extremes. Deleting all four examples changes the score by zero cases and the terse paraphrase is the best of the three; compound_fact moves 20/20/26 in the same order. The lift is prompt-invariant within a few cases, and an example-free prompt recovers all of it, so it is not a prompt-engineering artifact. Variant verdicts are in `data/ablate_*_v07_probed.json`.

Run-to-run variance, and what it forbids us from claiming. Every cell in Table 12 is a single run at temperature 0, which does *not* make an API-served model deterministic. We therefore repeat the reference configuration end-to-end **five** times: 348, 344, 342, 338 and 338 of 385 — **mean 342.0, sd 4.2 cases (1.1 pt)**. Every headline figure in this paper comes from the run at that mean rather than from the best of the five, and the regressions range from 0 to 2 across them, so the hook is close to but not exactly non-regressing. An earlier seven-run repeat of DeepSeek-V4-Pro under a narrower completion budget gave a tighter spread (sd 0.46 pt); we quote the wider one because it bounds the configuration the headline uses. Per case the picture is noisier than either aggregate: **38/385 (9.9 %)** of cases do not return a unanimous verdict across those seven V4-Pro runs, and that instability is concentrated in compound_fact (17), identifier_obfuscation (12) and cross_lingual_identifier (5) — the same three categories that carry the placement finding. We report this rather than smooth it: the categories doing the most work in our argument are also the ones an individual re-run is least likely to reproduce case-for-case.

Two consequences follow, one restrictive and one reassuring. **(i)** The 2.9-pt spread across the six frontier models in Table 12 is *smaller than twice* a single model's own run-to-run range, so no within-tier ranking can be read from it; the defensible claim is that every frontier model saturates the lift, not that any one of them is best. Statements of the form "model X leads model Y by 1–2 pt" are not supported by this evidence and we do not make them. **(ii)** The full-suite lift is $+25.2$ pt (63.6 against 88.8 %), roughly **23 times** the sd of 1.1 pt from the five-run repeat of the reference configuration that bounds the headline, and the $+20.3$ pt lift on the 345 forgetting cases is still $\sim 18\times$ that sd — so run-to-run noise does not reach the headline. Per-run verdicts are released in `data/hook_run_variance_*.json`.

Execution mode does not bias the result. Five of the seven V4-Pro runs were executed sequentially and the other two with five concurrent workers (each holding its own adapter and embedder, since `GeneratedCase.run` resets the store first). The parallel runs scored 89.6 and 90.4 %, inside the sequential range $[89.6, 90.9]$, at $\sim 4.7\times$ lower wall time. We note this because the concurrency is a harness convenience, and it would be a poor one if it moved the numbers.

Q CROSS-ARCHITECTURE LLM-HOOK ABLATION

To disentangle the LLM-hook contribution from LETHE-specific primitives, we apply the same hook (DeepSeek-V4-Flash, same JSON prompts) to LangGraph's `InMemoryStore`. LangGraph has no in-place edit primitive: for partial `supersede` the LLM-planned merged text is added as a fresh row replacing the deleted old row (functionally equivalent to LETHE's `surrender(edit)` under substring scoring).

Both backends, end to end. Same adapter code, identical hook, one suite:

Backend	deterministic	+hook	Δ
reference store	63.6	88.8	+25.2
LangGraph `InMemoryStore`	62.9	89.4	+26.5

Table 13 gives the per-category split. Paired tests give $p = 4\times10^{-27}$ and 5×10^{-26}; the discordance is one-directional on both (99/101 and 107/112 fixes). The two backends share no storage code, land within **0.5 pt** of each other, and concentrate 91/99 and 96/107 of their gains in the *same* three categories. Both figures are single runs of a configuration whose run-to-run spread is 1.1 pt (Appendix P), so the 0.5 pt between them is smaller than the noise on either and should be read as agreement rather than as a ranking; their lifts differ by 1.3 pt because their deterministic starting points differ by 0.7.

Table 13: The same contract on two unrelated backends, FORGETEVAL-Adv (385 cases), DeepSeek-V4-Flash at mutation time on both; each column is the designated run of §5 and sums to the total beneath it. The two land 0.5 pt apart, inside the 1.1 pt run-to-run spread of either.

Category	LETHE+LLM	LangGraph+LLM
substring_trap	36/36 (100%)	36/36 (100%)
prefix_collision	32/39 (82%)	32/39 (82%)
paraphrase_supersession	31/38 (82%)	31/38 (82%)
negation_trap	36/40 (90%)	35/40 (88%)
temporal_qualifier	37/37 (100%)	37/37 (100%)
shared_attribute	40/40 (100%)	38/40 (95%)
compound_fact	27/40 (68%)	27/40 (68%)
identifier_obfuscation	35/38 (92%)	35/38 (92%)
cross_lingual_identifier	32/38 (84%)	37/38 (97%)
recursive_supersession	36/39 (92%)	36/39 (92%)
Overall	**342/385 (88.8 %)**	**344/385 (89.4 %)**

Note that LangGraph+LLM matches LETHE+LLM on compound_fact (27/40 each) and on identifier_obfuscation (35/38 each) despite lacking a native edit primitive — the LLM-planned "delete-old + add-merged-new" suffices. This rules out the edit primitive as a uniquely necessary architectural feature for this benchmark; the architectural contribution is the *hook pattern* (narrow JSON contract at mutation time), not the specific storage backend.

R EXTENDED-SYSTEM EVALUATIONS

Five deployed stores, each run on the full 385-case suite through the same Adapter Protocol and the same probe scorer, with the versions, endpoints and integration details in Appendix R.1. All five expose no `release` primitive, so the 75 cases needing one are N/A: counted as failures in the strict row and excluded from the evaluable one. A-MEM, Letta and OpenMemory share the mechanism §5.3 describes, a canonical form the deletion path never consults; Graphiti fails earlier, in edge synthesis; Letta+LLM is the control that repairs Letta. Each gets a table and only what is specific to it.

R.1 HOW EACH SYSTEM WAS RUN

This appendix collects the per-system engagement reports referenced from Appendix I: which systems ported to the Adapter Protocol, which primitives each lacks and what that costs it — every

re-measured system is scored over all 385 cases, so a missing primitive loses the cases it would have served rather than excusing them — the deployment path used, and the two MEM0 `infer` configurations. Per-category breakdowns for each system are in the system-specific appendices that follow.

