# Structural Trust: An Uncertainty-Typed Property Hypergraph for Agent Memory

**Ray Deiotte** Novel Sciences and Engineering, Inc. rdeiotte@novelsciences.co

2026-09-01

## 0. Abstract

Agent-facing knowledgebases compute confidence and provenance at answer time, as properties of the response. This paper argues that is too late, and that the argument is not a matter of engineering quality: information about whether a claim is true enters through the primary record, and every later stage is a rendering of that record, so no downstream scoring can add a bit that the record did not already carry. Confidence computed from the exhibit has an evidential ceiling fixed before it runs.

We describe the store that follows from taking that seriously. The substrate is an **uncertainty-typed property hypergraph**: every unit carries the same five fields, a knowledge type, provenance graded on a four-valued lattice and carrying a validity horizon, the condition under which the entry held, a version with its conflict history, and a confidence separated into write-time and serving-time quantities, and relations are hyperedges because the relations that matter in agent memory are irreducibly n-ary and reifying them into binary edges destroys exactly the structure later stages need. On that substrate we specify three mechanisms. **Consolidation** is gated on whether a merged region still decomposes into the inferences it was meant to support, a structural and label-free test that needs no paired QA data and no per-environment training. **Conflict** is preserved through four lifecycle primitives, demotion, supersession, revocation and invalidation, of which three share a single inverse transition and revocation deliberately has none. **Retrieval** returns a coverage-guaranteed evidence subgraph with the guarantee conditional on the type of edge retrieved rather than marginal over a query distribution, and it answers with three outcomes rather than one score: supported, insufficient, refuted.

The paper's most portable result is a criterion for **when two memory mechanisms can affect each other at all**: only where one writes a property of the store that the other reads, at a granularity they share. The criterion is checkable from the specifications before anything is built, it discriminates rather than merely predicting failure, and it carries a stated falsifier.

We built the store and measured it, and report that compactly in §9, including the premises that did not survive.

## 1. Introduction

### 1.1 The observable is not the target

A knowledgebase serving language-model agents is judged by a single inference: this answer looks well-supported, therefore it probably is. Write the thing we can see as $C$ (the retrieved evidence and the answer built on it look convincing under whatever evaluation procedure we apply) and the thing we care about as $T$, that the claim is actually true. We never observe $T$. We observe $C$ and reason back.

That inference is only as good as the correlation between the two, and its strength is exactly the likelihood ratio

$$\Lambda \;=\; \frac{\Pr(C \mid T)}{\Pr(C \mid \neg T)} \tag{1}$$

whose logarithm is Good's weight of evidence. The numerator saturates: a genuinely well-supported answer will look convincing given adequate retrieval. So the evidential value of "this looks well-supported" is governed almost entirely by the denominator: by how often the system produces something equally convincing *when the claim is false*. Kamitani and Shirakawa put the point sharply: evidence lives in the denominator, and a generative system is a machine for inflating it [Phantom Evidence].

This matters for knowledge infrastructure because of where the denominator can be attacked from, and where it cannot. Information about $T$ enters through one gate: the **primary record**, the stored entry in contact with the world. Everything after that is a rendering of that record, and we call any such rendering the **exhibit**: the retrieved subgraph, the summary, the consolidated node, the answer, a score attached to the answer, a second model grading the first. So long as the chain $T \to$ primary record $\to$ exhibit $\to C$ is one-way, the data-processing inequality bounds what each stage can retain, writing $H(T)$ for the entropy of the target, the total information there is to have about whether the claim is true, and $I(X;Y)$ for the mutual information between $X$ and $Y$, the amount that observing one tells you about the other:

$$H(T) \;\geq\; I(T;\text{primary record}) \;\geq\; I(T;\text{exhibit}) \;\geq\; I(T;C) \tag{2}$$

The consequence is unforgiving: **no amount of downstream processing can add a single bit of information about the target.** Polishing an answer, raising its fluency, re-summarizing it, or having a language model score its own output are all functions placed further down the same chain. They can lose information. They cannot create it.

