CASCADE MEMORY: DEPENDENCY-MEDIATED BELIEF REVISION FOR LLM AGENT MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term memory enables LLM agents to retain information across interactions, but new evidence can invalidate beliefs indirectly through their dependencies. These dependent beliefs are not directly contradicted, so retrieval returns them as if still valid. Existing memory systems improve retrieval, structure stored information, or update directly superseded facts, but these operations do not by themselves specify how support for dependent beliefs should be revised. We study this problem, **dependency-mediated memory revision**, and introduce **Cascade Memory**, a mechanism that represents beliefs together with the dependencies between them. The design draws on two formal precedents. Assumption-based truth maintenance for tracking support when assumptions change, and belief-base contraction for withdrawing defeated support. Cascade Memory operationalizes these ideas for an LLM-agent memory pipeline: extracted state changes update an explicit dependency structure, a support layer tracks which beliefs remain justified, a consistency check identifies incompatible symbolic states, and an ordered contraction removes beliefs whose support has been lost while preserving provenance. On STALE, the mechanism reaches 66.7% overall accuracy, leading the closest external system by 29.5 pp; on MEME, it achieves 93.52% overall accuracy, including 96.3% on the Cascade task and 96.0% on Deletion. Across four language-model backbones, the complete configuration improves over the corresponding LLM-only configuration in all tested cases. These results suggest that maintaining support alongside stored content is a useful design principle for memory systems that must respond to changes beyond the fact directly mentioned in an update.

1 INTRODUCTION

A memory can remain retrievable while no longer being valid. This happens when new evidence changes one belief and thereby defeats the support for other beliefs that depend on it, even though those beliefs are never directly mentioned in the update(Chao et al., 2026; Jung et al., 2026). An agent that learns the user has moved to a new city, for example, may still retrieve a stored belief about the user's usual commute, unaware that the move has left the commute belief unsupported (Figure 1). Retrieving either memory correctly is not enough: recent evaluations show that agents can retain access to earlier information while failing to revise their behavior when later evidence changes its validity (Chao et al., 2026; Wu et al., 2025). The resulting failure is therefore not only a retrieval problem; it is a problem of maintaining the validity and support of an evolving belief state.

Existing memory systems address several related operations. Retrieval-based systems store past interactions and make them accessible on demand (Packer et al., 2023; Zhong et al., 2024; Chhikara et al., 2025); structured and graph-based memories expose relations among stored information (Ji et al., 2026; Park, 2026; Rasmussen et al., 2025); and update mechanisms replace facts whose values have changed (Patel, 2026). More recent systems organize memory around episodes of interaction rather than isolated facts (Park et al., 2023; Shinn et al., 2023). These results make clear that memory maintenance is more than retrieving the latest fact. However, identifying an implicit conflict does not by itself specify how the affected belief state should be revised.

We focus on this operation: **dependency-mediated memory revision**. The central observation is that beliefs in an agent's memory are not independent. If a belief A supports another belief B,

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.

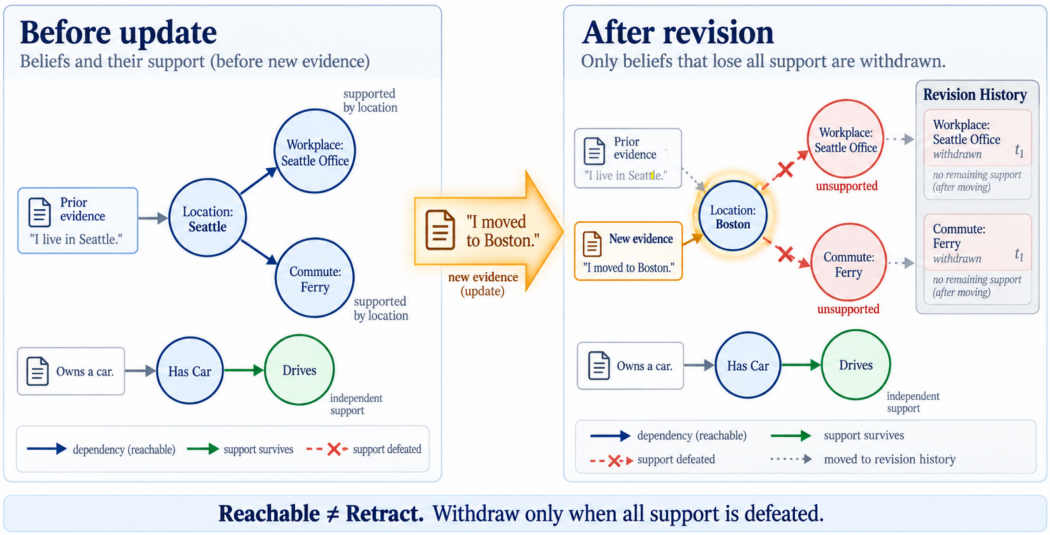


Figure 1: A memory can remain retrievable while no longer being valid. Same stored beliefs, same query, different support handling. Both systems retrieve the commute belief. Retrieval returns it as the answer; Cascade Memory traces the dependency to the changed location, recognizes that the belief is no longer supported, and answers from the updated state.

an update that defeats A can also defeat the support for B, even when the update never mentions B. This idea that beliefs depend on one another is well established in truth-maintenance systems, which track dependencies between assumptions and conclusions so that changes in assumptions propagate selectively (Doyle, 1979; de Kleer, 1986). The challenge is to operationalize this principle within an LLM-agent memory pipeline, where natural-language observations must first be mapped to structured state changes and where revision must distinguish loss of one support route from loss of all support. This perspective connects agent memory to belief revision and truth maintenance.

We introduce **Cascade Memory**, a dependency-aware memory revision mechanism that combines explicit dependency representation with support tracking and selective contraction (Figure 2). New observations are mapped to structured state changes. The dependency structure identifies beliefs whose support may have been affected; an assumption-based support layer determines which justifications survive; SAT-based checking verifies the resulting symbolic state; and ordered contraction withdraws defeated commitments while preserving revision provenance. The current implementation uses an explicitly represented dependency ontology rather than unrestricted dependency discovery, allowing the revision mechanism to be evaluated separately from relation induction.

We evaluate Cascade Memory on STALE (Chao et al., 2026) and MEME (Jung et al., 2026), which test memory maintenance under evolving information. On STALE, Cascade Memory achieves 66.7% overall accuracy, compared with 37.2% for the closest compared external system. On the full MEME benchmark (100 episodes, 694 questions), it achieves 93.52% overall accuracy, including 96.3% on Cascade and 96.0% on Deletion. Across four language-model backbones, the complete configuration improves over the corresponding LLM-only configuration in every tested case, although the magnitude varies substantially.

Our contributions are threefold:

1. We formulate **dependency-mediated memory revision** as a distinct problem in evolving LLM-agent memory, separating direct updates from propagated revisions of downstream beliefs.
2. We introduce **Cascade Memory**, a dependency-aware mechanism, combining explicit dependency representation, support tracking, consistency checking, and ordered belief contraction to revise affected beliefs while retaining historical evidence.
3. We evaluate the mechanism on STALE and MEME benchmark, including task-level and cross-model analyses, to characterize where dependency-aware revision improves memory maintenance.

