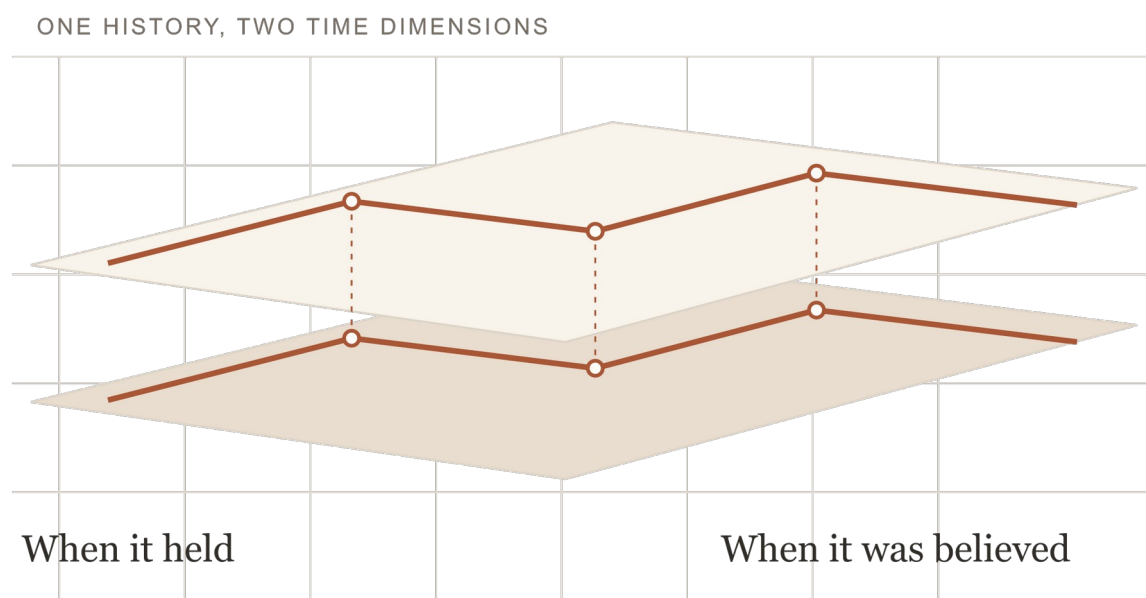


Lashley

Memory for changing facts

A bi-temporal architecture for long-lived agents



Event time and belief time are distinct dimensions of a recorded history.

Facts change.

A memory system must know how.

Update semantics, executable checks and preliminary evaluation of a memory architecture that separates what held in the world from what the system believed.

OMVY · AI Research & Product Studio

Report edition 7 October 2026 / Evaluation snapshot 18 August 2026

Preliminary technical report · Not peer reviewed

A memory needs an update model

Remembering a statement is not the same as knowing whether it still holds.

Long-lived agents encounter changes in roles, locations, commitments and claims. Retrieving an old passage can recover a fact accurately while still giving the wrong answer today. A useful memory therefore needs explicit rules for accepting new information, retiring superseded beliefs and preserving the history that remains relevant.

Lashley combines a bi-temporal relation store with episodic traces, prospective records and scheduled maintenance. This report concentrates on its most inspectable technical contribution: the connection between update intent, temporal postconditions and executable checks. It also introduces TULVING, a synthetic instrument for evaluating memory over changing facts.

Three contributions

Update semantics. Five outcomes separate a change in the world from a correction of the record, a disputed observation, a deferred commitment and a cancellation. The evaluated passive-ingest path routes four; full retroactive correction requires explicit temporal authority.

Scoped verification. Seeded programs exercise public write operations. Edge-level integrity, minimal incidental retirement and per-operation fragment bounds are checked separately from functional-slot enforcement and concurrency.

An evaluation instrument. Deterministic worlds and gold timelines make temporal questions testable. The present measurements establish a limited capability observation, not superiority over other systems.

How to read this report

Topic	Pages	Question
Temporal semantics	3–4	What should a memory update mean?
Architecture and verification	5–7	What is implemented and what is checked?
Evaluation and boundaries	8–10	What does the evidence actually support?
Next study and references	11–12	What is needed to make stronger claims?

Evidence note. This is a condensed revision of OMVY's August 2026 manuscript [1]. Internal results are author-reported; no experiments were rerun for this edition. Product capabilities developed after that snapshot are outside this report. No comparative efficacy or production-effectiveness claim is made.

