Retrieval Is Not Memory

Memory as a Governance Function over Experience

Perslis Research

2026-09-15

PROTOTYPE / DRAFT — unpublished. This is a working draft, not a peer-reviewed paper. Formal claims are stated as design-level propositions, not proved theorems. The reference systems in §5 are labeled PROTOTYPE / PILOT and are not enterprise-hardened. External citations were verified against primary sources on 2026-09-15.

Abstract

The barrier to shipping a system marketed as "AI memory" has collapsed to a weekend: embed some text, store the vectors, and return the top-k nearest neighbors of a query. We argue this near-universal pattern is a category error. A vector database with top-k cosine retrieval answers exactly one question — *what stored text is embedding-similar to my query?* — and none of the questions a memory system actually has to answer. We define **memory** not as a retrieval function over embeddings but as a **governance function over experience**: the discipline of deciding what becomes durable, what decays, what reactivates, how contradictions resolve, how differently-typed knowledge is stored and updated, what consolidates, what is forgotten, what is relevant despite surface dissimilarity, and what must be *suppressed* even when retrieved. Retrieval is one operation inside this function, not the function itself; formally, retrieval $\subset$ memory. A system that only performs nearest-neighbor lookup has *storage*, not memory. We formalize nine governance decisions as the open research questions the field has skipped, give a typed schema (fact / episode / procedure / working-state) with distinct persistence, decay, retrieval, and update semantics, and offer a constructive existence proof: a working stack (LIST, an offline consolidation layer, the jeeves-floor symbolic layer, and an evolution-memory store) that enforces several of these governance properties today, at PROTOTYPE/PILOT maturity. We connect memory governance to execution governance — a retrieved memory that shapes behavior is an *admission* decision — and close with an honest accounting of which decisions the reference architecture answers and which remain open.

Keywords: AI memory, retrieval-augmented generation, memory consolidation, knowledge governance, neuro-symbolic systems, forgetting, provenance.

1 Introduction — Retrieval Is Not Memory

Something quiet happened to the word *memory*. Two years ago, giving a language model persistent memory was a hard, open problem. Today it is a tutorial: chunk your documents, run them through an embedding model, put the vectors in a store, and at query time return the k chunks whose embeddings are closest to the query's embedding. This is a genuinely useful pattern — retrieval-augmented generation (RAG) [1] — and its accessibility is a real achievement. But its accessibility

is also how the word got redefined downward. Because the mechanism is easy, "memory" came to mean "whatever a vector database does." The field now routinely labels a similarity index a *memory system.*

This is a category error, and it is worth stating precisely rather than as a complaint. Nearest-neighbor retrieval over embeddings answers one question:

> *Which stored items are embedding-similar to the current query?*

Memory, as any account of memory in biological or engineered systems must, has to answer a different and larger set of questions: which experiences should persist and which should vanish; when an old memory should fade and whether fading is the same as deletion; when a faded memory should surge back because a situation recurred; how two memories that disagree should be reconciled; whether a given item is a timeless fact, a timestamped event, a reinforced procedure, or ephemeral scratch state — because those demand different treatment; what many raw observations should be distilled into; what ought to be forgotten by design; which memory is relevant to the task *despite* being surface-dissimilar; and, critically, when a retrieved memory should be **prevented** from influencing behavior at all.

None of these are similarity questions. A similarity index is silent on every one of them. It is, precisely, **storage with fuzzy keys** — and storage is not memory. The distinction is not pedantic: storage *remembers everything and knows nothing.* It has no opinion about what mattered, no mechanism to let the past correct itself, and no gate between recalling something and acting on it.

Thesis. *Memory is not a retrieval function over embeddings; it is a governance function over experience. Retrieval is one operation inside memory, not memory itself.* We write this as the containment relation **retrieval** $\subset$ **memory** and spend the paper unpacking the part of memory that is not retrieval.

Contributions.

1. **A precise statement of the category error** (§2): a definition of memory as governance over experience, contrasted with retrieval over embeddings, and the containment relation retrieval $\subset$ memory.
2. **A decomposition of memory into nine governance decisions** (§3), each elevated from an engineering afterthought to a first-class research question, with a typed schema (fact / episode / procedure / working-state) that most vector stores collapse into one flat namespace.
3. **A diagnosis of why the field misses this** (§4): the incentive gradient (RAG ships in a weekend; benchmarks reward retrieval@k, not governance), the embedding monoculture, and the conflation of storage with memory.
4. **A constructive existence proof** (§5): a working stack that enforces several governance properties today — time-bounded typed observation indexing, offline consolidation, symbolic contradiction resolution with functional-relation hard-vetoes, and a durable lesson store that informs but never authorizes — described at the property level and labeled honestly by maturity.
5. **A bridge to execution governance** (§6): a retrieved memory that shapes behavior is an *admission* decision, subject to the same provenance, fail-closed gating, and contradiction checks that govern actions [2].