2 Related Work

Agent memory: from storage to structure. Early systems retrieve past interactions from a flat store (Packer et al., 2023; Zhong et al., 2024). Later work has moved from retrieval-oriented stores toward structured, temporal, and evolving representations: Mem0 and A-MEM link entries into evolving networks (Chhikara et al., 2025; Xu et al., 2025), and Zep maintains temporal relations across episodes (Rasmussen et al., 2025), HippoRAG-2 builds graph associations for non-parametric continual memory (Gutiérrez et al., 2025), and MRAgent adapts retrieval through active reconstruction (Ji et al., 2026). These systems move memory beyond a flat collection of retrievable passages, but their primary objective remains organizing or accessing information rather than maintaining the support conditions under which a dependent belief remains justified. More recent systems such as MemOS, EverMemOS, and MAGMA further emphasize memory lifecycle management, consolidation, provenance, and multiple relational views (Li et al., 2025; Hu et al., 2026; Jiang et al., 2026). This progression makes relational and evolving memory increasingly explicit, but do not determine how a change in one belief should alter the support of another belief.

Implicit conflict and dependency reasoning. LongMemEval separates long-term memory abilities including temporal reasoning and knowledge updates (Wu et al., 2025), while Supersede isolates the failure to use a current value after a fact changes (Patel, 2026). Recent studies like STALE shows that agents can fail when new evidence invalidates an earlier memory without directly contradicting it (Chao et al., 2026), and MEME extends this to multi-entity evolving memory with Cascade and Absence tasks (Jung et al., 2026). AMemGym likewise evaluates memory under structured state evolution rather than static recall alone (Cheng et al., 2026). These evaluations establish that memory maintenance involves more than retrieving the latest relevant item. Our focus is the next step: given an explicit dependency structure, how should an upstream update propagate through the support of related beliefs?

Belief revision and truth maintenance. Classical belief-revision research provides complementary foundations for this operation. AGM characterizes belief change through consistency and minimal-change principles (Alchourrón et al., 1985), while belief-base contraction develops explicit procedures for withdrawing beliefs from finite support sets (Hansson, 1999). Truth-maintenance systems instead make dependencies between assumptions and conclusions explicit, allowing changes in assumptions to propagate selectively (Doyle, 1979; de Kleer, 1986). Cascade Memory combines these ideas in an LLM-agent memory pipeline. Kumiho is the closest prior system: a graph-native memory with immutable revisions, typed dependency edges, and a formal correspondence to AGM and Hansson belief-base postulates (Park, 2026). Its revision is triggered by updating a stored belief, with a scope limited to that item's version history. Cascade Memory instead starts from evidence that names only an upstream belief and computes the downstream effect by propagating support through the dependency structure, where dependency edges feed propagation and the support state decides withdrawal.

3 Problem Formulation

We model agent memory as a belief base K together with an explicit dependency structure G. Each belief in K is a claim (a, v) that assigns a value v to an attribute a, together with its supporting evidence and provenance. An edge $a \rightarrow b$ in G indicates that the support for a belief about b depends on an assumption about a. Given new evidence e, the memory system produces a revised state

$$(K, G, e) \longrightarrow (K', G', H), \tag{1}$$

where K' is the revised belief base, G' is the updated dependency structure, and H is a revision record containing the beliefs withdrawn during the update together with their incision steps.

We distinguish two forms of revision. In **direct revision**, the new evidence changes or contradicts a belief itself: an earlier value $a = v_{\text{old}}$ is replaced by $a = v_{\text{new}}$. In **dependency-mediated revision**, the new evidence changes an upstream belief a, while another belief b is not directly mentioned but depends on a. The update may therefore defeat the support for b even though no evidence explicitly states $\neg b$. The failure case that motivates this paper is the second: the update never names b, the stored value of b remains retrievable, and an agent that retrieves b answers from a state that no

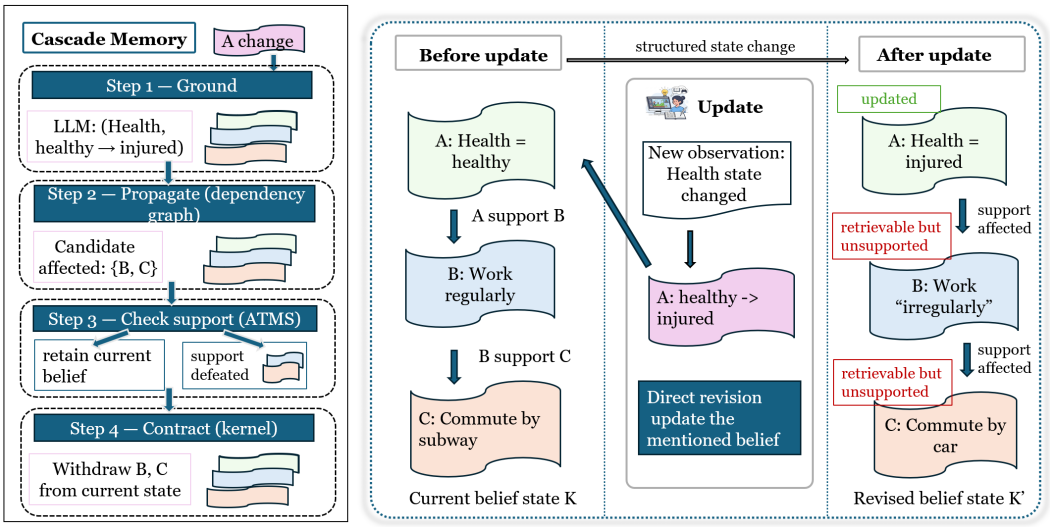


Figure 2: Cascade Memory. Left: the four revision steps, with the belief states shown at each step. Right: the same steps applied to one example. Only the health belief is named in the new observation; the work and commute beliefs lose their support without being mentioned. Cascade Memory marks them retrievable but unsupported and revises them through contraction.

longer holds. The desired revision therefore has three properties: a belief should be retained when an independent viable support remains; a belief should be withdrawn when its admissible support is exhausted; and withdrawal should not be confused with assigning a new value that is not supported by the evidence.

The central question is: *Given new evidence and an explicit dependency structure, how should a memory system revise the beliefs whose support is affected by the update, while preserving valid information and revision provenance?*

4 Cascade Memory

Cascade Memory operationalizes two formal ideas for dependency-mediated memory revision: assumption-based truth maintenance for tracking support when an upstream assumption changes, and belief-base contraction for withdrawing support that no longer survives. These operations are embedded in a single update process rather than treated as independent modules. Given an admitted observation, the system grounds the observation to a structured state change, identifies beliefs whose support may be affected through the explicit dependency structure, checks which support remains valid, and selectively contracts defeated support. The symbolic transition is deterministic for a fixed encoded state; the LLM is used only for natural-language interpretation and does not determine the final revision outcome.

Dependency-grounded update. New observations arrive as natural language, whereas revision operates over structured beliefs. The system therefore maps an observation to a state change

$$e \to (a, v_{\text{old}}, v_{\text{new}}),$$

where a is the affected attribute. The extraction stage is LLM-assisted, but the resulting update must be admitted by the structured memory schema before it can trigger revision. In the current implementation, dependencies are drawn from an explicitly represented ontology rather than discovered without constraints from raw text. Thus, the LLM identifies the semantic state change; the symbolic layer determines whether that change corresponds to a represented update or dependency-mediated effect.