What the requirements do to a system that looks solved. Twelve of the thirteen configurations are measured with the survivor and probing requirements of §5.3 in force; only HippoRAG is not, and it scores zero on both canonicalization categories either way.

Their canonicalization scores were not usable as evidence until the requirements were in force, because ranked-only scoring credits a store for burying a string it never removed. Under reachability scoring all three fall:

	ranked-only scoring	reachability scoring
A-MEM	100 % / 100 %	**0/38 / 0/38**
OpenMemory	97 % / 100 %	**0/38 / 0/38**
Letta	100 % / 100 %	**0/38 / 0/38**

The columns are identifier_obfuscation and cross_lingual_identifier. Nothing about these systems changed; the question did. Asking whether the forbidden string still *ranks* rewards a store that rewrites a fact into a form the query no longer matches, and all three rewrite at write time. Asking whether it is still *reachable*, by querying for it directly, finds the fact every time.

Three controls say this is the requirement working rather than a uniformly harsher scorer. Within Letta, paraphrase_supersession and recursive_supersession are unchanged at 84 % and 92 %, so the drop is confined to the two categories where rewriting can be mistaken for removal. Letta *with* a mutation-time hook holds both at 38/38 under the same scoring, so a system that genuinely removes the fact still scores full marks. And probing costs the deterministic store **exactly zero cases** — 244/385 with and without it — while it costs *normalize*, which rewrites rather than deletes, 8.9 pt and two fifths of its canonicalization score (96 to 59); the three requirements together cost it two thirds (96 to 32). The requirement separates removal from rewriting and charges nothing to a store that removes.

The implication is for the next forgetting benchmark rather than for these systems. A suite that scores deletion by absence from one query's results will report a store that rewrites as a store that forgets, and will report it at 100 %. Probing each forbidden string directly costs one retrieval per string, and is the difference between those numbers and these.

Table 14 records how each system was run. Per-system results are in Appendix R; nothing in the integration is load-bearing for the placement claim, which is settled by the ablation on one store.

Table 14: How each extended system was run, and why three were not. All are self-hosted on one Linux host with no hosted-service dependency.

System	Deployment	Note
A-MEM (Xu et al., 2025)	ChromaDB; DeepSeek-V3 (SiliconFlow)	tag extraction per note
Letta (Letta contributors, 2024; Packer et al., 2023)	`letta/letta` image `sha256:aa66c3ee...c102`, pgvector	archival REST direct
Graphiti (Zep AI, 2025)	graphiti-core, Neo4j 5.x; DeepSeek-V3.1-Terminus	KG episodes
OpenMemory	Docker, synthetic 1536-d embeddings	no LLM on recall
HippoRAG (Gutiérrez et al., 2024)	Python 3.10 image, `hipporag==2.0.0a4`; BAAI/bge-m3	*not re-measured*
Cognee (Cognee contributors, 2024) v1.0.9	—	forget API is dataset-level
Zep CE (Rasmussen et al., 2025)	—	support ended 2025-04
hosted-only services	—	outside the no-API-key envelope

Two of these need a sentence beyond the table. HippoRAG is the one system here not re-measured under the requirements of §5.3; it scores **31/385 (8.0 %)** either way and **0/38** on both canonicalization categories either way, so the requirements cannot move it, but its row in Fig. 1 is the only one carrying a pre-repair measurement. And A-MEM is absent from the Memora cross-evaluation (§5.4): its

per-session tag-extraction call makes the ten-persona suite run ≥ 8 wall-clock hours, which did not fit the submission window.

The Protocol's behavioural contract (§4) means any backend supporting `add` / `delete` composition can join; we encourage community-contributed adapters.

MEM0 version and configuration: both `infer` modes evaluated. We evaluate MEM0 v2.0.18 (the latest `mem0ai` release when the runs were made) on the full 385-case adversarial suite in *two* configurations, because `infer=True` engages MEM0's LLM-driven ADD/UPDATE/DELETE router (the design distinctive of MEM0 relative to a plain vector store) and is the configuration a real MEM0 deployment runs. (a) `infer=False` (ADD-only): **252/385 = 65.5 %** (Table 1), the path used for the headline comparator. (b) `infer=True`, measured with MEM0's own completion cap lifted: **229/385 = 59.5 %**, full per-category breakdown in Appendix S. The router runs DeepSeek-V4-Flash, the same model the mutation-time hook uses, so the two differ in where the model sits rather than in which model it is. Table 15 gives the two configurations category by category. The router trades deletion precision for consolidation: it gains on both canonicalization categories and on compound_fact, where merging a row is a partial edit, and loses enough elsewhere to end 6.0 pt below `infer=False` overall.

Two measurements of this configuration should not be mistaken for it. Scored without the survivor requirement it reaches 100 % on both canonicalization categories, which is the artifact §5.3 describes and which this router is the system that produced. And its own configuration capped completions at 1024 tokens, on the prompt that emits merged row text; under that cap it scores 2/38 and 0/38 on the same two categories rather than the 27/38 and 15/38 here. Neither figure is the system.

Table 15: MEM0 with and without its ADD/UPDATE/DELETE router, on the repaired suite under probe scoring and an uncapped completion budget. Generated by `scripts/mem0_delta_table.py`.

Category	`infer=False`	`infer=True`	Δ
substring_trap	30/36 (83)	28/36 (78)	−6
prefix_collision	13/39 (33)	12/39 (31)	−3
paraphrase_supersession	31/38 (82)	16/38 (42)	−39
negation_trap	38/40 (95)	36/40 (90)	−5
temporal_qualifier	37/37 (100)	29/37 (78)	−22
shared_attribute	37/40 (92)	37/40 (92)	+0
compound_fact	0/40 (0)	17/40 (42)	+42
identifier_obfuscation	23/38 (61)	27/38 (71)	+11
cross_lingual_identifier	7/38 (18)	15/38 (39)	+21
recursive_supersession	36/39 (92)	12/39 (31)	−62
Overall	**252/385 (65.5)**	**229/385 (59.5)**	−6.0

This refines the §5.5 placement-regime characterization rather than confirming it as written. MEM0+v3 sits in the inscribe-time-LLM regime alongside A-MEM and Letta, and it is the only member of that regime whose write-time work survives into the score: identifier_obfuscation 27/38 and cross_lingual_identifier 15/38, against **0/38** on both for A-MEM and Letta. The router is not a better canonicalizer than their write-time normalisation; the difference is that it consolidates onto one row, and a row is something a later delete can act on. The regime is also wide — A-MEM 143/385 (37.1 %) against MEM0+v3 229/385 (59.5 %) — so the extraction prompt and router algorithm, not the placement alone, decide where a system in it lands. One caveat on the comparison: MEM0's own default LLM is OpenAI `gpt-4o-mini`, which our no-API-key reproducibility envelope rules out, and its extraction prompt is tuned for it. We report the router driven by the same model as the hook, which is the substitution that keeps the placement contrast clean, not the one that shows MEM0 at its best.