We are deliberately narrow about what §5 proves. It does not show that any single system solves memory. It shows that memory-as-governance is *buildable* — that the nine decisions are implementable properties, not philosophy — because a running stack already enforces several of them.

2 The Category Error

2.1 Two definitions

Definition 1 (Retrieval over embeddings). Let E be an embedding function mapping items to vectors in $\mathbb{R}^n$, S a set of stored items, and q a query. *Retrieval* returns argtop-k over S of some similarity $sim(E(q), E(x))$, typically cosine similarity. Retrieval is a pure function of the current query and the current contents of S. It has no state of its own, no notion of time beyond what is encoded in the items, and no output other than a ranked list.

Definition 2 (Memory as governance over experience). A *memory system* is a set of policies that govern the life-cycle of experience: **admission** (what enters durable store and how strongly), **decay** (how salience diminishes with time and disuse), **reactivation** (how a dormant item is restored when its context recurs), **reconciliation** (how contradictory items resolve), **typing** (how differently-natured knowledge is stored and updated), **consolidation** (how many observations become a durable generalization), **forgetting** (what is removed by design), **relevance weighting** (what should influence the task, similarity aside), and **suppression** (what is withheld from influence even when recalled). Retrieval is the read primitive these policies wrap; it is necessary and radically insufficient.

Proposition 1 (Containment). *Retrieval* $\subset$ *memory.* Every governance policy in Definition 2 either produces the store retrieval reads, constrains what retrieval may return, or overrides retrieval's output before it acts. Retrieval implements none of them. A system offering only Definition 1 therefore implements a strict subset of memory; we call it **storage**.

This is a design-level proposition, not a theorem: it asserts a functional decomposition, and its force is definitional. But the decomposition is falsifiable in a useful sense — if a pure similarity index could be shown to make any of the nine decisions in §3 correctly *without additional policy*, the containment would be an equality and our thesis wrong. We argue in §3 that each decision requires state, semantics, or authority that similarity does not carry.

2.2 Why "fuzzy keys" is the whole of it

A vector store's contribution over a hash map is that its keys are fuzzy: you can look up by approximate meaning instead of exact string. That is valuable and it is *all* it adds. Fuzzy-keyed storage still cannot tell you that an item is stale, that it was superseded last week, that it belongs to a different tenant, that it is a one-off episode rather than a standing fact, or that recalling it is not permission to use it. Those are governance facts about the item, and they live outside the geometry of the embedding space. Cosine distance measures textual-semantic proximity; it does not measure recency, authority, provenance, applicability, or trust. Treating proximity as a proxy for all of these is the error the rest of the paper anatomizes.

3 The Decisions a Memory System Must Make

We now state the nine decisions. Each is a question the field's default architecture cannot answer, phrased as a research question and paired with the governance policy it demands. Table 1 (§3.10) summarizes them.

3.1 Durability and differential encoding

> ***Q1.*** *Why should experience A become durable while experience B disappears?*

Not all experience deserves equal persistence, yet flat vector stores encode every ingested chunk with equal permanence and equal weight. A memory system must make a *differential encoding* decision at admission. The signals that should drive it are well studied outside AI: **salience** (how much the experience matters to current goals), **surprise / prediction-error** (the experience violated an expectation — the classic learning signal), **consequence or reward** (the experience was followed by a significant outcome), **repetition** (it recurred), and **explicit tagging** (a human or a trusted process marked it as important). A system that weights admission by these signals remembers the consequential and lets the noise wash out; a system that admits everything at uniform weight has abdicated the decision.

Claim 3.1. Uniform-weight admission is not a neutral default; it is a decision to treat all experience as equally worth keeping, which is almost never true and which no similarity score can correct after the fact.

3.2 Decay

> ***Q2.*** *When should an old memory decay — and is decay the same as deletion?*

Human memory follows a forgetting curve: retention falls with elapsed time absent reinforcement [3]. An engineered memory needs an analogous decay policy driven by **age**, **disuse** (not retrieved when it could have been), and **supersession** (a newer item covers the same ground). The critical design point: **decay ≠ deletion.** Decay should be *graceful and reversible* — a diminishing weight that lowers an item's default influence and retrievability without destroying it, so that reactivation (§3.3) remains possible. A store that only ever appends has no decay; a store that hard-deletes on a timer conflates decay with forgetting (§3.7) and forecloses reactivation.