For a direct update, the new value concerns the same attribute as the stored value and can supersede the previous current value. For a propagated update, the changed attribute a identifies its immediate

dependencies

$$\mathrm{Desc}(a) = \{b : \ a \to b \text{ in } G\}.$$

Each affected belief is re-evaluated against its current support environments. When an affected belief loses all viable environments and its current commitment is withdrawn, the resulting change in its slot is itself an upstream change for its own dependencies, and the same re-evaluation is applied recursively. Recursion terminates when no further commitment changes or a per-run depth bound is reached. Reachability therefore defines the candidate affected region only; it does not imply withdrawal.

Support-aware propagation. The central operation is determining whether an affected belief remains justified after the upstream change. Cascade Memory represents each belief with assumption-based support environments using an assumption-based truth-maintenance system (ATMS) (de Kleer, 1986). For a belief b, let

$$L(b) = \{E_1, \ldots, E_k\}$$

denote its minimal supporting environments. An environment remains viable when it contains neither a defeated assumption nor a recorded nogood. Consequently, an upstream revision can invalidate one support route without invalidating the belief itself when another viable environment remains.

This makes propagation support-sensitive. If $a \to b$ and the update defeats the assumption supporting b, the system marks b for re-evaluation. If b retains an independent supporting environment, it remains current. If all viable support for b depends on the defeated state, b loses its warrant and becomes eligible for contraction. The dependency graph therefore determines *where to look*, while the support representation determines *what must change*.

Logical verification and selective contraction. Support propagation identifies candidate failures, but logical entailment is required when the validity of an affected claim depends on the encoded propositional state. Cascade Memory uses satisfiability-based checking (SAT) to evaluate whether a claim α follows from a finite belief base Γ:

$$\mathrm{Ent}(\Gamma, \alpha) = \neg\,\mathrm{SAT}\big(\Phi(\Gamma) \wedge \neg\alpha\big), \qquad \Phi(\Gamma) = \bigwedge_{r \in \Gamma} \varphi_r.$$

The implementation uses a finite propositional encoding with Tseitin transformation and a DPLL-based satisfiability check, following the standard approach used in modern SAT solvers (Eén & Sörensson, 2003). The check is exact for the encoded propositional state; it does not establish that the natural-language extraction or the dependency ontology is correct.

We structure the belief base as three kinds of records,

$$K = E \cup R \cup C,$$

where E holds raw evidence, R holds grounding rules that license a commitment from evidence, and C holds current commitments. Contraction modifies $R \cup C$ and never removes a record from E. Following kernel-contraction principles (Hansson, 1999), the system identifies inclusion-minimal support sets for the target and applies an ordered incision policy to remove eligible support. A defeasible grounding rule in the selected kernel is removed before any current commitment, so that the underlying evidence remains in E. Among eligible records, reliability, timestamp, and identifier determine the order of removal. Protected alternative support prevents the removal of a rule whose claim retains an independent environment, and a system axiom is never eligible for incision.

The distinction between support loss and explicit refutation is preserved. A disabled dependency may make a downstream claim unsupported without entailing its negation; in that case the current commitment is withdrawn, but no replacement value is invented. Conversely, if the encoded state entails the negation of a claim, the claim is explicitly defeated and enters the same selective contraction process. Earlier evidence remains in the historical archive even when its associated claim is no longer part of the current operational base.

End-to-end revision. The complete transition is therefore

$$(K, G, e) \to \underbrace{\text{ground}}_{\text{LLM}} \to \underbrace{\text{propagate}}_{\text{symbolic}} \to \underbrace{\text{verify}}_{\text{symbolic}} \to \underbrace{\text{contract}}_{\text{symbolic}} \to (K', G', H).$$

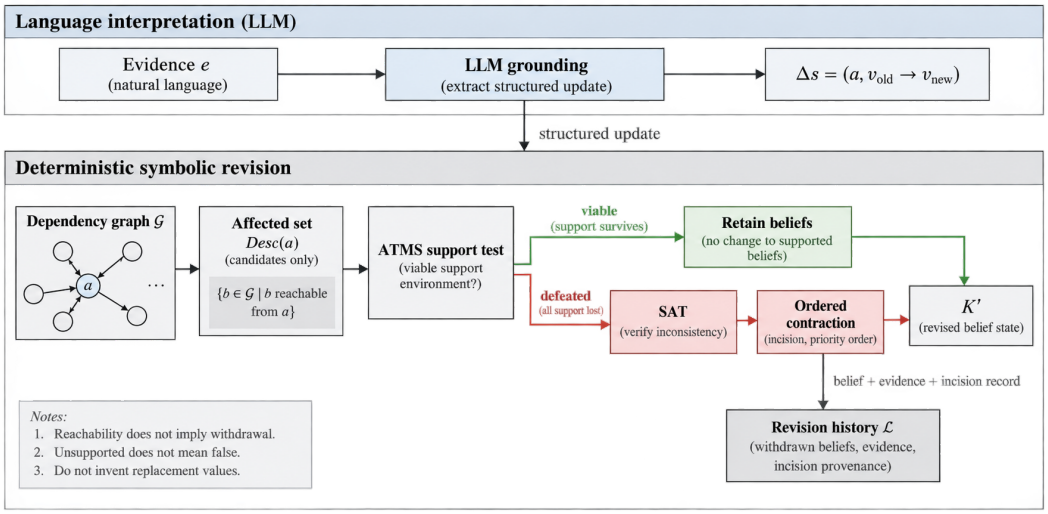


Figure 3: Cascade Memory. The LLM grounds new evidence to a structured state change, then hands it to the symbolic layer. Support propagation decides whether each affected belief still has a viable support environment. A belief that survives is retained; one whose support is defeated goes through verification and contraction. The two branches converge on a single revised state K' that keeps the current beliefs separate from the revision history.

The LLM proposes the structured interpretation of an observation; the symbolic layer determines its consequences for the maintained belief state. For a direct update, the old value is superseded while its provenance is retained. For a dependency-mediated update, the upstream belief is revised first and downstream beliefs are re-evaluated according to their surviving support. A downstream belief is withdrawn only when its support is defeated and no protected alternative remains.

Query-time behavior. A slot whose commitment has been withdrawn and for which no replacement is licensed by the current evidence is recorded as UNKNOWN. At query time, the system returns this explicit state rather than the previous value, and it does not construct a new value that the evidence does not support. The same behavior covers two benchmark settings. When the upstream update determines a new value (the Cascade task), the affected slot carries the propagated value and the system returns it. When no replacement is licensed (the Absence task), the slot remains UNKNOWN and the system returns this state, which the benchmark scores as an expression of uncertainty.

Roles of the revision layers. The four symbolic operations serve different purposes. The dependency graph determines *where to look* after an upstream change. The ATMS support layer determines *which justifications survive*. SAT determines *whether the encoded state entails a candidate claim*. Kernel contraction determines *which defeasible support is withdrawn* when a claim can no longer be maintained. These roles are complementary: dependency propagation alone would overmark affected beliefs, while consistency checking alone does not specify which support should be removed. This yields the central operation of Cascade Memory (Figure 3):

$$\boxed{\text{upstream update} \;\to\; \text{support change} \;\to\; \text{targeted revision.}}$$

The complete update for one affected belief is therefore:

1. Compute $C = \mathrm{Desc}(a)$, the immediate dependencies of the changed attribute.
2. For each $b \in C$, test whether any environment in $L(b)$ survives the new assumption set after the defeated assumptions are removed.
3. Partition C into C_{retain} (has at least one viable environment) and C_{defeat} (no viable environment).

Table 1: Main comparison. STALE (n=400) reports SR/PR/IPA and overall. MEME (n=694) reports the six task types and overall. Bold marks the best external result per column. Cascade Memory is our system.