R.2 A-MEM

A-MEM scores **0/38** on both canonicalization categories, below even deterministic LETHE (5 % and 3 %). The zeros are not an extraction failure: the agentic tags do canonicalize the identifier, which is why scoring that asked only whether a forbidden string still *ranks* read these two columns as solved. What the deletion path leaves is the annotated rewrite — a row the final query no longer matches and a direct probe still reaches. Its best columns are the supersession-shaped ones, where it matches the

Table 16: A-MEM on the full 385-case FORGETEVAL-Adv (DeepSeek-V3 via SiliconFlow, evolution disabled). A-MEM lacks `release`, so the 75 cases needing one are N/A: they count as failures in the strict row (143/385 = 37.1 %) and are excluded from the evaluable one (143/310 = 46.1 %). The right-hand column is deterministic LETHE (245/385 = 63.6 %) on the same cases.

Category	A-MEM	vs. LETHE
substring_trap	**20/20 (100%) [16 N/A]**	92 %
prefix_collision	0/39 (0%)	82 %
paraphrase_supersession	31/38 (82%)	82 %
negation_trap	**19/21 (90%) [19 N/A]**	95 %
temporal_qualifier	**37/37 (100%)**	100 %
shared_attribute	— [40 N/A]	88 %
compound_fact	0/40 (0%)	0 %
identifier_obfuscation	0/38 (0%)	5 %
cross_lingual_identifier	0/38 (0%)	3 %
recursive_supersession	**36/39 (92%)**	92 %
Overall (evaluable)	**143/310 (46.1%)**	63.6 %
Overall (strict, N/A=fail)	**143/385 (37.1%)**	63.6 %

deterministic baseline rather than beating it: temporal_qualifier 37/37 and recursive_supersession 36/39.

R.3 LETTA

Table 17: Letta (Docker, PostgreSQL+pgvector) on the full 385-case FORGETEVAL-Adv. Recall is direct archival-memory search (no LLM in the loop), scored under the survivor and probing requirements of §5.3. The 75 cases needing `release`, which Letta does not expose, are N/A: failures in the strict row (127/385 = 33.0 %) and excluded from the evaluable one (127/310 = 41.0 %).

Category	Letta	vs. LETHE
substring_trap	**19/20 (95%) [16 N/A]**	92 %
prefix_collision	0/39 (0%)	82 %
paraphrase_supersession	32/38 (84%)	82 %
negation_trap	**19/21 (90%) [19 N/A]**	95 %
temporal_qualifier	21/37 (57%)	100 %
shared_attribute	— [40 N/A]	88 %
compound_fact	0/40 (0%)	0 %
identifier_obfuscation	0/38 (0%)	5 %
cross_lingual_identifier	0/38 (0%)	3 %
recursive_supersession	**36/39 (92%)**	92 %
Overall (evaluable)	**127/310 (41.0%)**	63.6 %
Overall (strict, N/A=fail)	**127/385 (33.0%)**	63.6 %

Letta and A-MEM converge on nearly the same profile from very different architectures — raw passage embedding over pgvector against LLM-mediated tag extraction over ChromaDB — agreeing to within a single case on every category but temporal_qualifier. The convergence has two causes and they mean different things. The prefix_collision, shared_attribute and compound_fact zeros are a primitive gap that no representation choice repairs. The two canonicalization zeros are not a gap in what the store can do but in what it consults.

R.4 LETTA WITH A MUTATION-TIME HOOK

The same deployment with the mutation-time contract wired into the write path and nothing else changed — same store, same BAAI/bge-m3 embedder, same adapter, same scorer — takes both canonicalization columns from **0/38** to **38/38**. This is the paper's one non-observational cross-system comparison: only the call site moves, and it moves on a store we did not build. It is paid for elsewhere, recursive_supersession falling 36/39 to 10/39 and temporal_qualifier 21/37 to 11/37, for a net gain of 64 cases. The 38/38 pair is in-house; on the externally-authored subset the same hook reaches 7/8 on identifier_obfuscation and 0/8 on cross_lingual_identifier (Appendix U).

Table 18: Letta with a mutation-time LLM hook, full 385-case FORGETEVAL-Adv, scored under the survivor and probing requirements of §5.3. Recall is unchanged from Table 17 (direct archival-memory search, no LLM in the loop). The 75 release-dependent cases are N/A here too: 191/385 = 49.6 % strict, 191/310 = 61.6 % evaluable.

Category	Letta+LLM	vs. LETHE
substring_trap	**19/20 (95%) [16 N/A]**	92 %
prefix_collision	26/39 (67%)	82 %
paraphrase_supersession	28/38 (74%)	82 %
negation_trap	**21/21 (100%) [19 N/A]**	95 %
temporal_qualifier	11/37 (30%)	100 %
shared_attribute	— [40 N/A]	88 %
compound_fact	0/40 (0%)	0 %
identifier_obfuscation	**38/38 (100%)**	5 %
cross_lingual_identifier	**38/38 (100%)**	3 %
recursive_supersession	10/39 (26%)	92 %
Overall (evaluable)	**191/310 (61.6%)**	63.6 %
Overall (strict, N/A=fail)	**191/385 (49.6%)**	63.6 %

Table 19: OpenMemory (`mem0ai` server profile, vector-only) on the full 385-case FORGETEVAL-Adv. The 75 cases needing `release` are N/A: failures in the strict row (105/385 = 27.3 %), excluded from the evaluable one (105/310 = 33.9 %). The right-hand column is deterministic LETHE on the same cases.