3.3 Reactivation and reconsolidation

> ***Q3.*** *When should a decayed memory become important again because a recurring situation returned?*

This is the decision pure similarity retrieval misses *worst*. A memory that has decayed into the background is, by construction, ranked low by a similarity index — its low weight is exactly what a top-k query discards. But the signal that should restore it is not "the query resembles this item"; it is **"the situation this memory belongs to has recurred."** Context-triggered reactivation keys off entity, task, temporal, or environmental cues — *this project, this counterparty, this error, this time of month* — and surfaces the dormant memory precisely because the *context* matches, even

when the *query text* does not. Biological memory reconsolidates on reactivation: a recalled memory becomes labile and is re-stored, updated by the current context [4]. An engineered analogue restores the item's weight and merges any new context into it. A similarity index has no notion of "the situation came back"; it re-ranks text, not situations.

Claim 3.3. The most valuable reactivations are the ones a similarity query suppresses — the faded memory whose *context* recurred but whose *surface form* does not match the current query.

3.4 Contradiction

> ***Q4.*** *How should contradictory memories interact?*

When two stored items disagree, a memory system must decide among **recency** (the newer wins), **authority** (the more trusted source wins), **provenance** (the better-evidenced wins), **supersession** (the new one replaces the old), and **coexistence** (both are kept — because they are not actually in conflict). The last point is where typing (§3.5) becomes load-bearing: a contradicted **fact** is a genuine conflict that must resolve (a person cannot both live in Boston and not live in Boston), but two differing **episodes** are not a contradiction at all (they attended a meeting on Monday *and* a different meeting on Tuesday). A flat vector store cannot tell these apart; it will happily return both contradictory facts side by side and let the generator average them, which is how confident-sounding contradictions reach the user. Contradiction resolution needs a layer that (a) knows which relations are *functional* — can hold at most one value — and (b) can veto a write that violates one. Embedding geometry does not encode functionality; "lives in Boston" and "lives in Seattle" are *near* each other in embedding space, which is exactly backwards for detecting that they conflict. A symbolic layer with typed relations resolves what embeddings cannot (§5.3).

Claim 3.4. Detecting a factual contradiction requires knowing a relation is functional; functionality is a symbolic property, not a metric one, and near-neighbor items in embedding space are frequently *contradictory* rather than *consistent*.

3.5 Typing

> ***Q5.*** *What is the difference between a fact, an episode, a procedure, and current working state — and why does conflating them poison the store?*

We distinguish four types, each with different persistence, decay, retrieval, and update semantics (Table 2, §3.10):

- **Fact** — a timeless assertion (*the capital of France is Paris*; *this customer's tier is Gold*). Long persistence; decays only by supersession; retrieved by entity/relation; updated by *replacement* under contradiction rules.
- **Episode** — a timestamped event (*at 14:03 the deploy failed*). Persistence bounded by relevance window; decays by age; retrieved by time/context; **never overwritten** — a later episode does not contradict an earlier one, it follows it.
- **Procedure** — a reinforced how-to (*to roll back, do X then Y*). Persistence grows with successful reuse; decays by disuse and by supersession when a better procedure is learned; retrieved by *applicability* to the current task; updated by reinforcement.

- **Working state** — ephemeral, task-scoped scratch (*the variable we are currently tracking*). Must **not** leak into durable memory; retrieved only within the task; discarded at task end (§3.7).

The root error of the flat vector store is putting all four in one namespace with one persistence policy. A fact and a scratch note get identical treatment; an episode gets overwritten as if it were a fact; working state leaks into the durable corpus and poisons future retrieval with the residue of finished tasks. Typing is not an optimization; it is the precondition for every other decision in §3 being *correct* rather than uniform.

Claim 3.5. The four types have mutually incompatible correct policies; any single flat policy is wrong for at least three of them.