Two clocks for one history

What was true then? What did the system believe then?

Event time, also called valid time, describes when a fact held in the world. Belief time, represented through transaction-time records and the declared read surface, describes when the system held the corresponding belief. A late update makes the difference visible.

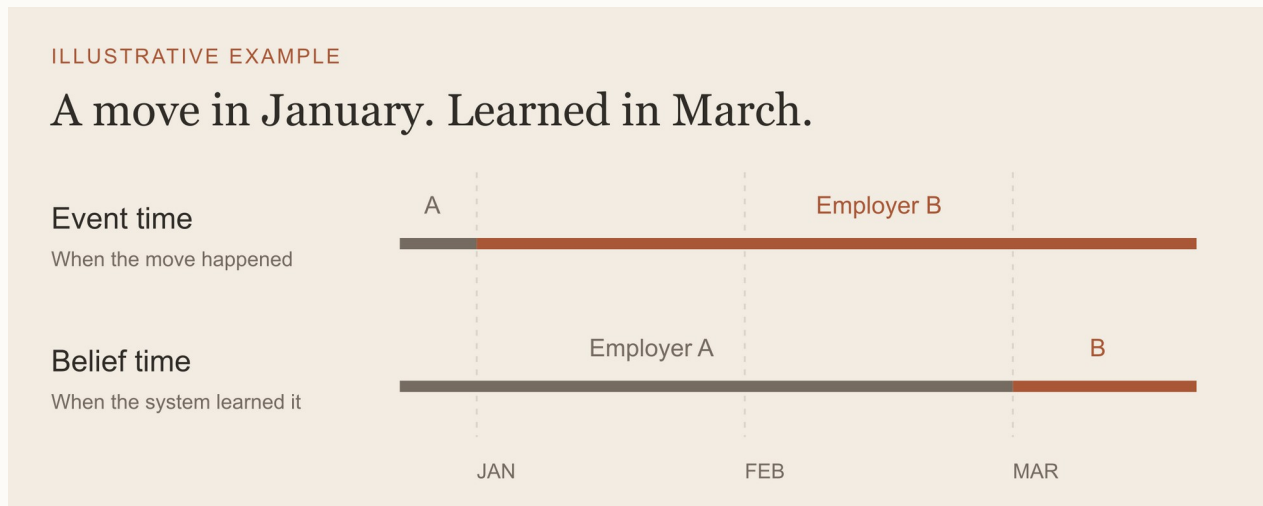


Figure 1. A fictional employee moves from A to B in January, but the system learns this in March. The two views answer different questions about February.

After the update, an event-time question about February should return employer B. A question about what the system believed in February can still return employer A. Preserving the earlier record makes that distinction representable; exposing the correct read surface makes it usable.

Intervals and provenance

The relation store uses half-open event windows: a fact holds from its start instant up to, but not including, its end. Transaction retirement records that a stored belief is no longer current without necessarily deleting its history. A zero-width event window is valid at no instant and can therefore retain a contested observation without asserting it as a world fact.

Established semantics in an agent system

Bi-temporality is not new. SQL:2011 combines application time with system versioning and supports changes to bounded portions of an interval [2]. Lashley applies this established distinction to a relation store, then connects it to agent-facing intent, authority, recall and verification. Its contribution is not the invention of two clocks.

Read boundary. The evaluated belief-time route reconstructs history above the store and is capped at 100,000 rows. Physical retention alone does not establish complete or correct historical answers. See page 10.

Five meanings of a new statement

A similar sentence shape can require a different operation on memory.

Suppose the store believes that Marco has worked at Ferrari since 2018. A new statement may describe a move, deny the original record or merely contest it. The distinction matters because an update can change current answers, historical answers or neither.

Outcome and example	Required memory effect
World-state transition "He moved to Ducati in 2024."	Close the outgoing event window at the change date and assert the new value. Ferrari remains the answer for 2019.
Retroactive correction "He was never at Ferrari."	Remove the outgoing claim from the corrected event-time scope while preserving the record of having believed it. Requires explicit bounds and authority.
Dissent "I dispute that employment history."	Retain the observation as contested, without making it a rival current fact. The production parking path uses a zero-width row and a resolution queue.
Deferral "Move the renewal to October."	Close the old prospective commitment and create its successor. The vacated date must stop answering as an active commitment.
Cancellation "Drop the renewal."	Close the prospective commitment without creating a successor. Record cancellation in the audit vocabulary.