Almost every confidence signal in current practice is such a function. Token log-probabilities, verbalized "I am 80% confident," semantic entropy, self-consistency across resampled chains, and LLM-as-judge grades are all computed from the exhibit. The inequality says their evidential ceiling was fixed before they ran. This is not a claim that they are noisy and could be improved by better prompting; it is a claim that improvement of that kind is bounded by construction.

The empirical literature has been arriving at the same conclusion from several directions without naming the common cause. Verbalized confidence is severely miscalibrated and accuracy diverges from calibration [ConfidenceBench]. Every self-readable signal is a functional of the model's own output distribution and is therefore structurally blind to the error term that dominates at scale, with semantic entropy firing measurably *less* on the branch carrying four times more fabrications [Reliability Scales Inversely]. In production document extraction, log-probability, verbalized confidence, and five-sample self-consistency all fail as routing signals, the last at five times the cost for negligible gain, because "extraction errors are frequently caused by failures the model cannot observe" [ExtractConf]. Self-consistency checks are neutral by construction with respect to the belief that generated the samples [Prior Laundering]. A calibrated board of LLM reviewers accepts deliberately fabricated papers up to 82% of the time [Theorist Toolbox]. Each of these is a measurement of the same ceiling.

There is exactly one way to raise it: let information about $T$ enter from somewhere other than the exhibit. That is what a negative control does, what an independent verifier a fake cannot cheaply pass does, and what going back to collect new evidence does. Everything else is rearrangement.

### 1.2 The consequence for a knowledgebase

If confidence cannot be manufactured downstream, it has to be recorded upstream. That single observation reorganizes the design of a knowledgebase, because it means confidence, provenance and composability are not reporting concerns to be handled at answer time. They are properties of the stored knowledge, and they have to be captured when the knowledge is written or not at all.

We call the resulting position **structural trust**: confidence, provenance and composability should be structural, type-conditional properties of the graph rather than afterthoughts on the answer. Three commitments follow, specified in full in §7.

**Consolidation must preserve composability, not just content.** Every practical knowledgebase compresses, and compression is where information is silently lost. The failure is not that facts disappear but that individually retrievable facts stop composing, and because a consolidating system writes its own synthesized output back into itself, the confidence attached to that output is the consolidation policy's belief wearing the credibility of evidence [Prior Laundering].

**Conflict must be governed, not overwritten.** Overwrite does not merely lose signal, it manufactures certainty: storing a single best value instead of the distribution collapses the reported credible interval to

zero width on exactly the dimensions the evidence never constrained [Prior Laundering].

**Retrieval must be calibrated per type, and must be able to refuse.** A single marginal guarantee under-covers exactly the rare, high-value regions an agent reaches for when the common path has failed, and insufficiency is a distinct state from falsehood rather than a low value of the same scalar [Quiet Failure; SURE-RAG].

**Whether two of these mechanisms can affect each other is decidable before either is built.** This is the paper's fourth contribution and it is a criterion rather than a mechanism: **two mechanisms can affect each other only where one writes a property of the store that the other reads, at a granularity they share.** It is checkable from the specifications, it discriminates rather than merely predicting failure, and it is falsified by a pair that writes and reads the same property at a shared granularity and still fails to interact. §8 states the criterion against the model and reports what it accounted for. That criterion is the part of this paper we would most expect to transfer unchanged to a store built on different mechanisms.

**Why a small typed check should beat a larger model at this job.** The three commitments share one bet: that the properties worth trusting are cheaper to *certify structurally* than to *predict behaviourally*, because they are not visible in the channel the model reasons over. That bet is not ours alone. It has been measured in three unrelated fields with the same shape each time, and in all three, adding model capacity did not close the gap.

| Domain | The general model | The typed check | Cost of the check |
|---|---|---|---|
| Tool-call authorization | a 32B point judge admits **92%** of a known-unsafe witness set; a **36B judge still admits 61% in its best configuration** | an exact affine policy test admits **none of them (0%)** | **under 1 microsecond** [CAGE] |
| Best-of-$K$ causal reasoning | a 72B judge beats plurality by **+0.4 pp** $[-2.6, 3.0]$ | a symbolic axiom checker beats it by **+11.6 pp** | **~7%** over generation already paid [CALVER] |
| Biosecurity screening | a general 8B guardrail costs **11.3%** benign over-refusal; a prompt-level defence collapses domain accuracy 46.7 → 36.6 | a small domain-specific classifier reaches the same risk reduction at **1.0%** over-refusal | one encoder forward pass [SPIKE-Bench] |