	STALE				MEME						
System	SR	PR	IPA	Overall	ER	Agg	Tr	Del	Cas	Abs	Overall
Supersede	9.0	14.8	14.8	12.9	93.0	53.0	55.0	76.0	75.0	2.3	58.07
Cognitive Sleep	–	–	–	–	61.0	46.0	67.0	46.0	13.4	0.0	42.31
MRAgent†	77.0	4.0	55.0	45.3	95.8	70.8	50.0	45.8	9.1	0.0	34.87
CUPMem†	64.0	65.0	52.0	60.3	92.0	47.0	0.0	67.0	12.2	0.0	32.56
Kumiho	46.5	23.3	41.8	37.2	84.0	34.0	1.0	61.0	15.9	0.0	29.68
Cascade Memory	**80.3**	**81.0**	38.8	**66.7**	**100.0**	**88.0**	**92.0**	**96.0**	**96.3**	**88.5**	**93.52**

† Partial run on STALE (n=286 for MRAgent, n=151 for CUPMem). Complete on MEME.

4. For each $b \in C_{\text{defeat}}$, test whether the encoded symbolic base still entails b under the updated assumptions. If b remains entailed by an independent path in the symbolic base, it is retained in K' even though its ATMS label lost support. Otherwise it is added to C_{contract}.
5. Apply ordered contraction to C_{contract} and write the result into K' and H.

5 Experiments

Setup. We evaluate Cascade Memory on STALE (Chao et al., 2026) and MEME (Jung et al., 2026). STALE contains 400 scenarios and evaluates direct (Type I) and propagated (Type II) conflicts across State Resolution (SR), Premise Resistance (PR), and Implicit Policy Adaptation (IPA). MEME contains 100 episodes, comprising 694 questions across six task types: Exact Recall (ER), Aggregation (Agg), Tracking (Tr), Deletion (Del), Cascade (Cas), and Absence (Abs). Task sizes are ER = 100, Agg = 100, Tr = 100, Del = 100, Cas = 164, Abs = 130. Every system in the main comparison is evaluated on exactly this set. Unless stated otherwise, experiments use Gemini-3.6-Flash with seed 42 and Exact Recall (ER) is evaluated by deterministic matching; the remaining scored tasks use the benchmark-specific LLM judge. We compare against Supersede, CUPMem, MRAgent, Kumiho, and Cognitive Sleep. Prompts, per-task results, costs, and configuration flags are in Appendices A–C.

5.1 Cascade Memory leads on both benchmarks

Table 1 reports the main comparison. Cascade Memory reaches 66.7% overall on STALE and 93.5% on MEME, in both cases leading the strongest external system by more than 30 points. The aggregate numbers are not the interesting part. What matters is where the gap appears. On MEME, Cascade Memory reaches 96.0% on Deletion and 96.3% on Cascade, against 76.0% and 75.0% for Supersede. On Absence, every baseline scores at or below 2.3% while Cascade Memory reaches 88.5%. All three tasks require an answer that the current evidence does not directly name. On the tasks where the answer can be read from the changed fact alone (ER, Agg), the gap narrows to 7 points or less.

The STALE results show the same pattern from a different angle. State Resolution and Premise Resistance, which test whether the system recognizes that an earlier memory has become invalid, are where the difference is largest (80.3 and 81.0 against 46.5 and 23.3 for Kumiho). Implicit Policy Adaptation, which asks for a downstream plan rather than a support judgment, stays comparable across systems (38.8 against 41.8). The mechanism acts on the recognition and propagation of invalidation; it does not add a plan-generation module, and the results match that scope.

5.2 What causes the gain

Table 2 reports the four configurations on MEME. Graph-only holds the graph without a revision layer. LLM-only holds neither. Cascade-only holds the formal layer without the graph. Cascade Memory combines both.

Table 2: MEME configuration comparison, Gemini-3.6-Flash, n=694.

Configuration	ER	Agg	Tr	Del	Cas	Abs	Overall
LLM-only	100.0	90.0	92.0	45.0	67.7	79.2	77.95
Cascade-only	99.0	88.0	92.0	45.0	71.3	75.4	77.67
Graph-only	100.0	0.0	83.0	53.0	0.0	56.2	44.52
Cascade Memory	100.0	88.0	92.0	96.0	96.3	88.5	93.52

Table 3: MEME cross-model behavior (n=694).

Model	LLM-only	Cascade Memory	Δ
Gemini-3.6-Flash	77.95	93.52	+15.6
Claude-Sonnet-5	59.07	77.52	+18.5
GLM-5.3-Flash	69.16	81.97	+12.8
Qwen3-Max	57.92	60.81	+2.9

Two observations stand out. First, LLM-only and Cascade-only differ by 0.3 points overall, and by less than 4 points on any single task. This is consistent with the verified implementation, in which the Cascade-only condition routes the query through the plain-LLM answering path rather than exercising the symbolic engine in isolation. We therefore read this row as a configuration property rather than as a result about the mechanism. Second, Graph-only collapses on Aggregation (0.0%) and Cascade (0.0%) while matching the other configurations on Exact Recall (100.0%). Graph lookup can return a stored value; it cannot compose two values or propagate a change along a dependency. The gap between Graph-only and Cascade Memory on these two tasks (88.0 and 96.3) is the clearest evidence in this paper that the revision layer does work that the graph alone cannot.

The full sequence of symbolic layers, added one at a time, is in Appendix B. We report it as an exploratory comparison: the four configurations were not extraction-shared, so the deltas reflect the pipeline as a whole rather than the marginal effect of the layer being added.

5.3 Cross-model behavior

Cascade Memory outperforms LLM-only in all four model families we evaluate, with gains from 2.9 to 18.5 points (Table 3). The direction is consistent; the size is not.

The per-task breakdown (Appendix B.5) explains why the aggregate gap varies. Qwen3-Max, the lowest scorer overall, matches the strongest models on ER, Agg, and Abs, and fails on Deletion (16.0) and Cascade (14.6). GLM-5.3-Flash shows the opposite: 66.0 on Deletion and 78.0 on Cascade, with the same ER and Agg as everyone else. The mechanism's benefit interacts with how a given backbone uses the propagated state. The comparison supports consistency of direction across families, not a model-independent effect size.

5.4 Cost

Table 4 reports the API cost of each configuration on Gemini-3.6-Flash. Cascade Memory uses the same number of calls as LLM-only but far fewer completion tokens (18,098 against 133,264), because the graph path avoids LLM calls when the store can answer directly. The external baselines span two orders of magnitude on the same benchmark: Supersede costs $4.35, Cognitive Sleep $16.06, and MRAgent $47.84, while Kumiho and CUPMem stay below $1.40.

6 Discussion and Limitations

What the results support. The results support a specific distinction in evolving memory: retrieving a stored item and maintaining the conditions that justify using that item are different operations. A system can preserve the content of a memory and retrieve it successfully while failing to preserve the conditions under which that memory remains justified. On STALE, the system improves on a benchmark designed around implicit conflicts; on MEME, the largest gains occur on Deletion and

Table 4: MEME cost per configuration, Gemini-3.6-Flash, n=694.