Category	OpenMemory	vs. LETHE
substring_trap	17/20 (85%) [16 N/A]	92 %
prefix_collision	0/39 (0%)	82 %
paraphrase_supersession	8/38 (21%)	82 %
negation_trap	14/21 (67%) [19 N/A]	95 %
temporal_qualifier	32/37 (86%)	100 %
shared_attribute	— [40 N/A]	88 %
compound_fact	3/40 (8%)	0 %
identifier_obfuscation	0/38 (0%)	5 %
cross_lingual_identifier	0/38 (0%)	3 %
recursive_supersession	31/39 (79%)	92 %
Overall (evaluable)	**105/310 (33.9%)**	63.6 %
Overall (strict, N/A=fail)	**105/385 (27.3%)**	63.6 %

R.5 OPENMEMORY

OpenMemory sits at the same canonicalization floor as Letta, from the same cause. An earlier measurement put it at 97 and 100 % on those two columns, and we read the agreement with Letta — across a synthetic-embedding swap — as evidence that the vector neighbourhood was doing the canonicalizing. OpenMemory's container would not serve its own embedding endpoint, so its vectors came from a deterministic hash into 1536 dimensions, which is why agreeing with Letta's real embeddings is the informative part. Under reachability scoring both are at zero, so the agreement identifies a shared mechanism rather than a shared capability.

R.6 GRAPHITI

Graphiti is the one store whose failure is representational rather than procedural. It synthesises edges and stores the synthesised `fact` string, so the adversarial `must_not_contain` substrings drop out of the synthesis — a categorical mismatch with this suite, not a tuning issue. Its two strongest columns are the ones edge-level updates are built to serve, recursive_supersession 12/39 and paraphrase_supersession 7/38, both far below deterministic LETHE's 92 and 82 %; and temporal_qualifier, the category temporal edge invalidation exists for, is 4/37 against 100 %. The mechanism most specific to the architecture is not carrying the cases written for it.

Table 20: Graphiti on full 385-case FORGETEVAL-Adv (DeepSeek-V3.1-Terminus via SiliconFlow, Neo4j 5.x). Cases needing `release` are N/A — counted as failures in the strict row (29/385 = 7.5 %) and excluded from the evaluable one (29/310 = 9.4 %). The strict denominator is the one the primary systems are reported on.

Category	Graphiti	vs. LETHE
substring_trap	2/20 (10%) [16 N/A]	92 %
prefix_collision	0/39 (0%)	82 %
paraphrase_supersession	7/38 (18%)	82 %
negation_trap	2/21 (10%) [19 N/A]	95 %
temporal_qualifier	4/37 (11%)	100 %
shared_attribute	— [40 N/A]	88 %
compound_fact	2/40 (5%)	0 %
identifier_obfuscation	0/38 (0%)	5 %
cross_lingual_identifier	0/38 (0%)	3 %
recursive_supersession	12/39 (31%)	92 %
Overall (evaluable)	**29/310 (9.4%)**	63.6 %
Overall (strict, N/A=fail)	**29/385 (7.5%)**	63.6 %

S MEM0 v2.0.18 `infer=True` (TOKEN-EFFICIENT ROUTER) DETAILED BREAKDOWN

MEM0's headline architectural feature is its LLM-driven ADD/UPDATE/DELETE routing: at every `add()` call, MEM0 sends the new fact plus a fingerprint of existing memories to an LLM (the `ADDITIVE_EXTRACTION_PROMPT`) which decides whether to ADD a new row, UPDATE an existing one, or DELETE a stale one. This is the path `infer=True` engages; `infer=False` in our headline (Table 1) bypasses it and stores the literal text only. We therefore report the full 385-case suite under both modes for fairness.

The `infer=True` integration drives MEM0's router with DeepSeek-V4-Flash via OpenRouter — the same model and endpoint the mutation-time hook uses — with MEM0's completion cap lifted to 3000 tokens, since the router emits rewritten row text and a small budget truncates it. Every extraction parsed: a parse failure inside `Memory.add` raises, the harness re-raises rather than scoring the case, and all 385 cases completed. MEM0's extraction prompt is tuned for OpenAI `gpt-4o-mini`, which our no-API-key envelope rules out, so this is MEM0's algorithm on a substituted model rather than MEM0 as its authors ship it.

Table 21: MEM0 v2.0.18 on FORGETEVAL-Adv (385 cases), both `infer` modes, same Adapter Protocol, under probe scoring and an uncapped completion budget. "Δ" = infer=True − infer=False (positive = LLM router helps). Generated by `scripts/mem0_delta_table.py`, the same source as Table 15.

Category	`infer=False`	`infer=True`	Δ
substring_trap	30/36 (83)	28/36 (78)	−6
prefix_collision	13/39 (33)	12/39 (31)	−3
paraphrase_supersession	31/38 (82)	16/38 (42)	−39
negation_trap	38/40 (95)	36/40 (90)	−5
temporal_qualifier	37/37 (100)	29/37 (78)	−22
shared_attribute	37/40 (92)	37/40 (92)	+0
compound_fact	0/40 (0)	17/40 (42)	+42
identifier_obfuscation	23/38 (61)	27/38 (71)	+11
cross_lingual_identifier	7/38 (18)	15/38 (39)	+21
recursive_supersession	36/39 (92)	12/39 (31)	−62
Overall	**252/385 (65.5)**	**229/385 (59.5)**	−6.0

Reading the result. Switching the router on costs 6.0 pt overall, and what it costs is deletion precision: −39 on paraphrase_supersession and −62 on recursive_supersession, because the token-efficient `ADDITIVE_EXTRACTION_PROMPT` routes "A says X" followed by "A says Y" to LINK-AND-KEEP rather than UPDATE-AND-DELETE, leaving the old fact in the store. A write-time router that merges without seeing a deletion request loses precisely what the request would have told it, which is the controlled result of §5.5 appearing in a third-party system.