The first states the mechanism outright: added judge capacity "does not provide a neighborhood certificate." The third isolates *which* capacity matters, since a same-size encoder without domain pre-training matches on risk and destroys utility, so it is the domain-specific representation and not the parameter count doing the work. We take this as the empirical case for attaching guarantees to the knowledge rather than to the answer, and, per §9, as the reason the cost of doing so is not the objection it appears to be.

### 1.3 What this paper delivers

This is a design paper. It specifies a store and three mechanisms over it, argues for the substrate rather than assuming it, and reports measurement compactly rather than as the spine.

1. **The substrate** (§4, §5, §6): what a unit carries and why each field is forced; why the relations are hyperedges rather than binary edges, and what binary reification costs; and how such a store is actually built, stage by stage, with the cost of each stage.
2. **Three mechanisms** (§7): a composability-gated consolidation, a four-primitive conflict lifecycle with a single shared inverse, and type-conditional retrieval with a three-valued refusal channel.

3. **A composition criterion** (§8): a design-time read-write test for whether two mechanisms over one store can affect each other at all.
4. **A compact evidence section** (§9) and an explicit statement of what we do not claim (§10). §9 reports experiments we ran on an implementation of this substrate, designed to test the claims made for the three mechanisms and the composition criterion rather than to benchmark the system against others. Appendix F gives the criteria each was pre-registered against.

The organizing choice is that §4 through §8 can be read as a specification, without consulting a single measurement of ours, and that a reader who disagrees with our evidence can still take the substrate. What the measurements are good for is saying which parts of the specification we would now write differently, and §9 reports that, including two premises we started from that did not survive.

## 2. Three failures that motivate the substrate

The design is a response to three measured failures. Each is stated here in the form that constrains the architecture, and each does double duty: it motivates one of the mechanisms in §7 and, because all three are failures of *binding* rather than of storage, it also motivates the substrate those mechanisms need. §5 makes that second argument; this section supplies the failures both rest on.

**Compaction does not fail gracefully, and it destroys inferences before it destroys facts.** Multi-hop accuracy swings by more than forty points at matched atomic accuracy, and failure persists even when the facts are supplied in context: 88.9% failure in the in-context condition, with chain-of-thought recovering 70–75% [Composition Collapse]. The bottleneck is generation-time assembly, not storage, so a gate that checks whether facts are *present* is checking the wrong property. Nor is the loss smooth. In a controlled probe, ten facts held only in a conversational channel were absent from 106 of 108 forced compactions while the same facts injected from an externalized store arrived at 138 of 138, and the channel behaved all-or-nothing: no summary ever carried a strict subset of the ten [Delivery Not Storage]. Compression drops blocks whole rather than degrading toward a smaller answer set.

**Overwrite is a corruption surface once there is more than one writer.** In a single-writer system overwrite is merely lossy. In a knowledgebase written concurrently by humans, software and agents, eager synchronous deduplication destroyed 51% of the contradictions the system existed to detect, while per-element provenance gave complete reconstruction of the write chain [Governed Shared Memory]. The information-theoretic version is worse: a consumer of an overwritten node cannot tell the difference between a value the evidence pinned down and a value the deduplication policy happened to keep, which is the indistinguishability of §1.1 arriving through the write path.

**A marginal guarantee collapses exactly where it is needed.** Marginal conformal prediction guarantees coverage on average over the query distribution, which it achieves by silently under-covering rare subgroups: as imbalance grows, minority coverage falls from 90% to **4.2%** [Quiet Failure]. Those subgroups are precisely the rare, high-value edges an agent reaches for when the common path has failed. A guarantee that holds in aggregate and collapses where it is needed is worse than no guarantee, because it is trusted.

These three are not independent complaints about implementation quality. They are the same fact about information appearing at three points in a store's lifecycle: at merge, at write and at read. That is why the response is a substrate rather than three patches.