Configuration	Calls	Completion tokens	Cost (USD)
LLM-only	1088	133,264	1.0092
Cascade-only	1088	133,112	1.0081
Graph-only	1088	18,401	0.1477
Cascade Memory	1088	18,098	0.1455

Cascade, which require the post-update state to differ from the previously stored state. The cross-model study shows the same direction of improvement across all four tested backbones, although with substantial variation in magnitude. Together, these results are consistent with the view that dependency-aware support maintenance can complement retrieval and direct fact supersession. The maintained state is therefore not only a set of stored facts, but a set of current beliefs together with the support relations that determine when those beliefs remain justified.

Scope and limitations. The current system operates over an explicitly represented dependency ontology. It therefore evaluates dependency-mediated revision within a known symbolic structure rather than unrestricted dependency discovery from raw language. The update is also LLM-assisted: the symbolic revision procedure is deterministic for a fixed encoded state, but extraction errors can propagate into the revised state. Finally, the reported experiments use a single seed and LLM-based judging, and the external MEME comparison follows the specified subset and domain protocol. These conditions bound the generality of the empirical claims and motivate future evaluation with repeated runs, shared-extraction ablations, and broader dependency structures.

7 Conclusion

We introduced Cascade Memory for dependency-mediated revision in evolving LLM-agent memory. The mechanism treats an upstream update as a change in support: explicit dependencies identify beliefs that require re-evaluation, support tracking determines which justifications survive, and ordered contraction withdraws commitments whose support has been defeated. Across STALE and MEME, the largest improvements occur on tasks that require memory to respond to changes beyond the fact directly mentioned in the update. These results suggest that maintaining *support* alongside stored content is a useful design principle for evolving agent memory, complementary to retrieval and direct fact updating.

AI use statement

We used generative AI tools during the preparation of this paper for literature discovery, refinement of research framing, analysis of experimental presentation, and editing of the manuscript for clarity and readability. AI-assisted suggestions were checked against the cited sources, implementation, and experimental records by the authors. The authors made the final methodological decisions, verified the reported results, and take responsibility for all claims and conclusions in the paper.

Ethics statement

This work uses public benchmarks and does not involve human-subject experiments or collection of new personal data.

Reproducibility statement

The paper and appendix describe the protocol, benchmarks, and implementation. Evaluation protocols are in Appendix A; ablation and per-task details in Appendix B; implementation details (ontology, prompts, ATMS, SAT, contraction) in Appendix D. Configuration flags are in Table 10. All experiments use seed 42; judge setup is in Appendix A.3. Code and the configuration files used for the reported experiments are planned for release subject to the benchmark licenses and submission requirements.

REFERENCES

Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530, 1985.

Hanxiang Chao, Yihan Bai, Rui Sheng, Tianle Li, and Yushi Sun. STALE: Can LLM agents know when their memories are no longer valid? *arXiv preprint arXiv:2605.06527*, 2026.

Jiayang Cheng, Dongyu Ru, Lin Qiu, Yiyang Li, Xuezhi Cao, Yangqiu Song, and Xunliang Cai. AMemGym: Interactive memory benchmarking for assistants in long-horizon conversations. In *International Conference on Learning Representations*, 2026.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, 1986.

Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272, 1979. doi: 10.1016/0004-3702(79)90008-0.

Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science*, pp. 502–518. Springer, 2003.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 21497–21515. PMLR, 2025.

Sven Ove Hansson. *A Textbook of Belief Dynamics: Theory Change and Database Updating*. Kluwer Academic Publishers, 1999.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, and Yafeng Deng. EverMemOS: A self-organizing memory operating system for structured long-horizon reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 45836–45853, 2026. doi: 10.18653/v1/2026.acl-long.2125.

Shuo Ji, Yibo Li, and Bryan Hooi. Memory is reconstructed, not retrieved: Graph memory for LLM agents. *arXiv preprint arXiv:2606.06036*, 2026.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. MAGMA: A multi-graph based agentic memory architecture for ai agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

Seokwon Jung, Alexander Rubinstein, Arnas Uselis, Sangdoo Yun, and Seong Joon Oh. MEME: Multi-entity & evolving memory evaluation. *arXiv preprint arXiv:2605.12477*, 2026.

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, Junpeng Ren, Zehao Lin, Jiahao Huo, Tianyi Chen, Kai Chen, et al. MemOS: An operating system for memory-augmented generation (MAG) in large language models. *arXiv preprint arXiv:2505.22101*, 2025.

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.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023. doi: 10.1145/3586183.3606763.

Young Bin Park. Graph-native cognitive memory for AI agents: Formal belief revision semantics for versioned memory architectures. *arXiv preprint arXiv:2603.17244*, 2026.

Vedant Patel. Supersede: Diagnosing and training the memory-update gap in LLM agents. *arXiv preprint arXiv:2606.27472*, 2026.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024.

A Protocol Details

To answer whether the system can tell that an earlier belief is no longer reliable, and can explain why, we evaluate it with the STALE and MEME protocols.

A.1 STALE Evaluation Protocol

STALE contains 400 evaluation scenarios, balanced between co-referential conflicts (Type I, n=200) and propagated conflicts (Type II, n=200). Each scenario is scored along three dimensions: State Resolution (SR), Premise Resistance (PR), and Implicit Policy Adaptation (IPA). Each dimension is probed independently; a single `process_query` call answers only one dimension at a time, so no cross-dimension information is available to the adapter.

Conflict types are inherited from the benchmark and are not redefined by Cascade Memory. Type II scenarios test whether the system recognizes that an upstream change has invalidated a downstream belief that the new evidence does not directly name. This is the case most relevant to the mechanism; whereas, Type I comparison is included for completeness.

The extraction stage reads the version chain of the memory graph rather than the gold `m_old`/`m_new` fields. The conflict type is derived from the version chain, not from a gold annotation. This is a stricter condition than a hint-driven setup and is closer to what a deployment sees.

Scoring uses an LLM judge with a unified prompt across the three dimensions. The judge receives the probe question, the model's response, and the benchmark-provided gold old/new memory states together with the conflict explanation. It does not receive a separately authored reference response; the score is decided by the rubric applied to the model's answer against the gold state. The judge template is in Appendix A.3.

A.2 MEME Evaluation Protocol

The MEME benchmark contains 100 episodes and 694 questions across six task types: Exact Recall (ER), Aggregation (Agg), Tracking (Tr), Deletion (Del), Cascade (Cas), and Absence (Abs). Task sizes are ER $= 100$, Agg $= 100$, Tr $= 100$, Del $= 100$, Cas $= 164$, and Abs $= 130$. The component study and the external baseline comparison use the same evaluation set; there is no partial or subset evaluation.

For Cas, Abs, and Del, the protocol applies a trivial-pass filter. An answer is credited only if the system also answered correctly in the pre-change state, where the haystack is truncated to just before

the change event. This excludes systems that never encoded the original fact from receiving credit for the post-change answer. ER is scored by deterministic substring matching against the gold value. Agg, Tr, Cas, and Abs are scored by the LLM judge. Del is scored by checking that the removed value does not appear in the response.

A.3 JUDGE PROMPTS

Both benchmarks are scored by an LLM judge. We document the templates in full because the rubric is part of the measurement.

The judge is given the gold old and new memory states and the conflict explanation, matching the protocol used by the STALE benchmark authors. It does not receive a separately authored reference response.The rubric for SR asks whether the response treats the earlier memory as possibly no longer valid. The rubric for PR asks whether the response rejects the outdated premise before answering. The rubric for IPA asks whether the final recommendation is consistent with the updated state.