Three categories move the other way, and they are the ones a merge can serve. compound_fact gains $+42$, because merging two clauses at write time produces a row a later whole-row delete can act on correctly; the two canonicalization categories gain $+11$ (identifier_obfuscation, 23/38 $\rightarrow$ 27/38) and $+21$ (cross_lingual_identifier, 7/38 $\rightarrow$ 15/38), because the extraction step rewrites variant surface forms onto one row. The router trades deletion precision for consolidation rather than losing across the board, which is why the overall cost is 6.0 pt and not the collapse the two deletion categories on their own would suggest.

MEM0's own configuration capped completions on the prompt that emits merged row text, and every figure here is measured with that cap lifted (Appendix R.1): under the cap the two canonicalization categories read 2/38 and 0/38 rather than the 27/38 and 15/38 above. We do not run MEM0+v3 with OpenAI `gpt-4o-mini` (MEM0's tuned default), so the -6.0-pt drop bounds the `infer=True` loss in our setup rather than characterising MEM0 at its best.

T LONGMEMEVAL-S SETUP AND PER-TYPE RESULTS

Setup. We run LETHE v1 on LongMemEval-S (Wu et al., 2025) using the released `longmemeval_s.json` distribution (500 questions, 6 question types). Each question has a haystack of up to 158 conversational sessions; we follow the bench convention and use *session granularity* (one row per session, user-turn texts joined). The retrieval pipeline is LETHE's default hybrid: BM25 (FTS5) + dense ANN (sqlite-vec) over MiniLM-L6-v2 embeddings, fused with reciprocal rank fusion (Cormack et al., 2009) (no LLM hook, no reranker, single CPU). No fine-tuning or in-domain adaptation.

Table 22: LETHE v1 on LongMemEval-S at session granularity, all 500 questions. R@k = fraction of questions where the gold session is in the top-k retrieved.

Question type	N	R@5
knowledge-update	78	0.987
multi-session	133	0.940
single-session-assistant	56	0.964
single-session-preference	30	0.900
single-session-user	70	0.900
temporal-reasoning	133	0.925
Overall (micro)	500	**0.938**
		R@1=0.796, R@10=0.982

Results.

Comparison to public LongMemEval-S numbers. The originally reported strong baselines on LongMemEval-S use LLM-judged answer accuracy rather than retrieval-only R@k, so direct head-to-head is not apples-to-apples; we report R@k as the recall-axis number to demonstrate LETHE's retrieval component is not the bottleneck on the FORGETEVAL-Adv vs. Memora gap discussed in §5.4. The runner is released as `bench/longmemeval_v1.py` and takes the public `longmemeval_s.json` distribution as its argument; we do not redistribute the haystacks.

U EXTERNAL-AUTHORED SUBSET

The externally-authored cases are harder, and one of our claims does not survive them. The external cases were brought under the same three requirements as the rest of the suite (§5.3) and scored by the same rules, which pass the same gate here: an oracle scores 62/77 with 8/8 on both canonicalization categories, while *null* scores 0/77 and *nuke* 24/77 with 0/8 on both. Under them the deterministic store scores **26/77** and the mutation-time hook **35/77**, a $+11.7$ pt lift — roughly half the in-house margin. This subset exists to be harder than what we would write ourselves, and on one category it was.

identifier_obfuscation replicates the in-house asymmetry: 0/8 deterministic, 8/8 hooked. cross_lingual_identifier **does not** on our store — 0/8 deterministic and **0/8 hooked**, against 32/38 in house. The external cases give an entity three surface forms where ours give two, so the obvious

Table 23: External-authored 77-case subset (4 contributors × 20 cases each, admission-filtered). Aggregate pass rate is substantially lower than the in-house 385-case suite for every system that scores above zero in house, indicating that external authors writing under the same category schema set a tighter difficulty bar. Per-category, the placement asymmetry *replicates* on identifier_obfuscation (0/8 deterministic, 8/8 hooked on both backends) and *does not* on cross_lingual_id, where LETHE+LLM scores 0/8 and LangGraph+LLM 3/8: the third surface form here is a historical exonym, which no rule over the request string reaches (§5.5). negation_trap and compound_fact collapse to 0 % across all systems on this subset and recursive_supersession nearly so, suggesting the external authors' realisations of these categories exceed what any of our adapters can currently solve.

Cat.	LETHE	LangGraph	MEMPALACE	MEM0	LETHE+LLM	LangGraph+LLM
substr_trap	8/8	8/8	N/A	8/8	8/8	8/8
prefix_coll (n=5)	5/5	5/5	N/A	1/5	5/5	5/5
paraphr_super	6/8	6/8	N/A	5/8	6/8	6/8
neg_trap	**0/8**	**0/8**	N/A	**0/8**	**0/8**	**0/8**
temp_qual	3/8	3/8	N/A	3/8	3/8	3/8
shared_attr	3/8	2/8	N/A	2/8	4/8	4/8
compound	**0/8**	**0/8**	N/A	**0/8**	**0/8**	**0/8**
ident_obf	**0/8**	**0/8**	N/A	**1/8**	**8/8**	**8/8**
cross_ling	0/8	0/8	N/A	0/8	0/8	3/8
recursive	1/8	1/8	N/A	0/8	1/8	1/8
Overall	26/77	25/77	0/0 [77 N/A]	20/77	35/77	38/77
%	33.8	32.5	—	26.0	45.5	49.4

In-house 385 reference: LETHE 63.6, LangGraph 62.9, MEMPALACE 0.0, MEM0 65.5, LETHE+LLM 88.8, LangGraph+LLM 89.4.

reading is that a third form is past the hook's reach. Three matched conditions differing only in what the third form *is* say otherwise (§5.5): the hook scores **8/8** on two forms, **7/8** when the third is a transliteration variant, and **0/8** when it is a historical exonym (*Peking* for *Beijing*), with an oracle at 8/8 on all three. **Form count costs one case; what costs eight is a third form that shares no morphology with the others, because no rule over the request string reaches a renamed entity.** The in-house 32/38 should therefore be read as canonicalization over morphologically related forms rather than as cross-script canonicalization in general, and the boundary the external authors found is addressable: a second retrieval on the model's own alternate names would close it.

This is what the external subset is for, and it is worth being explicit that it earned its keep by contradicting us rather than by agreeing.