3.6 Consolidation

> ***Q6.*** *What gets consolidated, and when?*

Raw experience is verbose and redundant; durable memory should be compact and general. Consolidation is the decision to distill many low-level items into fewer high-level ones: **episodes → facts or procedures** (twenty observations of "deploy failed after config change" become the standing fact "config changes are high-risk" and the procedure "validate config before deploy"), and **many observations → a durable generalization**. The natural implementation is an *offline digest pass*: periodically, away from the request path, read the raw episodic buffer and write consolidated, higher-typed memory — the engineering analogue of sleep-dependent memory consolidation, where the hippocampal record is replayed and integrated into neocortical structure (the complementary-learning-systems account) [5]. Consolidation is also where noise is dropped rather than distilled, linking it to forgetting (§3.7). A store that only appends at ingestion time and never runs an offline pass never consolidates; it accumulates.

3.7 Forgetting

> ***Q7.*** *What should be forgotten — as a designed feature?*

Forgetting is not failure; it is function. A memory system should deliberately discard: **working state after its task completes** (§3.5), **superseded facts** once resolution has picked a winner (§3.4), **secrets and PII** that should never have durably persisted or whose retention window has expired, and **noise** that consolidation judged not worth distilling. There are two distinct forgetting regimes and both matter: *hygienic forgetting* (keeping the store clean and relevant) and *governance forgetting* (honoring deletion requests, retention limits, and the right to be forgotten). The second is a hard requirement in regulated domains and is impossible in an append-only vector store that treats deletion as an afterthought. Designing forgetting up front — deciding what *must* leave and proving it left — is a memory function, not a maintenance chore.

Claim 3.7. A memory system that cannot forget cannot be compliant, cannot stay relevant, and cannot prevent finished-task residue from degrading future retrieval.

3.8 Relevance beyond similarity

> ***Q8.*** *Which memories should affect the current task despite poor embedding similarity?*

Similarity is a *weak proxy* for relevance. The memory that most needs to fire is often surface-dissimilar to the query. Relevance has dimensions embedding distance does not capture: **causal relevance** (this earlier event *caused* the condition you are now in), **procedural applicability** (this how-to applies to your task even though its text shares few tokens with your query), **entity and temporal links** (same actor, same object, same window), and the blunt but vital **"this bit me before"** signal (a past failure in a structurally analogous situation). None of these is monotonic in cosine distance; some are *anti-correlated* with it. A relevance policy must combine similarity with these signals, not defer to it. Ranking memory by embedding proximity alone systematically discards the causally and procedurally relevant in favor of the merely lexically alike.

Claim 3.8. For a large class of tasks, the highest-relevance memory is not in the top-k by cosine similarity; retrieval@k by embedding therefore has a relevance ceiling below what the store contains.

3.9 Suppression and gating

> ***Q9.*** *When should a retrieved memory be prevented from influencing behavior?*

This is the decision that most sharply separates memory from storage. Retrieval finding an item is not permission to use it. A retrieved memory must pass a **gate** before it may shape behavior, and the gate must be able to *deny*: the item is **stale** or **superseded**; it is **out of scope** for the current task; it belongs to the **wrong tenant** or user; it is **unverified or low-trust**; it is **security-sensitive** and must not enter this context; or it has been **contradicted** by a more authoritative item (§3.4). We state the governing principle as a proposition:

Proposition 2 (Retrieval is not authorization). *That an item is retrievable does not entail that it may influence behavior. Between recall and use there must be a gate whose default, under ambiguity, is to deny (fail-closed).*

Suppression is precisely the point where memory governance meets execution governance (§6): denying a retrieved memory the right to act is the same admission decision as denying a tool call the right to execute. A pure retrieval system has no gate — whatever it returns flows straight into the prompt and thence into behavior — which is why "the model confidently used a stale/leaked/wrong-tenant fact" is a *memory-architecture* failure, not a model failure.

3.10 Summary tables

4 Why the Field Misses This

The category error is not a failure of intelligence; it is a response to an incentive gradient. Naming the incentives is more useful than deriding the outcome.

4.1 RAG ships in a weekend. The default architecture is popular because it is cheap to build and immediately demoable. Embedding APIs, vector stores, and top-k retrieval are commoditized; wiring them together is an afternoon. Governance — typing, decay, consolidation, gating — is real engineering with no turnkey library. The path of least resistance produces storage, and storage demos well enough to be called memory.

4.2 Benchmarks reward retrieval@k, not governance. The metrics the field optimizes — recall@k, MRR, nDCG, answer accuracy over a static corpus — measure *whether the relevant chunk*

Table 1: The nine governance decisions.