## 3. Related work

The design commitments of §1.2 each have close published neighbours, and several of the ideas we build on were named by others first. This section places each contribution against its nearest work precisely, since that boundary is what determines whether our measurements are worth making.

### 3.1 Routed and typed memory

Routing memory is not our idea and we do not claim it. Routing is proposed as future work in the paper that frames the trilemma [MemFail], and a routed agent-memory architecture is published [Supra Cognitive

Modes]. Typed memory banks (core, episodic, semantic, procedural) are standard practice [AttriMem], and a benchmark now organizes 193 retrieval tasks along exactly that content taxonomy [LMEB].

Our routing differs in object and in what gets measured. Published routers select over *procedures within one substrate* on the basis of query shape, with a frozen classifier, explicitly deferring efficiency, latency, and any fixed-policy comparison [Supra Cognitive Modes]; or over *model backends* at task granularity with latency in the reward [TRACE-ROUTER]; or over *entries within one encoder* [EAR]. We route on **knowledge type over genuinely heterogeneous storage backends** and report the frontier against single backends, a fixed policy, and a latency-matched random mixture. Our taxonomy is also a different kind of object from LMEB's: theirs classifies what a memory *is*, ours predicts which backend *wins*, and Appendix F states the criterion that kills any axis failing to earn oracle lift.

### 3.2 Consolidation and promotion criteria

Explicit promotion gates exist, and the framing that "nobody specifies the promotion criterion" is no longer accurate. A learned budget-conditioned utility regressor gates retain-versus-consolidate [Retain-or-Consolidate]; a reinforcement-learned policy selects extract, store, update, compress, and discard actions using token-level attribution [AttriMem]; a state-conditioned controller learns over atomic evidence operations [DynaKRAG]; and a per-item keep/drop policy gates insertion into a knowledge graph specifically, the closest setting to ours [Short-Term-to-Long-Term Transfer]. Two of these transfer across backbones without retraining, better than a training-free story would predict, and we say so.

Being label-free is not by itself the difference. Structural, unlearned gates do exist: a hard-conflict guard that vetoes a merge on any conflicting discriminator attribute regardless of similarity score [Ontology-Guided Dedup], a frequent-itemset admission rule with a support and all-confidence floor [AdaMM], and a similarity-dedup admission with utility-counted eviction [SESA]. The first of those *dominates* a learned similarity score rather than supplementing it. So the field is not uniformly learned, and saying it was would be overreach.

Our gate differs on **what it asks, and when**. The published gates (learned and unlearned alike) score a candidate *before or instead of* merging: predicted utility, attribute conflict, co-occurrence recurrence, observed downstream helpfulness. Ours is a **post-merge decomposability test**: it asks whether the merged region can still be decomposed into the inferences it was supposed to support. No published gate asks that question. That it also requires no paired QA labels, no reward model, and no environment to train in is a deployment consequence of the choice, not the claim itself. §7.1 gives the mechanism, §9 the comparison at matched budget, and **Appendix F's E1d criteria the test of that distinction rather than an assertion of it**, including against the fidelity-style objective that the closest published comparison [ThinkReset] tried and abandoned.

Context management as a *lifecycle* rather than a store, and the demand that compaction be validated rather than trusted, are both stated before us [ACM]. That work also names the gap between retrieval success and reasoning sufficiency, and reports it as "an observation rather than a result," because its scoring could not quantify it, with its validation mechanism undisclosed. The observation is theirs; the instrument is ours.

### 3.3 Sufficiency, calibration, and abstention

Evidence sufficiency as a decision layer is published, with the framing we would have used: relevance is not sufficiency, and a system needs a layer asking not whether evidence is relevant but whether it suffices [SUFE-RAG]. That work supplies the three-way label we adopt (supported, refuted, insufficient) and the argument that sufficiency is a *set-level* property, so pairwise pooling is brittle in two opposite directions: max-pooling over-answers when a required hop is missing, mean-pooling dilutes a decisive refutation. Sufficiency checking as an operation that emits an actionable gap description rather than a score is likewise published, with an ablation showing it is load-bearing [DynaKRAG].