To address the strongest version of the circularity concern (§3.3: the 132 hand-crafted cases are written by the paper authors, who also designed the hook), we recruited four external contributors at a different research institute than the paper authors and asked each to independently write 20 adversarial cases (80 total) following our 10-category schema. Contributors received only a 6.4 KB plain-text brief specifying the case JSON shape and category definitions; the brief did *not* reveal the placement hypothesis or any system-specific results, so contributors could not optimize their cases to favour or disfavour any architecture.

We run the same Stage-1 structural admission filter as the in-house bench: **77/80 cases admitted**, 3 rejected for "self-trap" (`must_not_contain` substring inside a `must_contain` string). The admitted cases cover all 10 attack categories with 8 cases each, except prefix_collision which drops to 5 (3 rejections fell here).

Three readings. (1) On this subset the canonicalization-side asymmetry is a mutation-time effect, not a vec-only one. On identifier_obfuscation both vec-only stores score **0/8** (Letta, OpenMemory) while Letta with the mutation-time hook scores **7/8** — the same asymmetry the controlled ablation isolates in house, where Letta scores 0/38 and Letta+LLM 38/38 (Fig. 1). On cross_lingual_identifier all three score **0/8**, against Letta+LLM's 38/38 in house. The external authors' third surface form is a historical exonym, and the matched conditions of §5.5 locate the failure there rather than in the number of forms: the hook holds at 7/8 when the third form is a transliteration variant and falls to 0/8 only when the entity was renamed. This is where the subset contradicts us rather than replicating. Four independent contributors writing without knowledge of our hypothesis produced both results. **(2) Aggregate lifts on the harder external bar are positive**

Table 24: Ecosystem systems on the same 77-case external subset Every cell is strict over its own category. Letta+LLM (the same store, model at mutation time) achieves **32/77 (**41.6 %) — a +20.8-pt lift over Letta-only (16/77, 20.8 %), *exceeding* the lift the same move buys the same pair in house, +16.6 (33.0→49.6). The canonicalization gain on this subset comes from the mutation-time hook alone: on identifier_obfuscation both vec-only stores score 0/8 (Letta, OpenMemory) against 7/8 for Letta+LLM, while cross_lingual_id is 0/8 for all three — the renamed-entity limit this subset exposed, where the third surface form is a historical exonym rather than one more variant of the same name (§5.5).

Category	Letta	OpenMemory	Letta+LLM
substr	0/8	0/8	8/8
prefix	0/5	0/5	4/5
paraphr	8/8	7/8	5/8
negation	0/8	0/8	0/8
temporal	8/8	2/8	5/8
shared	0/8	0/8	0/8
compound	0/8	0/8	0/8
ident_obf	0/8	0/8	7/8
cross_ling	0/8	0/8	0/8
recursive	0/8	2/8	3/8
Overall	**16/77**	**11/77**	**32/77**

on every store we could move the model on. LETHE+LLM 45.5 % vs. LETHE 33.8 % (+11.7 pt), LangGraph+LLM 49.4 % vs. LangGraph 32.5 % (+16.9 pt), and the same move on a third-party store: **Letta+LLM 32/77 (**41.6 %**) vs. Letta 16/77 (**20.8 %**),** +20.8 **pt** — a lift that *exceeds* the +16.6 the same pair shows in house. Moving the model to the deletion path therefore transfers both across stores and onto cases the authors did not write. Inscribe-time annotation without a mutation-time hook buys nothing on this axis: A-MEM scores 0/38 and 0/38 on the two canonicalization categories in house, so it brings no canonicalization gain for the external subset to replicate. **(3) Two categories collapse to 0 % across all six systems and one nearly so.** negation_trap (8/8 universal fail), compound_fact (8/8), and recursive_supersession (7/8 universal fail) plus shared_attribute at 25–50 % reveal external authors writing materially harder cases than our in-house core: the in-house near-ceiling on negation, temporal and recursive is a saturation artifact of how our authors realised those categories, not a true ceiling.

The comparison replicates on cases we did not write. Letta scores **0/8** on identifier_obfuscation and Letta with the mutation-time hook **7/8**, on the same 77 external cases under the same scoring — the effect the controlled ablation isolates, on a store we did not build and cases we did not author. cross_lingual_identifier is 0/8 for LETHE+LLM and 3/8 for LangGraph+LLM, which is the renamed-entity limit this subset exposed: no rule over the surface form of the request reaches *Peking* from *Beijing* (§5.5), and it is a limit rather than a barrier — saying so is what the subset is for. Earlier versions of this table reported Letta and OpenMemory at 8/8 on both columns, measured before the requirements of §5.3 existed. Both stores normalise entities on write and neither deletion path consults that work, so the forbidden form is rewritten rather than removed, and scoring that asked only whether it still ranked credited the rewrite; asking whether it is still reachable puts them at 0/8 here and at 0/38 on both columns in house. The 63–66 % pass band should be read as the empirical saturation of this suite rather than as a proven upper bound.

What the external subset shows and does not show. **Shows**: (a) the *mutation-time* canonicalization advantage replicates on independently-authored cases, while inscribe-time annotation alone does not; (b) the mutation-time hook lift is positive on both backends (+11.7 LETHE, 10 fixed against 1 broken, $p = 0.012$; +16.9 LangGraph, 13 against 0, $p = 2 \times 10^{-4}$), attenuated relative to in-house (+25–27 at $p < 10^{-25}$); (c) the in-house 63–66 % deterministic pass band is provenance-dependent (the same deterministic store scores 26/77, or 33.8 %, here). **Does not show**: since two categories collapse to 0/8 universally (negation_trap, compound_fact) and recursive_supersession nearly so, we cannot distinguish among systems on those categories on this subset (no signal). And where there *is* signal, what replicates is surface-form canonicalization rather than cross-lingual identity: identifier_obfuscation goes 0/8 to 8/8 on both backends, while cross_lingual_identifier stays at 0/8 for LETHE+LLM and reaches only 3/8 for LangGraph+LLM. A larger external corpus with

calibrated difficulty would strengthen the validation; we ship the brief and protocol so external authors can continue contributing.