#	Decision	Question	Policy signal(s)	What similarity alone cannot do
Q1	Durability / differential encoding	What becomes durable?	salience, surprise, consequence, repetition, tagging	weight admission by importance
Q2	Decay	When does it fade?	age, disuse, supersession	lower influence without deletion
Q3	Reactivation / reconsolidation	When does it return?	context recurrence (entity/task/time)	surface a low-ranked item because the *situation* recurred
Q4	Contradiction	How do conflicts resolve?	recency, authority, provenance, supersession	know a relation is functional; veto a conflicting write
Q5	Typing	Fact vs episode vs procedure vs working-state?	item nature	apply four incompatible policies in one namespace
Q6	Consolidation	What distills into durable form?	offline digest, generalization	turn many episodes into a fact/procedure
Q7	Forgetting	What must leave?	task-end, supersession, PII, noise, deletion requests	remove by design and prove removal
Q8	Relevance beyond similarity	What matters despite dissimilarity?	causal, procedural, entity/temporal, "bit me before"	rank by relevance, not proximity
Q9	Suppression / gating	What must not act, even when recalled?	staleness, scope, tenant, trust, sensitivity, contradiction	deny influence; fail closed under ambiguity

Table 2: Typed memory schema — distinct semantics per type.

Type	Persistence	Decay	Retrieval key	Update semantics
Fact	long; timeless	by supersession only	entity / relation	replacement under contradiction rules
Episode	bounded by relevance window	by age	time / context	append-only; never overwritten
Procedure	grows with successful reuse	by disuse; supersession by better how-to	applicability to task	reinforcement
Working state	task-scoped only	at task end (discard)	within-task only	mutated freely; must not leak to durable store

was fetched, not whether the system decided correctly what to keep, what to forget, which contradiction to resolve, or what to suppress. This is Goodhart's law operating on memory research [6]: when retrieval@k becomes the target, it stops being a measure of memory and becomes a measure of retrieval, and systems are built to maximize it. None of the nine decisions in §3 has a standard benchmark; several (forgetting, suppression) are actively *penalized* by retrieval metrics, which reward returning more, not withholding correctly. A field measures what it can measure and then mistakes the measure for the thing.

4.3 The embedding monoculture. Once the embedding-plus-vector-store stack became the default substrate, it became the default *ontology*. Problems get reshaped to fit the tool: if the only operation is nearest-neighbor over a flat namespace, then contradiction becomes "return both," typing becomes "one collection," decay becomes "nothing," and suppression becomes "trust the top-k." The monoculture is self-reinforcing because the tooling, tutorials, and hiring all assume it.

4.4 Conflating storage with memory. Underlying all of the above is the vocabulary slip we opened with. Because a vector store *persists* things and *returns* them later, it feels like memory, and the word gets applied. But persistence and recall are storage. Memory is the *governance* around persistence and recall — the deciding. The field skipped the deciding because the substrate had no place to put it.

4.5 The absence of a typed schema. Finally, the flat namespace is not just a performance choice; it is the structural cause of most of the failures above. Without types (§3.5), there is nowhere to hang differential persistence, no way to say "this is an episode, never overwrite it" or "this is working state, never persist it," and no basis for type-specific decay or update rules. The missing schema is why the other eight decisions have nowhere to live.

5 A Memory That Governs, Not Just Retrieves — Reference Architecture

We now give the existence proof. The claim is narrow and we hold to it: memory-as-governance is *buildable*, because a working stack already enforces several of the §3 properties. We describe the systems at the **property level only** — the enforced behaviors, not internal module names, code, or recipes — and we label each by maturity. None of these components is enterprise-hardened; they are PROTOTYPE/PILOT. What they demonstrate is that the governance decisions are implementable, not that any one of them is a finished product.

5.1 LIST — time-bounded, typed observation indexing (addresses Q3, Q5, Q8)

LIST is an observation index whose read interface is **time-bounded and typed by construction**, not a flat similarity blob. A read for recent observations takes an explicit time lower bound (defaulting to a recent window) and a per-project scope, and returns a bounded set of typed, timestamped records. Two governance properties follow directly from this shape. First, **relevance is temporal and scoped, not only semantic**: the primary retrieval key is *when* and *in what project*, which is exactly the context-recurrence signal §3.3 and §3.8 argue similarity discards. Second, the interface is declared **read-only and non-executing** — the records are explicitly advisory context that performs no capture, no model calls, no writes, and no execution. Recalling an observation through LIST cannot, by construction of the interface, cause anything to happen. That is the suppression/gating stance of §3.9 built into the read primitive: memory hands you