Five represented outcomes, four passive routes

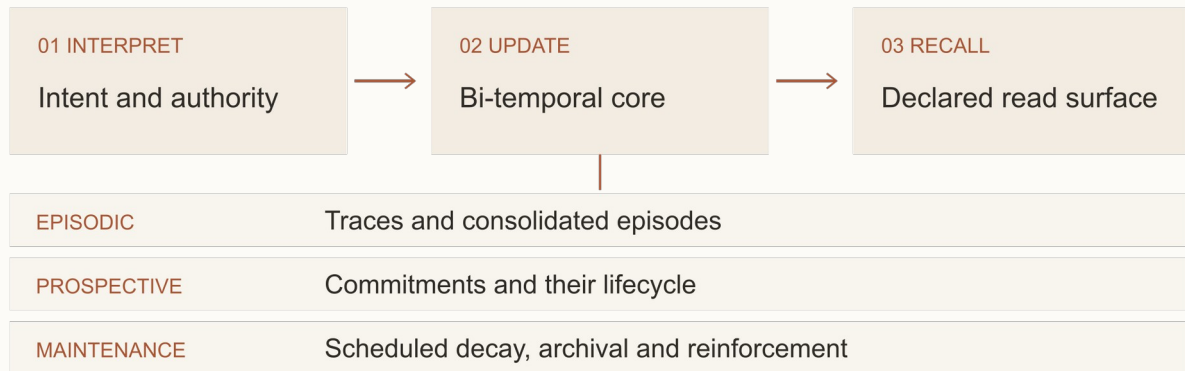
The evaluated passive-ingest path routes transition, dissent, deferral and cancellation. It does not infer an unlimited correction of the past from an ordinary dateless replacement. A bounded historical assertion by itself also cannot declare that an outgoing value never held over its entire lifetime; that requires the appropriate authorised storage operations.

For a future commitment, deferral closes the old row at its own due date; a commitment that has already matured closes at the time of the deferring act. Cancellation is limited to prospective edges and is not a general-purpose route for retiring arbitrary semantic facts.

Dissent boundary. The zero-width production path leaves the incumbent unchanged. A dormant queue-cap fallback can promote a contested observation and remains excluded from the functional-slot claim. This is a scoped implementation contract, not an architecture-wide guarantee that dissent can never become current.

One temporal core

Supporting memory systems have different jobs and evidence boundaries.



Session attention and outcome monitoring span these components.

Figure 2. Conceptual architecture grouped by responsibility, not measured retrieval benefit.

Semantic and episodic memory

The semantic core stores relations with temporal scope. Episodic memory retains traces and consolidated episodes, so recall can select an appropriate level of detail for the question's time window. Cognitive terminology is design inspiration, not evidence of psychological equivalence.

Prospective memory and maintenance

Future-dated records represent commitments with due, deferred, resolved and cancelled states. Scheduled maintenance handles consolidation, FRS-based decay and archival. Archival retires relation rows; destructive erasure belongs to a separate control plane. Retaining history has a storage cost.

Attention and associative access

Session attention binds active entities across processing layers. Associative preload can bring neighbouring context into view. The reported tests solve constructed one-hop cases, miss two-hop cases and show no change on a 40-query session evaluation. No end-to-end benefit is claimed for this component.

Monitoring without overstating confidence

The monitoring surface reports confirmation and revision history, temporal precision and truncation—not calibrated per-fact confidence. No outcome rate is returned below two observations. Recall buffers reinforcement in memory; a declared writer persists it through a durable leased queue. Reading is not itself persistent maintenance.

Preserve the past without duplicating the present

The verification vocabulary distinguishes a relation edge from a functional slot.

Property	Scope and meaning
P1-edge Non-overlap	For one full edge—workspace, source, target and predicate—transaction-current event windows must not overlap. Different targets are different edges.
P2-edge Incidental retirement	Beyond caller-authorised retirement, a row may be retired only where keeping it would violate P1-edge. Archive destruction and in-place history rewrites are checked separately.
P3 Fragment bound	One operation may mint at most two remnants of a given parent. This is not a bound on lifetime row count or total storage growth.