Failure-mode taxonomy. We attribute each all-system failure to one of three root causes using per-case inspection: **(i) Substring-scoring blind spot on negation (8 negation_trap cases)**: external authors wrote "Dana *does not have* production access" which contains "has production access" as a substring, so the affirmative `must_not_contain` string fires on the negated form; this exposes a methodology limitation of our deterministic substring scoring (orthogonal to placement claims) and motivates an NLI-aware judge in a future bench revision. **(ii) Primitive-existence confirmed (8 compound_fact cases)**: *"Henry's phone is X and email is Y"* in a single fact; `purge "Henry phone"` removes the whole fact, taking the email with it. All six systems score 0/8 here, including the two that reach 27/40 in house, so on this subset the category separates nothing (§5.5). **(iii) Over-broad-query semantic execution (7–8 recursive_supersession cases)**: external query "Iris primary browser" matches all three browser facts in a Chrome→Brave→Chrome chain; all systems honour the wide query and delete every match, leaving the expected "Chrome" `must_contain` unsatisfied (LETHE, LangGraph, LETHE+LLM, LangGraph+LLM each pass exactly 1/8 by lucky BM25 tiebreak on one chain). This is a real production failure mode (overly-broad GDPR-style purges) the in-house suite did not exercise and is a useful bench extension.

Of the 23 cases producing all-system universal failures: 15 are scoring / query-design artifacts of bench construction (i + iii), 8 confirm a known primitive-existence result (ii). Zero are counterexamples to the placement asymmetry claim itself, which remains intact on identifier_obfuscation (deterministic 0/8, LLM-hook 8/8 on both backends) and partially cross_lingual_id (LangGraph+LLM 3/8).

Per-case verdicts (admitted + rejected with reasons) are in `data/external_subset_cases.json` and `data/external_subset_results.json`.

V REAL-WORLD FAILURE MAPPING

To check whether FORGETEVAL-Adv's primitive families and attack categories correspond to the failure modes a deployer must handle, we map three classes of memory-retention failure to our taxonomy. These are *illustrative scenarios*, constructed from the obligations that regulations such as GDPR Article 17 (European Parliament and Council, 2016) place on deployers and from the failure shapes those obligations imply. They are not documented incidents: we cite no specific published incident and claim that none in particular occurred, and we run no deployment trace through our adapter. The mapping shows only that the bench surface was organised around deployment-relevant failure shapes rather than around synthetic adversarial constructions alone.

Table 25: Illustrative memory-retention failure scenarios, and the families and categories that reproduce each. These are constructed exemplars, not documented incidents.

Failure scenario	Family / categories	What the case makes concrete
Deletion acknowledged at the API, data retained in the store (the governing obligation is GDPR Art. 17 (European Parliament and Council, 2016))	purge: prefix_collision, identifier_obfuscation, cross_lingual_identifier, compound_fact	verbatim surface retention; the same entity kept under a canonicalisation variant; a reference embedded in a derived fact. The deterministic store scores 2/38, 1/38 and 0/40 on the last three where the hook scores 35/38, 32/38 and 27/40
Rotated credentials and consumed one-time codes still suggested	supersession, decay	license keys and BIC/SWIFT codes in the hand-crafted core, mirroring the credential-rotation staleness scenario
Contradictory job titles, family members or locations across sessions	drift, compound_fact	recursive_supersession tests the chain endpoint recovering an earlier-superseded value (Chrome→Brave→Chrome), which is the mode that leaks intermediates

What this mapping is and is not. This is a *structural* mapping, not an empirical one: we do not show that systems scoring higher on FORGETEVAL-Adv actually leak less in production deployments (which would require enterprise-scale memory traces we do not have). We report the mapping as a transparency exercise demonstrating that the bench's categories were designed around deployment-relevant failure shapes rather than chosen for benchmark tractability alone. An empirical correlation

study against real memory traces (enterprise CRM, agentic productivity tools, or AI assistants with public deletion logs) is the natural next step and would substantially strengthen ecological validity claims; it is left to future work.

W NAIVE-SQL BASELINE

To test whether the depth-axis reference store earns its keep over a textbook deletion baseline, we run a pure SQLite + FTS5 store with no vector recall, no LLM, and the most standard naive mutation semantics a backend engineer would reach for: BM25 top-k recall, `supersede` as delete-best-lexical-match-then-insert, `release` as lexical hard-delete, and `purge` as `DELETE WHERE text LIKE '%q%'` (substring hard-delete). This is exactly the "`DELETE FROM memories` + FTS5/BM25 reindex" comparison point, scored by the same deterministic substring scorer on the full 385-case suite.

Table 26: Naive-SQL vs. LETHE (deterministic, no LLM) on FORGETEVAL-Adv, identical scorer. Naive substring-`LIKE` deletion collapses on the precision categories: on prefix_collision it over-deletes (the `LIKE` pattern for `TXN-12345` also removes the legitimate `TXN-123456`), scoring 28 % where LETHE's BM25-precise purge reaches 82 %.

Attack category	Naive-SQL	LETHE
prefix_collision	28 %	**82 %**
paraphrase_supersession	29 %	**82 %**
shared_attribute	35 %	**88 %**
negation_trap	40 %	**95 %**
substring_trap	75 %	92 %
temporal_qualifier	100 %	100 %
recursive_supersession	92 %	92 %
identifier_obfuscation	11 %	5 %
cross_lingual_identifier	0 %	3 %
compound_fact	0 %	0 %
Overall	**156/385 (40.5 %)**	**245/385 (63.6 %)**

Naive SQL trails LETHE by **23 points overall**, with the gap concentrated in the intent-precision categories (prefix_collision -54 pt, paraphrase_supersession -53, negation_trap -55). Where the operation is lexically unambiguous (temporal_qualifier, recursive_supersession) the two tie; where it requires canonicalization (cross_lingual, compound_fact) both fail without an LLM. The naive baseline wins one column, identifier_obfuscation at 4/38 against 2/38: substring deletion clears two identifiers a precise purge leaves, the same imprecision that costs it 54 points on prefix_collision. Naive `DELETE` therefore does *not* recover LETHE's deterministic score, and the BM25-precise purge is the mechanism that prevents the substring over-deletion that sinks the naive baseline.

X SUBSTRING-SCORER RELIABILITY AUDIT

This audit was run on an earlier revision of the suite and concerns the *substring* scorer; the separate, larger defect in how `must_not_contain` was *scoped* — and the probe-based scorer that repairs it — is §5.3.