context, never authority. *Maturity: PILOT.*

5.2 Offline consolidation layer — digests over raw observations (addresses Q6, Q7)

Above the raw observation stream sits an **offline consolidation pass** that reads many low-level observations and produces durable, higher-level summaries — the episodes-to-generalization move of §3.6 — with a narrative layer generated on top of the consolidated record rather than over the raw stream. This is the sleep-consolidation analogy made concrete: an away-from-the-request-path digest that distills, and in distilling drops the noise it judges not worth keeping (the hygienic-forgetting half of §3.7). The consolidated summary, not the raw firehose, is what later reads see by default. *Maturity: PROTOTYPE.* We do not claim the consolidation policy is optimal or that its drop decisions are audited to a governance standard; we claim the offline distill-and-drop pass exists and produces durable generalizations from raw episodes.

5.3 jeeves-floor symbolic layer — typed relations, functional hard-vetoes, contradiction trichotomy (addresses Q4, Q9)

The jeeves-floor layer is a typed symbolic store that does what embedding geometry cannot: resolve contradictions and veto conflicting writes. It carries typed relations, a subset of which are marked **functional** (can hold at most one value for a given subject, in the OWL FunctionalProperty sense) [7]. A contradiction check exposes an explicit **trichotomy** — *consistent*, *contradiction*, or *unknown* — and when a claim contradicts a functional relation already in the store, the verdict is *contradiction* with a hard **veto**. Two properties matter for §3.4 and §3.9. First, the layer resolves factual conflict *symbolically*: it detects that "lives in X" and "lives in Y" conflict because the relation is functional, which no cosine distance encodes. Second — and this is the honesty that makes it a memory rather than a guesser — the third verdict is **unknown**: when the graph contains no checkable relation, the layer **abstains** rather than fabricating a ruling. We confirmed this live: a claim whose relation was not grounded in the graph returned *unknown / veto=false* with the explicit reason "no checkable relation found," rather than guessing. Abstention-under-ignorance is itself a governance property; a system that always answers has no notion of the boundary of its own knowledge. The check is read-only and never writes. *Maturity: PROTOTYPE/PILOT.*

5.4 Evolution memory — durable lessons vs obstacles, informs but never authorizes (addresses Q1, Q5, Q9)

The evolution-memory store keeps two durable, typed records of past self-improvement attempts: **lessons** (what was proven to work) and **obstacles** (what was proposed and rejected). This is differential typed encoding (§3.1, §3.5): proven and rejected experience are stored distinctly, not as undifferentiated text. The load-bearing governance property is what the store is *forbidden* to do: it is **retrieval-only and is never a verdict authority**. A lesson can *inform* a later decision; it can never, by itself, *authorize* a promotion or an action — that authority lives in a separate, out-of-loop gate. This is Proposition 2 enforced at the architecture seam: memory contributes evidence to a decision, but a distinct authority makes the decision. It is also the direct link to §6: the same principle that keeps a lesson from authorizing a promotion is the principle that keeps a retrieved memory from authorizing behavior. *Maturity: PROTOTYPE.*

5.5 What the existence proof does and does not show

Table 3: Reference architecture coverage of the nine decisions.

Decision	Component(s)	Coverage	Maturity
Q1 durability	Evolution memory (typed lesson/obstacle)	Partial — typed differential encoding; not a general salience/surprise model	PROTOTYPE
Q2 decay	—	**Open** — no graded reversible decay across the stack	—
Q3 reactivation	LIST (temporal/scoped retrieval)	Partial — context-keyed retrieval; no automatic reconsolidation-on-recall	PILOT
Q4 contradiction	jeeves-floor (functional veto, trichotomy)	Substantial — symbolic conflict resolution + abstention	PROTOTYPE/PILOT
Q5 typing	LIST + jeeves-floor + evolution memory	Partial — typed at several layers; no single unified fact/episode/procedure/working-state schema	mixed
Q6 consolidation	Offline digest layer	Partial — episodes→generalization + noise drop; policy not audited	PROTOTYPE
Q7 forgetting	Offline digest (hygienic)	Partial — noise dropped; governance/right-to-delete forgetting **not** demonstrated	PROTOTYPE
Q8 relevance beyond similarity	LIST (temporal/scoped)	Partial — temporal+scope beat pure cosine; causal/procedural relevance not modeled	PILOT
Q9 suppression / gating	LIST (advisory-only), evolution memory (informs≠authorizes), jeeves-floor (veto)	Substantial — three independent enforcement points	PROTOTYPE/PILOT

The honest reading of Table 3: the stack answers Q4 and Q9 substantially, several decisions partially, and leaves **graded reversible decay (Q2)**, **automatic reconsolidation-on-recall (Q3)**, a **unified typed schema (Q5)**, and **governance-grade forgetting (Q7)** open. That is the correct shape for an existence proof: it establishes that the properties are buildable and that some are built, without pretending the problem is solved.