Neither is conformal, and neither is conditional on type. Reliability varies by evidence type and calibration by question type, but the published treatments calibrate *marginally* [Calibrated Selective Fact-Checking; ConfidenceBench]. Marginal conformal prediction is precisely what fails on the rare regions an agent needs, with minority coverage collapsing from 90% to 4.2% under imbalance [Quiet Failure]. The factorization

argument has been made along three different axes (decision axis, signal source, and information state) each beating a single thresholded score by a measured margin [Two Axes of Abstention; ExtractConf; CALIBER]. We note that these three are reasonably independent in domain and model but that their support does not add additively, for the reason given in §1.1.

**Per-edge-type conformal coverage over a graph** is, to our knowledge, unclaimed: a guarantee conditional on the type of the retrieved edge, with the coverage gap against the marginal guarantee as the headline measurement. §7.3 states the mechanism and Appendix F the criterion that would kill it.

### 3.4 Conflict, governance, and delivery

Conflict preservation over overwrite is validated empirically (eager deduplication destroyed 51% of the contradictions the system existed to catch, and per-element provenance gave complete chain reconstruction [Governed Shared Memory]) and independently required by a deep-research agent whose state must retain provenance, negative evidence, unresolved constraints, and rejected candidates [AREX]. Rate governance of eviction is imported from a cautionary result in the KV-cache domain, where ungoverned fill-then-evict loops cascade into a limit cycle [Eviction Cascade], and sharpened by a formal treatment in which the governed quantity is the *ratio* of uncertified to certified commitments, past which recovery is hysteretic rather than symmetric [Recoverability Collapse]. Our contribution here is the four-primitive decomposition (demotion, supersession, revocation, invalidation) not the framing, which the rate-distortion compaction literature already supplies [Compaction Survey].

One result reorganizes our access-contract design rather than supporting it. With a store pre-seeded with task-relevant notes, connected tools, and explicit instruction, an agent made **zero** memory calls across 114 turns; knowledge present in a store contributes nothing by itself, and a memory system that adds material and relies on the model choosing to use it inherits the model's compliance failure rate [Delivery Not Storage]. §7.4 treats delivery as part of the access contract for this reason: a coverage guarantee over a subgraph that is never requested is vacuous.

## 4. The unit: what a single entry carries

### 4.1 What a unit carries, and why each field is forced

The substrate is an uncertainty-typed property hypergraph. §5 argues for the hypergraph specifically, against the binary-edge alternative; this section describes what a single unit carries, independently of how units are connected.

**Every node and hyperedge carries the same five fields.** None is a convenience. Each is forced by an argument in §1.1, and the table names which one.

| Field | What it holds | Why it cannot be dropped |
|---|---|---|
| **Type** | the knowledge type, from a fixed taxonomy | conditions storage on the write side and calibration and routing on the read side; the guarantee in §7.3 is conditional on it |
| **Provenance** | writer, source, timestamp, an **epistemic grade** on a four-valued lattice, and a **validity horizon** | a consolidating store writes its own output back into itself, and without a grade nothing distinguishes what was observed from what was rendered |
| **Validity condition** | the state under which the entry was true | not the same as where it came from; it is what makes transitive invalidation computable (§7.2) |

| Field | What it holds | Why it cannot be dropped |
|---|---|---|
| **Version and conflict history** | the ordered record of what this unit has been, including retirements | supersession must retire a value without erasing it, and the history is what makes a retirement auditable and reversible |
| **Confidence** | two quantities, not one | write-time and serving-time confidence answer different questions; §4.3 gives the argument |

The rest of this section takes the fields in that order and says what forces each. Confidence is deferred to §4.3 because its argument needs the whole field set in place.