The deterministic scorer can mis-judge in two known ways: a `must_not_contain` identifier that is a substring of a legitimately-surviving longer one, and an old entity that survives only inside a past-tense clause. We quantify the rate on a 50-case stratified sample of LETHE's deterministic *failures*, re-adjudicated case-by-case (reading the actual retrieved text) by an independent Claude Opus judge into TRUE-FAIL vs. SCORER-ARTIFACT. Result: **35/50 true failures, 15/50 scorer artifacts**, all 15 confined to three families with two mechanisms between them — prefix_collision (7, the forbidden ID is a prefix of a surviving longer ID) and paraphrase_supersession with recursive_supersession (6 and 2, past-tense survival clause). Critically, the canonicalization categories that carry the LLM-placement findings — cross_lingual_identifier, identifier_obfuscation, compound_fact — contain **zero** artifacts: every failure there is a genuine canonicalization failure.

The artifacts therefore make our *deterministic* baselines conservative and do *not* inflate the LLM-hook lift, which is measured on artifact-free categories.

Full NLI-aware re-score. We implement an NLI-aware scorer — an LLM judge (GPT-5.5) that re-decides each case *semantically* from the retrievals the probe scorer reads, correcting the negation, prefix-substring and cross-script pitfalls — and re-score the whole suite with it.

Deterministic LETHE scores 245/385 under substring and **255/385** under NLI, a $+2.6$-pt conservative bias. The re-score uses its own execution of each configuration rather than the runs behind Table 1; the deterministic side is identical (245 both times, as it must be), and the hooked side lands two cases above its Table 1 run (344 vs. 342), which is inside the run-to-run spread of the LLM call and is an order of magnitude below the effect being measured. The corrections are confined to the families the sample audit predicted — prefix_collision $32\rightarrow38$ and paraphrase_supersession $31\rightarrow37$ — while identifier_obfuscation and cross_lingual_identifier move by -1 each and compound_fact not at all. The categories carrying the placement finding are untouched by the choice of scorer, which is the property that matters: the finding is not an artifact of substring matching.

The central placement result is thus *scorer-invariant*: the substring artifacts shift only the categories the audit flagged, leaving the canonicalization asymmetry — on which the inscribe-time / mutation-time placement conclusions rest — untouched. **Most importantly, the lift itself is scorer-invariant.** Re-scoring the hooked store under the same judge gives $344\rightarrow357$, against $245\rightarrow255$ deterministic, so the lift is **99 cases under substring scoring and 102 under NLI**. Three cases on a hundred-case effect is the size of the disagreement between a lexical scorer and a semantic one here; it is not the size of anything that would change a conclusion. The conservative correction is symmetric because it lands in prefix_collision and paraphrase_supersession, where the two systems behave identically; the categories where they differ are the ones NLI does not touch. A semantically-aware judge that corrects every documented substring pitfall leaves the mutation-time-hook lift unchanged.

Two notes on reading these numbers. The hooked blob capture scores 344 where the designated run scores 342; two cases is well inside the run-to-run band measured in Appendix P, and we report both rather than picking one. And the NLI judge is itself an LLM, so it is used only to *check* the deterministic scorer, never in the evaluation loop — if the two disagreed on the categories carrying the finding, that would be a reason to distrust the finding, and they do not. Per-case verdicts and the scorer are released: the verdicts in `data/nli_blobs_*_ckpt.jsonl`, the aggregates in the matching `_summary.json` files, and the scorer in `scripts/nli_scorer.py`.

Y REPRODUCIBILITY

Regeneration. Every figure, and the eighteen tables built from run data, is emitted by a script in `scripts/` from the released `data/*.json`; `scripts/verify_all.py` re-runs them and fails if any differs from what is committed, which is the check that catches a number moving without the prose following.

Hardware and keys. The four primary adapters (LETHE, MEM0, LangGraph `InMemoryStore`, MEMPALACE) run on a single CPU with no GPU and no API key. The mutation-time hook needs an OpenAI-compatible endpoint; the extended systems additionally need their own services (Docker, Postgres, Neo4j) as noted per system in Appendix R.1.

Seeds. Case generation is deterministic given a seed: the template suite is `seed=42` with four distractors per case, and the second scale point is `seed=43`. The adversarial layer is a fixed case list, not sampled. Retrieval and scoring involve no sampling; the hook runs at temperature 0.

Versions. The reference store's dependency ranges are in `pyproject.toml`. Four third-party systems are pinned at the version evaluated — MEM0 v2.0.18, MEMPALACE v3.7.1, A-MEM v0.0.1 (`agentic_memory`) and `hipporag==2.0.0a4` — and Appendix R.1 records how each extended system was deployed. Cognee v1.0.9 and Zep CE were inspected but not run. The remaining systems were installed from the then-current release and we did not record a version, which is a gap: those rows may not reproduce exactly. The hook models are named with their exact provider identifiers in Appendix P. We ship no lock file, so a rerun resolves the ranges afresh.

Table 27: Per-experiment wall-clock and LLM-call counts, read from the released run files. *LLM calls* counts calls the mutation-time hook issues; A-MEM and MEM0's router own their own clients, so their internal calls are not counted here and their wall-clock is dominated by them. The MEM0 `infer=True` row was recovered from its checkpoint after a write-guard false positive, which did not preserve timing. Generated by `scripts/repro_table.py`.

Experiment	cases	wall (s)	LLM calls
Deterministic, 385 cases	385	37	0
Mutation-time hook, 385 cases	385	394	445
LangGraph deterministic	385	35	0
LangGraph + hook	385	453	445
MEM0 `infer=False`	385	1533	0
MEM0 `infer=True`	385	—	—
MEMPALACE	385	247	0
A-MEM	385	10068	0
Annotation arm	385	1828	999
Readable annotation arm	385	2033	999
Merge-authority arm	385	990	718
Both arms	385	2720	1442
External subset, deterministic	77	8	0
External subset, hook	77	373	77

Cost. Wall-clock and LLM-call counts as measured, one row per experiment the paper reports:

Release. The benchmark, all adapters, the LETHE reference store, and the generation scripts are released under MIT at `https://anonymous.4open.science/r/forgeteval-anon-supp-0C84`, an anonymized mirror for review; the camera-ready will replace it with the public repository.