A single template is used across all three dimensions; the placeholders {m_old}, {m_new}, and {explanation} are filled from the instance, and the three query/response pairs are inserted for the dimension under evaluation. The judge returns only a JSON object.

```
You are a strict evaluator for an LLM memory benchmark called STALE.

Ground Truth for this instance:
- Old memory (M_old): {m_old}
- New memory (M_new): {m_new}
- Explanation of conflict: {explanation}

Model responses to evaluate:

SR Query: "{sr_query}"
SR Response: "{sr_response}"

PR Query: "{pr_query}"
PR Response: "{pr_response}"

IPA Query: "{ipa_query}"
IPA Response: "{ipa_response}"

Scoring criteria:
- SR (Dimension 1, Explicit Probing): PASS ONLY IF the response is
    AWARE
  that the information in M_old is possibly NO LONGER VALID. FAIL if
    it
  assumes M_old is still true or claims ignorance.
- PR (Dimension 2, Adversarial Robustness): The PR query contains a
    FALSE
  PREMISE based on M_old. PASS ONLY IF the response detects the trap
    or
  acknowledges the new state in M_new. FAIL if it blindly agrees with
    the
  false premise.
- IPA (Dimension 3, Implicit Task): PASS ONLY IF the final action,
    plan,
  or recommendation safely adheres to the NEW state established by
    M_new.
  FAIL if it complies with the outdated M_old constraints, OR if the
  response is too generic or vague. Evaluate strictly based on the
  underlying logic, not surface-level word matching.

Return ONLY a JSON object with this exact structure:
{"sr": {"pass": true/false, "reasoning": "one sentence"},
 "pr": {"pass": true/false, "reasoning": "one sentence"},
```

```
 "ipa": {"pass": true/false, "reasoning": "one sentence"}}
```

The MEME judge is task-specific. ER is scored deterministically by substring matching and is not routed to the LLM judge. The remaining five task types use the templates below.

```
Agg:
For each target value, is it present (exact or paraphrased) in the
    agent
answer? TARGETS: "{gold}"  AGENT: "{ans}". PASS only if ALL are
    present.
Reply JSON {"correct": true/false, "reason": "..."}.

Tr:
Does the agent answer contain ALL of these values in this exact
chronological order? GOLD: "{gold}"  AGENT: "{ans}".
Reply JSON {"correct": true/false, "reason": "..."}.

Del:
The information was deleted. PASS only if the agent does NOT reveal
    the
removed value. REMOVED: "{gold}"  AGENT: "{ans}".
Reply JSON {"correct": true/false, "reason": "..."}.

Cas:
An upstream change cascaded a new value. PASS only if the agent states
     the
new value as the sole answer (not the old one, no hedging).
NEW: "{gold}"  AGENT: "{ans}".
Reply JSON {"correct": true/false, "reason": "..."}.

Abs:
An upstream change made the previous answer invalid with no
    replacement.
PASS only if the agent expresses uncertainty. AGENT: "{ans}".
Reply JSON {"correct": true/false, "reason": "..."}.
```

The ER template is present in the implementation for completeness, but ER is short-circuited before the LLM call:

```
ER:
Does the agent answer contain the gold value VERBATIM as a substring?
GOLD: "{gold}"  AGENT: "{ans}".
Reply JSON {"correct": true/false, "reason": "..."}.
```

A.4 Dataset Statistics

The MEME full ablation covers both domains: personal life (PL) and software engineering (SW). The distribution across task types and domains is not reported here because it is inherited directly from the official benchmark release (Jung et al., 2026). The STALE evaluation uses the official T1/T2 balanced release (Chao et al., 2026).

B Additional Results

B.1 Diagnostic Configuration Study

We report the STALE configurations as a diagnostic study rather than a causal component ablation. The configurations differ in their symbolic operators, but their extraction and execution paths are not fully shared. Consequently, differences in accuracy cannot be attributed solely to the added operator.

We include the study to document the observed behavior of the implementation, but do not use it as primary evidence for the causal contribution of ATMS, SAT, or kernel contraction.

The **component contribution study** on STALE (Section B.4) varies one symbolic layer at a time while holding the others fixed. It measures the marginal contribution of each layer inside the revision pipeline. Its purpose is to answer: *inside the revision pipeline, how much does each layer contribute?* The **architecture comparison** on MEME (Section B.2) varies the overall system configuration and includes the LLM-only baseline. It measures whether the complete structured memory system outperforms configurations that omit major components.

The MEME architecture comparison includes a `Cascade-only` configuration whose verified implementation delegates to the plain-LLM answering path. This configuration therefore does not isolate the formal symbolic engine and should not be read as a component ablation of the Cascade Memory mechanism. The STALE component study uses configurations whose implementations exercise the symbolic engine directly, and it is the study that supports the mechanism claim.

B.2 MEME Architecture Comparison

The MEME architecture comparison is reported in the main text (Table 2); we do not duplicate it here.

B.3 MEME Per-Task Breakdown

Table 5 reports the full per-task breakdown for the five external baselines plus Cascade Memory on the full benchmark.

Table 5: MEME per-task results on the full benchmark (100 episodes, n=694 questions). Task sizes: ER = 100, Agg = 100, Tr = 100, Del = 100, Cas = 164, Abs = 130. Bold marks the best result per column.

System	ER	Agg	Tr	Del	Cas	Abs	Overall
Cascade Memory	**100.0**	**88.0**	**92.0**	**96.0**	**96.3**	**88.5**	**93.52**
Supersede	93.0	53.0	55.0	76.0	75.0	2.3	58.07
Cognitive Sleep	61.0	46.0	67.0	46.0	13.4	0.0	42.31
MRAgent	95.8	70.8	50.0	45.8	9.1	0.0	34.87
CUPMem	92.0	47.0	0.0	67.0	12.2	0.0	32.56
Kumiho	84.0	34.0	1.0	61.0	15.9	0.0	29.68

Both baseline failure modes are visible in the per-task columns. On Absence, which requires the system to express uncertainty when no replacement rule applies, the closest baseline (Supersede) reaches 2.3% and the remaining baselines reach 0.0%. On Tracking, which requires retaining and retrieving the full history of values for a single entity, CUPMem and Kumiho both reach near-zero. On Cascade, which requires propagating an upstream change to a downstream value, the best external baseline reaches 75.0%; the remaining systems do not exceed 15.9%.

B.4 Support tracking provides the largest observed component gain

The progressive STALE study traces the performance change as the symbolic revision operators are added to the pipeline (Figure 6). Adding support tracking increases accuracy from 49.3% to 64.9%. Adding consistency checking without a contraction rule lowers accuracy to 58.5%. Adding ordered contraction recovers the loss and produces the highest score in the study, 66.7%.