6 Memory Governance Is Execution Governance

The suppression decision (§3.9) is not a special case; it is the general theory of the whole paper viewed from one angle. A companion paper, *The Orchestration Gap* [2], argues that in a multi-model runtime, safety is a property of the orchestrator and its tools — not of any model — so control must attach to the execution chain, with four invariants: out-of-loop verification, signed

provenance, fail-closed admission, and contradiction checking.

Those are the same invariants a memory system needs. **A retrieved memory that influences behavior is an admission decision.** When a recalled item flows into a prompt and shapes what the system does next, it has been *admitted* to the execution context exactly as a tool call is admitted to execution. The mapping is one-to-one:

- **Provenance.** A memory should carry where it came from and how it was evidenced, so the gate can weigh authority (§3.4) — the same signed provenance the execution chain requires.
- **Fail-closed admission.** Under ambiguity — stale? wrong tenant? unverified? — the default is to *withhold* the memory from influence, not to include it (Proposition 2) — the same fail-closed default the execution chain requires.
- **Contradiction checking.** Before a memory acts, it should be checked against more authoritative memory and vetoed if it conflicts (§3.4, §5.3) — the same contradiction invariant the execution chain requires.
- **Out-of-loop authority.** Memory informs; a distinct authority decides (§5.4) — the same referee-is-not-a-player principle the execution chain requires.

So §3.9 is the admission gate of *The Orchestration Gap* applied to memory. This is not an analogy of convenience; it is the same gate. A system that governs which *actions* may execute but lets *any retrieved memory* flow unfiltered into the context that drives those actions has left the gate open on the input side. Memory governance and execution governance are one problem with two faces.

7 Open Problems / Research Agenda

Restating the nine decisions as the agenda the field must take up, with an honest note on what §5 answers:

1. **Differential encoding (Q1).** A principled, tunable salience/surprise/consequence model for admission weighting. *Partially addressed* (typed lesson/obstacle); a general model is open.
2. **Graded reversible decay (Q2).** Decay that lowers influence and retrievability without deletion, and that reactivation can undo. *Open* in the reference stack.
3. **Reconsolidation-on-recall (Q3).** Context-recurrence triggers that restore dormant memory and merge new context on access. *Partially addressed* (temporal/scoped retrieval); automatic reconsolidation is open.
4. **Contradiction resolution (Q4).** Recency/authority/provenance policies plus functional hard-vetoes, with the fact-vs-episode distinction. *Substantially addressed* (§5.3); generalizing beyond the modeled relations is open.
5. **Unified typed schema (Q5).** A single fact/episode/procedure/working-state schema with per-type persistence, decay, retrieval, and update semantics. *Partially addressed* across layers; a unified schema is open.

6. **Consolidation (Q6).** Auditable offline distillation of episodes into facts and procedures. *Partially addressed* (§5.2); policy quality and auditability are open.
7. **Governance-grade forgetting (Q7).** Provable deletion for right-to-delete, retention limits, and secret/PII expiry. Hygienic forgetting is *partially addressed*; governance-grade forgetting is **open and important**.
8. **Relevance beyond similarity (Q8).** Causal, procedural, and entity/temporal relevance combined with, not subordinate to, embedding similarity. *Partially addressed* (temporal/scope); causal/procedural relevance is open.
9. **Suppression / gating (Q9).** A fail-closed admission gate between recall and use. *Substantially addressed* (§5.4, §5.1, §5.3, §6); a unified, auditable memory-admission gate is the natural next build.

A cross-cutting open problem: **evaluation.** The field needs benchmarks that score governance — correct forgetting, correct suppression, correct contradiction resolution, correct reactivation — not just retrieval@k. Until such benchmarks exist, §4.2's Goodhart dynamic will keep steering effort toward retrieval and away from memory.