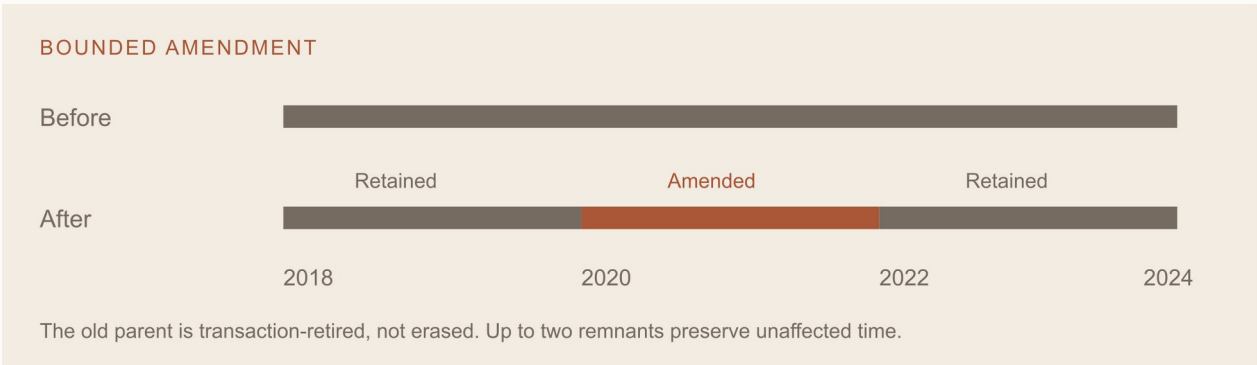


Figure 3. An authorised amendment to 2020–2022 retains the unaffected portions of a 2018–2024 interval. Schematic half-open intervals; not a measured result.

Functional slots require a separate policy

P1-edge does not prevent both `works_at(Marco, Ferrari)` and `works_at(Marco, Ducati)` from answering at the same time: their targets differ. P1-functional is a separate write-path postcondition for predicates in the workspace’s enforced registry. In the evaluated design, a curated floor covers 13 predicates and graph-derived expansion requires 60 contested subjects.

Functional-slot evidence comes from branch tests and exported-state audits, not the same seeded two-backend gate as P1-edge. Predicates outside the registry have no slot-level guarantee. Any non-zero functional violation is a safety violation in that run, not statistical noise.

Authorisation matters. Permission to retire is not permission to destroy. A bounded amendment must preserve unaffected history; archival must not silently become erasure. The checker inspects raw before-and-after state so that a read filter cannot conceal corruption.

Test the rules and the tests

A green run is useful only when the detector is known to catch its target defect.

The seeded checker generates programs over eight public write operations, addresses rows symbolically and shrinks failures into reproducible sequences. Its violation ledger is checked in both directions: unexpected classes fail the gate, and listed classes must remain reproducible.

Regression-only detectors are exercised on deliberately constructed bad states.

Reported regime	Budget	What it establishes
Sequential CI SQLite	400 seeds × 40 steps 16,000 calls	A bounded regression search over the generated write surface.
Wide sequential run SQLite	3,000 seeds × 80 steps 240,000 calls	A larger search that also exposed an incorrect earlier formulation of P3.
Backend parity PostgreSQL 16.14	40 seeds × 20 steps 800 program steps	Identical observed violation-class sets for the replayed programs, not identical step-by-step behaviour.

Table 1. Author-reported verification regimes from the August manuscript. These are declared test budgets, not public reproduction records or proof over all possible programs.

Controls that make the result interpretable

Historical defects are replayed independently of the repaired code so the controls must still expose them. Mutation checks remove individual protections to test whether assertions notice. Delete-entity and erase-workspace operations are excluded from random generation because they destroy the fixture; dedicated witnesses exercise their hard-delete behaviour instead.

Concurrency is a different experiment

Sequential programs on one connection cannot establish race safety. The PostgreSQL design orders workspace and triple locks and uses forced-interleaving tests. A lock-removal mutation on the dissent door remains undetected. Quantitative concurrency results are withheld because the reported denominator no longer matches the executed program and the full-budget rerun is unresolved.

The supported statement is consequently narrow: internal regression checks cover declared store properties in declared regimes. They do not prove extraction correctness, complete functional-slot enforcement, historical-read correctness or universal concurrency safety.

An instrument for changing facts

A deterministic synthetic world makes temporal questions explicit.