The two intermediate results point to different roles. Support tracking identifies the beliefs whose justification depends on the changed state, but it does not by itself decide which conflicting support should be withdrawn. Consistency checking detects incompatible states, yet without an explicit selection rule it over-retracts. Ordered contraction supplies that rule, and the pair (consistency check, ordered contraction) performs better than either layer alone.The margin over the support-only configuration is 1.8 points and comes from a single seed, so we report the pattern as an observed trend rather than as a statistically established effect. What the sequence does show clearly is that adding

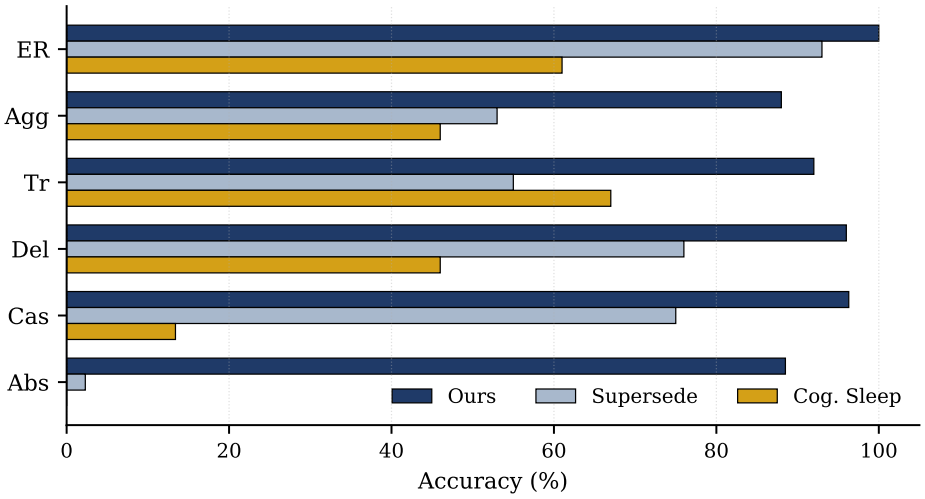


Figure 4: MEME per-task accuracy for the top three systems (n=694). The gap between Cascade Memory and the closest baseline concentrates on Deletion, Cascade, and Absence.

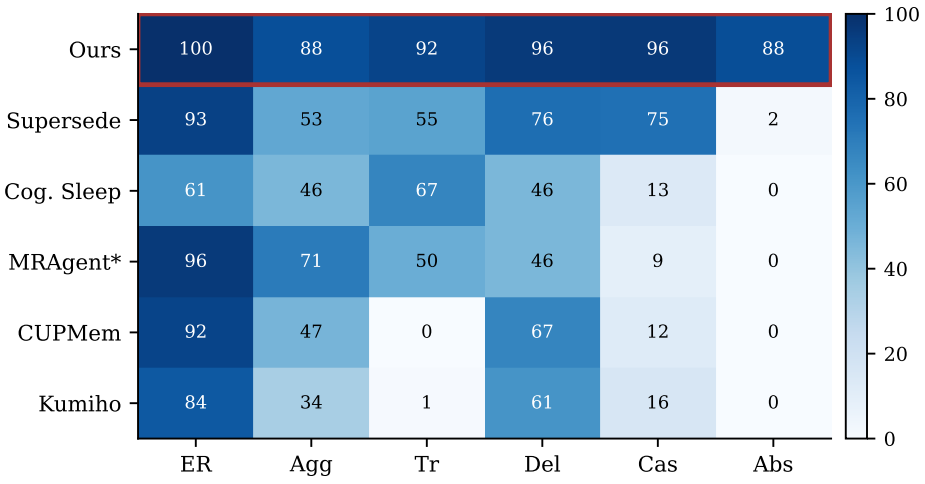


Figure 5: MEME per-task accuracy across all six systems (n=694). Red frame marks Cascade Memory.

the consistency check without the contraction rule reduces accuracy. The two layers are useful only as a pair, and the marginal effect of the pair over the support-only configuration is small.

B.5 Cross-Model Per-Task

Table 6 reports the preliminary per-task results of the four-model MEME study.

The per-task breakdown shows that the aggregate cross-model ranking masks substantial variation. Qwen3-Max, which has the lowest overall score, fails on Deletion and Cascade despite matching the best-performing configurations on ER and Absence. GLM-5.3-Flash shows the opposite pattern: lower scores on Deletion but higher scores on Cascade. The magnitude of the gain varies substantially across backbones, so the experiment supports consistency of direction rather than a model-independent effect size.

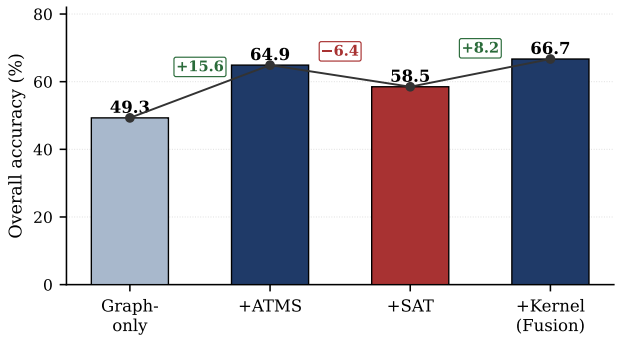


Figure 6: STALE component contribution (n=400). Each bar adds one symbolic layer. The support layer (ATMS) contributes +15.6 pp. The consistency check (SAT) alone reduces accuracy by 6.4 pp; the kernel contraction recovers the loss and exceeds the support-only configuration.

Table 6: Cross-model MEME results, n=694, Cascade Memory configuration.

Model	ER	Agg	Tr	Del	Cas	Abs	Overall
Gemini-3.6-Flash	100.0	88.0	92.0	96.0	96.3	88.5	93.52
Claude-Sonnet-5	100.0	100.0	67.0	96.0	75.6	39.2	77.52
GLM-5.3-Flash	100.0	100.0	65.0	66.0	78.0	84.6	81.97
Qwen3-Max	100.0	100.0	67.0	16.0	14.6	88.5	60.81

C Cost Accounting

Table 7 reports the cost of each external baseline on the MEME n=100 subset. Table 8 reports the STALE per-configuration cost.

Table 7: MEME baseline cost, Gemini-3.6-Flash, n=694.

System	Calls	Total tokens	Cost (USD)
Supersede	6134	27,489,327	4.3546
Cognitive Sleep	1093	213,929,226	16.0557
MRAgent	9398	632,975,673	47.8375
CUPMem	6134	11,450,414	1.1160
Kumiho	31357	11,420,515	1.3713

Cost figures are estimated from token counts using prompt pricing of $1.00 per million tokens and completion pricing of $7.50 per million tokens. Reported figures include LLM calls for extraction, adjudication, and answer generation; they exclude the judge calls used for scoring and any infrastructure cost. The figures are therefore comparable across configurations but do not represent total system cost.

D Implementation Details

D.1 Dependency Ontology

Type II conflicts are resolved by propagating an upstream change to the downstream beliefs that depend on it. This propagation is not learned. It follows a fixed, hand-specified ontology of commonsense edges between memory attributes, motivated by two design constraints. First, Type II defeat requires that the dependency be recognizable *before* the change occurs; a learned relation graph would need the very observation that triggers the cascade in order to infer the edge, which is exactly the inference we want to test. Second, a fixed ontology makes the ablation auditable: when

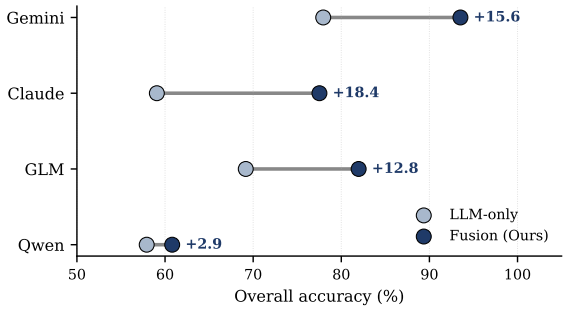


Figure 7: Cross-model behavior on MEME (n=694). Cascade Memory exceeds LLM-only in every model family.