8 Limitations and Conclusion

8.1 Limitations

- **The central propositions are design-level, not proved theorems.** Proposition 1 (retrieval $\subset$ memory) and Proposition 2 (retrieval is not authorization) are functional decompositions with definitional force; they are falsifiable in the sense of §2.1 but are not formal proofs.
- **The reference systems are PROTOTYPE/PILOT, not enterprise-hardened.** §5 is an existence proof of buildability, not a claim of production maturity, scale, or independent evaluation. We have not run controlled comparisons against baseline vector-store memory, and we report no quantitative benchmark of governance quality — because, per §7, the benchmarks do not yet exist. The two live confirmations we do report (the jeeves-floor *unknown*-verdict abstention; the LIST advisory-only, read-only, time-bounded interface) are single-call property checks, not statistical evaluations.
- **§5 is intentionally property-level.** We describe enforced behaviors, not internal designs, so this draft cannot be used to reconstruct the systems. That is deliberate and it does limit the reproducibility of the existence proof to the property claims, which are independently checkable through the systems' public interfaces.
- **Several decisions are unaddressed by the reference stack** — graded reversible decay, automatic reconsolidation, a unified typed schema, and governance-grade forgetting (Table 3). We do not claim to have solved memory. We claim the field has mis-defined it and that a large, tractable, mostly-unbuilt agenda follows from defining it correctly.
- **The critique targets the dominant pattern, not every system.** Some research systems do implement pieces of §3 (memory-augmented architectures, knowledge-graph memories, spaced-repetition schedulers). Our claim is about the *marketed default* — vector store plus top-k retrieval sold as "AI memory" — not that no one anywhere governs memory.

8.2 Conclusion

The barrier to entry collapsed, and in collapsing it quietly redefined *memory* down to whatever a vector database does. That redefinition is a category error with a precise shape: it mistakes storage — persistence plus fuzzy-keyed recall — for memory, which is the governance around persistence and recall. Retrieval is one operation inside memory, not memory itself. The nine decisions in §3 — durability, decay, reactivation, contradiction, typing, consolidation, forgetting, relevance, suppression — are the part of memory that is not retrieval, and they are, individually and demonstrably, buildable: a working stack already enforces several of them at the property level.

The field will keep shipping storage and calling it memory as long as the benchmarks reward retrieval@k and the substrate has nowhere to put the deciding. The correction is not a better embedding model; it is a typed schema and a set of governance policies wrapped around retrieval, and a gate between recalling a memory and letting it act.

> **Storage remembers everything and knows nothing. Memory is the discipline of deciding.**

References

[1] P. Lewis, E. Perez, A. Piktus, et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2005.11401.

[2] Perslis Research. "The Orchestration Gap: Why Model-Level Alignment Cannot Survive Multi-Model Runtimes." Preprint, 2026. https://research.perslis.com/orchestration-gap.html

[3] H. Ebbinghaus. *Memory: A Contribution to Experimental Psychology.* 1885 (Eng. trans. Ruger & Bussenius, 1913). Origin of the forgetting curve.

[4] K. Nader, G. E. Schafe, & J. E. LeDoux. "Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval." *Nature* 406:722–726, 2000. Memory reconsolidation on reactivation.

[5] J. L. McClelland, B. L. McNaughton, & R. C. O'Reilly. "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." *Psychological Review* 102(3):419–457, 1995. Hippocampal–neocortical consolidation.

[6] C. A. E. Goodhart. "Problems of Monetary Management: The U.K. Experience." In *Papers in Monetary Economics*, Reserve Bank of Australia, 1975. Popularized by M. Strathern (1997) as "when a measure becomes a target, it ceases to be a good measure."

[7] W3C. "OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)." W3C Recommendation. `owl:FunctionalProperty` — at most one value per individual.

[8] E. Tulving. "Episodic and Semantic Memory." In E. Tulving & W. Donaldson (Eds.), *Organization of Memory*, pp. 381–403. Academic Press, 1972. The episodic/semantic distinction underlying the episode/fact typing of §3.5.

[9] R. M. French. "Catastrophic forgetting in connectionist networks." *Trends in Cognitive Sciences* 3(4):128–135, 1999. Why naive continual updates destroy prior memory (Q1/Q2).

[10] A. d'Avila Garcez & L. C. Lamb. "Neurosymbolic AI: the 3rd wave." *Artificial Intelligence Review* 56(11):12387–12406, 2023. Symbolic layers constraining neural generators (§5.3).

Additional relevant prior art (not exhaustively surveyed): spaced-repetition scheduling (Leitner; the SuperMemo SM-2 algorithm) for Q2/Q6; and the knowledge-graph and memory-augmented neural-network literature as non-flat memory architectures.

Perslis Research — PROTOTYPE / DRAFT, unpublished. Not peer-review-ready.