TULVING generates a world and gold timeline from a seed, then asks questions about its changing state. A freeze manifest records the ontology, generator parameters, answerer configuration, prompts, evaluated surfaces and scorer digests. Headline scoring is deterministic; answer generation can still depend on a language model and provider behaviour.

Task family	Question or target	Evidence status
Current state	What holds now, and does the answer repeat a superseded value?	Reported on one world
As-of retrieval	What held at an event-time instant, or what was believed then?	Event and belief variants reported
Intent routing	Which evaluated update route does the utterance require?	Restricted probe reported
Prospective timing	Which commitments are due, and did any activate early?	Within reported task scope; no summary score added here
Audit	Why does this hold, and what was believed before?	System result withheld pending reconciliation
Oblivion	How should the system forget under the specified task?	Specified, not evaluated

Instrument checks are not product gains

An audit scorer initially rejected readable dates that the prompt had not required in a specific format. Normalising those dates repaired the instrument; it did not improve the memory system. The audit system result remains withheld until conflicting readings are reconciled. The printed belief-time control summary is also missing its amnesiac result.

What the present dataset cannot settle

All reported system runs use one generated world. Three repeats share a scorer digest and nominal answerer configuration but predate immutable writing-code attestation. TULVING v1 is single-turn, does not test conversational repair and does not generate every storage-level update shape. Its CORRECTION label uses transition semantics, so the current intent probe is not a five-way validation of the taxonomy on page 4.

No paired cross-system baseline or intent ablation was completed in the reported study. A synthetic gold timeline is exact within its own generated world; it does not establish realism or production prevalence.

A capability observation, not a performance claim

The data describe the evaluated August snapshot—not today’s product.

Belief-time answers became expressible

The evaluated pre-change interface did not expose a belief-time route to the benchmark adapter and scored 0/90. Four later, non-paired runs produced 12–19 correct answers out of 90. Those runs include other code and run-level differences; they cannot isolate the causal accuracy effect of adding the route.

The supported result is that the later interface represents and returns some answers to a previously unavailable class of query. This is not high-quality retrieval: 12–19/90 remains far below the oracle’s 90/90. No overall product accuracy follows from this one synthetic task.

Observed repeat-run counts

Task	Run 1	Run 2	Run 3	Denominator
Current state	124	133	131	180
As-of event time	21	15	15	90
As-of belief time	14	18	12	90
Intent probe	12	10	10	120

Table 2. Descriptive task counts transcribed from the manuscript’s three-repeat table. These are not a common performance scale across task families. They must not be conflated with the separate four-run belief-time range above. Runs use one world and lack immutable writing-code attestation; no confidence intervals or significance claim are inferred.

What exported state showed

Distinct extracted predicate strings changed from 164 to 85 when the extractor received a ranked archive vocabulary. This is a descriptive reduction in vocabulary fragmentation, not evidence that meanings were merged correctly. It needs labelled semantic review before a quality claim is made.

The latest reported exports contained zero P1-edge violations. That result does not establish P1-functional, extraction correctness or accurate answers. Intermediate functional-slot violations increased after the vocabulary change, while the final checkpoint did not increase. The suggested explanation—greater visibility to the functional vocabulary—remains a hypothesis, not a demonstrated improvement.

Where the evidence stops

Retaining history, exposing history and answering correctly are separate achievements.

A read route can change context without improving answers

On a separate 41-question event-time probe, closed windows rendered in context changed from 0 to 26, while present-day rows changed from 41 to 26. These are context-composition counts, not accuracy measurements. No paired on/off score was run, and the probe is one slice of one generated world.

Historical reading has operational boundaries

Native entity and event-time reads exclude transaction-retired rows. The evaluated belief-time view reconstructs prior belief in TypeScript over a full-history workspace read capped at 100,000 rows, without pagination. The sequential store gate does not validate that read surface, and completeness beyond the cap is not established. An early-resolved commitment may also remain visible in belief time until its due instant.

Retention has a cost and erasure has a different purpose

Archival keeps relation history and therefore does not reclaim its storage. Repeated updates can increase retained rows even when each individual split satisfies P3. The cost of large overlap sweeps, lock holding and long-term storage growth was not quantified. Destructive erasure must remain an explicit operation, not a hidden side effect of maintenance.