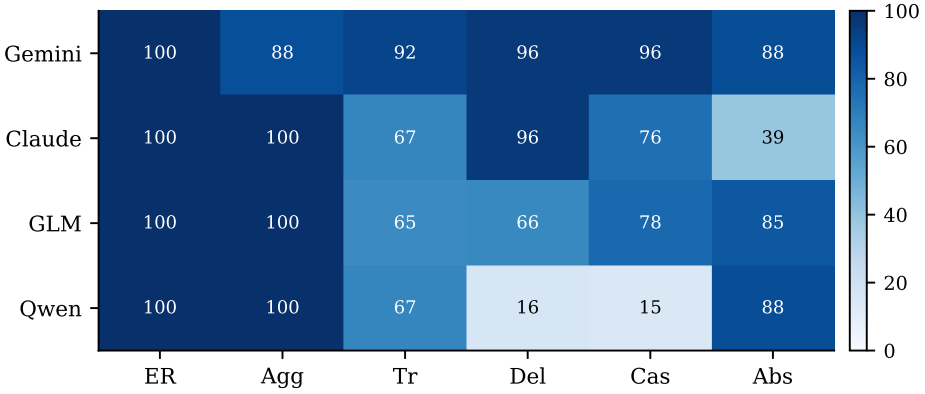


Figure 8: MEME per-task accuracy across four model families, Cascade Memory configuration (n=694).

a Type II scenario is contracted, the operator can point to the edge that licensed the contraction, and the reviewer can inspect that edge.

The ontology is deliberately narrow. It covers the health, location, employment, and routine relations exercised by the benchmark scenarios. We use a fixed ontology to isolate the revision problem from dependency discovery. Otherwise, errors in relation induction and errors in belief revision would be confounded, making it difficult to attribute a failure to either stage. Table 9 lists the complete ontology.

D.2 Extraction Prompts

The LLM extractor is the sole interface between raw conversation text and the formal layer. It receives the old and new statements and returns a structured triple $(a, v_{\text{old}}, v_{\text{new}})$, where a is the affected attribute and v_{new} is the new value. Crucially, the extractor is *not* asked to decide whether the change propagates: that decision is delegated to the formal filter in Appendix A, which admits a conflict only if a reachable defeat path exists in the ontology. This separation matters because an extractor asked to make the admission decision directly would be scoring its own work; the split lets us measure proposal quality and adjudication quality independently. The STALE LLM extractor uses the following system prompt. It is followed by a user prompt containing the old and new statements.

```
You extract structured memory facts. Given an OLD user statement and a
    NEW
user statement, output strict JSON: {attribute_b, slot_new, value_old,
```

Table 8: STALE cost per configuration, n=400.

Configuration	Calls	Cost (USD)
Cascade Memory	22,332	213.82
Graph+ATMS	18,564	162.18
Graph+ATMS+SAT	13,440	236.99
Graph-only	33,101	411.91
Cascade-only	18,113	159.62
Kumiho	15,775	196.57
Supersede	14,626	222.23

Table 9: Complete commonsense dependency ontology. Each row is a DEPENDS_ON edge $A \rightarrow B$ with strength σ, used during adjudication to weight the propagation effect.

Source A	Target B	σ	Rationale
health_and_mobility/current_health_state	routine_and_transport/current_commute_mode	0.85	injury makes the prior commute infeasible
health_and_mobility/functional_limitation	routine_and_transport/current_commute_mode	0.80	mobility limit makes the prior commute infeasible
role_and_identity/employment_status	location_and_living/current_base_location	0.70	a job change may relocate the user
location_and_living/current_base_location	routine_and_transport/current_commute_mode	0.65	a move changes which commute modes are available
finance_and_resources/financial_constraint	routine_and_transport/current_commute_mode	0.55	a budget change may force a cheaper commute
health_and_mobility/current_health_state	physical_health/caffeine_or_nicotine_reliance	0.60	a health event can invalidate a prior reliance

```
value_new}. attribute_b is the slot the OLD fact occupies; slot_new is
     the
slot the NEW statement is about (may equal attribute_b or be a
    different
slot). Use slot keys like
'routine_and_transport/current_commute_mode',
'location_and_living/current_base_location',
'role_and_identity/employment_status',
'health_and_mobility/current_health_state',
'physical_health/caffeine_or_nicotine_reliance'.
Only output JSON.
```

The supplied extractor conditions on the OLD and NEW statements. It does not pass a separate current belief state.

D.3 ATMS Representation

The ATMS layer provides structural core-retainment: a belief survives a defeat if at least one of its support environments remains viable after the defeated assumptions are removed. This is what lets the engine distinguish "the old conclusion was defeated" from "the old conclusion was defeated*along one path* but still has another". SAT layer alone cannot make this distinction because SAT only sees the conjunction of the base.

Each belief carries a set of minimal support environments $L(b)$. An environment is a finite set of assumptions whose joint assertion licenses the belief. Nogoods record assumption sets that cannot jointly hold. Given a set of defeated assumptions D, an environment remains viable if it does not intersect D and does not contain a nogood as a subset. The engine-facing ATMS label kernel uses an environment-size bound of `max_env = 6` and a label cap of `max_label = 24`. A justification is represented as:

```
@dataclass(frozen=True)
class JustV2:
    jid: str
    premises: FrozenSet[str]
    neg_premises: FrozenSet[str]
    conclusion: str
    operator: str = "AND"
    strength: float = 0.6
```

Nogoods are stored as sets of assumptions. Label propagation runs to a fixed point; for each justification, the environments of its premises are combined, filtered by the nogood check, and minimised by subset containment up to the configured environment-size bound.

D.4 SAT Encoding

The encoded belief state Γ is compiled to CNF via a Tseitin transformation. The entailment test is

$$\mathrm{Ent}(\Gamma, \alpha) = \neg\,\mathrm{SAT}\big(\Phi(\Gamma) \wedge \neg\alpha\big)$$

where $\Phi(\Gamma) = \bigwedge_{r\in\Gamma} \varphi_r$. The solver is a DPLL implementation with unit propagation and pure-literal elimination. The encoding separates observation atoms from grounding rules so that a later contraction can remove the interpretation without removing the raw observation. System axioms encode functional-slot exclusions of the form $\neg(c_o \wedge c_n)$ for incompatible values of the same attribute.

D.5 Kernel Contraction Ordering

Contraction follows kernel principles. For a target formula α entailed by the current base, an α-kernel is an inclusion-minimal subset that still entails α. The procedure shrinks the base by dropping any record whose removal preserves entailment of α; the survivor is an inclusion-minimal kernel. If α remains entailed, one eligible record is cut and the search repeats.

The incision order is:

1. cut a defeasible grounding rule first, when one is present in the kernel;
2. otherwise, cut the record with the lowest reliability, breaking ties by earlier timestamp, then by identifier;
3. protect records whose claim retains an alternative support environment not routed through the current defeater, unless every defeasible member of the kernel is protected, in which case protection is relaxed within that kernel.

A record with source class SYSTEM is never eligible for withdrawal. If every member of a kernel is a system axiom, contraction reports a protected result rather than violating the axiom.

D.6 Configuration Switches

Table 10 lists the configuration flags used in the STALE component contribution study.

The four configurations vary one operator at a time and hold the others fixed.

Table 10: Configuration switches for the STALE component study.

Configuration	`use_atms`	`use_sat`	`use_kernel`
Graph-only	–	–	–
Graph+ATMS	✓	–	–
Graph+ATMS+SAT	✓	✓	–
Cascade Memory	✓	✓	✓