Degraded extraction and evaluation validity

A provider fallback can preserve an entity title while omitting relation updates. That is degraded memory quality, not equivalent behaviour. Provider degradation was manually annotated rather than captured by a complete structured protocol. Synthetic single-world results also cannot establish realistic language coverage, multi-turn usefulness or reliability under real workloads.

What is not claimed

This report does not establish superiority over Zep or another memory system; a causal accuracy gain; universal write correctness; calibrated confidence; complete audit reconstruction; an oblivion result; or production effectiveness. Deployment volume, real stale-answer incidents, latency distributions, cost and long-horizon task success were not measured in the study.

These limits apply to the evaluated implementation and evidence snapshot. They are not an assessment of engineering work completed after 18 August 2026.

From inspectable claims to reproducible evidence

The next study should strengthen the evidence, not enlarge the claim faster than the evidence.

Position in the literature

Temporal databases already distinguish application time and system time [2]. Zep describes a temporal knowledge graph for agent memory [3], while MemGPT develops memory management beyond a model's immediate context [4]. These are relevant precedents, not evidence that the systems are equivalent. Lashley's focus here is the operational connection between update intent, temporal postconditions and scoped internal verification.

TULVING isolates questions about changing facts. An external benchmark such as LongMemEval [5] would add a useful, independently designed evaluation surface; it would complement rather than replace temporal and store-level tests. This edition deliberately makes no unverified competitor feature ranking.

The next evidence package

Priority	Required work	Claim it could support
01 Freeze	Attest code, models, prompts, datasets, adapters and scorer versions in immutable bundles.	A result attributable to a specific evaluated system.
02 Compare	Pair a temporal baseline and full-context control; repeat within multiple worlds and report uncertainty.	An interpretable comparative estimate.
03 Isolate	Run paired event-route and intent ablations with error analysis.	A component-level effect rather than a mixed version change.
04 Reconcile	Resolve the audit metric, missing controls and concurrency rerun; close declared coverage holes.	Clearer audit and verification boundaries.
05 Measure	Collect realistic task outcomes, latency, cost, storage growth and degradation behaviour.	Evidence about production usefulness.

A precise technical thesis

The present contribution is an architecture, a verification discipline and a benchmark instrument with explicit limits. The durable research question is how a memory system can change its answers without losing the record of why those answers changed.

Sources and availability

Research edition 01 · Prepared 7 October 2026

This condensed report adapts the source manuscript [1]. It preserves the reported evaluation snapshot, narrows claims to their evidence, replaces the broad capability matrix with a scoped discussion, and separates established temporal semantics from the system's implementation choices. The scientific experiments and internal property tests were not rerun for this edition.

Artefact availability

The engine is proprietary. The August manuscript describes a benchmark package prepared for release but cites no immutable public archive, DOI, release tag or checksum. This edition does not verify a subsequent release. Internal tests therefore remain author-reported evidence, not independently reproducible implementation results.

The intended release includes the generator, specification, ontology, seeds, manifests, adapters, prompt and scorer digests, stored traces and commands that regenerate reported tables. Public data and scoring artefacts could enable inspection without exposing the full engine, but they would not by themselves reproduce the proprietary write path. An independent reproduction has not been documented here.

Selected references

[1] OMVY. Lashley: An Executable Bi-Temporal Memory Architecture for Long-Lived Agents. Source manuscript, evaluation snapshot 18 August 2026. Unpublished author-supplied document.

[2] Kulkarni, K., and Michels, J.-E. Temporal features in SQL:2011. ACM SIGMOD Record 41(3), 2012, pp. 34–43. <https://sigmodrecord.org/publications/sigmodRecord/1209/pdfs/07.industry.kulkarni.pdf>

[3] Rasmussen, P., et al. Zep: A Temporal Knowledge Graph Architecture for Agent Memory. arXiv:2501.13956, 2025. <https://arxiv.org/abs/2501.13956>

[4] Packer, C., et al. MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560, 2023. <https://arxiv.org/abs/2310.08560>

[5] Wu, D., et al. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. arXiv:2410.10813, 2024. <https://arxiv.org/abs/2410.10813>

Source map

Report pages 3–4 draw on source §§1–2; page 5 on §3; pages 6–7 on §4 and Appendix A; pages 8–10 on §§5–8; page 11 on the stated research agenda and selected external references; this page on §10. The source manuscript retains the detailed internal test names, development record and broader bibliography.