**Type** The type is drawn from a taxonomy that does double duty: it conditions storage on the write side and both calibration and routing on the read side (§9). The justification for typing at all is empirical and now unusually broad. Reliability varies by evidence type [Calibrated Selective Fact-Checking], calibration by question type [ConfidenceBench], and whether truth-related behaviour is one axis or several is a property of the model rather than a modelling choice: factual and opinion sycophancy share a direction in Gemma-3-12B (steering vectors at cosine +0.84) but sit near-orthogonal in Llama-3.1-8B (cosine −0.11), and where they separate, **collapsing them to a single axis costs control**: subtype-aware steering reaches a sycophancy rate of 7–9% where one combined vector floors at 11–14% [Sycophancy Dissociation]. The effect reaches the systems layer too: whether a stored unit is a metadata-rich memory record or a raw passage changes KV-reuse behaviour, not merely accuracy [AgentKVShift]. Type is a structural property of knowledge, not a label applied to it.

**Provenance** The grade is four-valued: **declared** (authority without evidence: a convention, a policy, a chosen $\alpha$), **observed** (recorded with external provenance), **derived** (produced by a typed computation over other entries), and **verified** (carrying a re-checked proof term). An earlier version of this model used a binary observed-versus-synthesized marker, and it was under-resolved in two ways: it collapsed *derived* and *verified*, which carry different warrants, and it had no slot at all for *declared* content, which a multi-principal store accumulates constantly and which, lacking a grade of its own, is written as though it were observed. The **horizon** records how long the warrant stands before it must be re-established, because a provenance tag has a measurable validity period rather than an indefinite one.

**Why the grade is not bookkeeping.** A consolidating graph writes its own summaries back into itself and then serves confidence over them; by the data-processing inequality that confidence is bounded by what the primary record held, and on directions the sources never constrained it is the *consolidation policy's* prior wearing the credibility of evidence [Prior Laundering]. Without a marker separating what was observed from what was rendered, there is no way to tell which case a given confidence is in, and no downstream check on the stored data can recover the distinction, because the two are identical as far as the record is concerned.

Where the four grades come from, and why the grade is an invariant rather than a note. A typed knowledge-graph DBMS built for the same audit question stratifies every resource the same four ways, and enforces the grade as a *strict commit-time invariant* rather than a recorded property, which it describes as turning "data provenance into a structural invariant rather than a property reconstructed across subsystem boundaries" [Eigenius]. That is the stronger form of what the lattice above adopts, and it is the reason the two defects of the binary marker we replaced are defects of *kind* rather than of resolution: a marker that collapses derived and verified, or that has no slot for declared content, cannot be repaired by adding a third value. Their justification is the one that convinces: a scientific claim's leaves are "observations and conventions (an instrument reading, a chosen $\alpha$), that no prover can discharge," so the grade has to record what stands behind each leaf.

What that buys is not hypothetical. Re-computing a published *Nature* study from content-addressed, hash-pinned source data, all **52 of 52** derived conclusions held, and the recomputation surfaced **four machine-checked discrepancies in the original**, among them a corrected sample size where rows had been silently dropped as missing, and a pseudoreplication whose honest $p$-value differs from the published one by **thirteen orders of magnitude**. That is structural provenance finding real errors in published work, and it is the clearest argument we have for why this field is not bookkeeping.

Two production systems bracket this field usefully. One records, *per alias rather than per document*, which mechanism produced it (model extraction, algorithmic generation, or text mining) alongside an occurrence count and a confidence, which is the observed/synthesized distinction kept at exactly the granularity argued for here [Ontology-Guided Dedup]. The other is its complement: every extracted attribute carries a pointer to the supporting span, the extraction prompt instructs the model to retain source-faithful values and not to infer facts absent from the input, and nothing synthesized is ever written back [AdaMM]. That contrast is the argument in miniature. A store that only ever records observations does not need this field. A *consolidating* store does, and the moment it writes a summary back it has taken on the laundering problem whether or not it has a marker for it. Two cautions on populating it: three protocol-frozen scans of deployed tool pipelines found provenance documented in **0 of 31** cases [CAGE], and a published proof carries an AI-use statement crediting a model with three of its lemmas while noting the authors cannot be sure what the model contributed, "since our results are probably part of OpenAI's training data at this point", provenance collapsing at precisely the point where attribution matters.

Provenance also has a fault model, which we adopt rather than treating the field as a value that is simply present or absent: a stale tag, a confused policy pack, a schema-version mismatch, and read-then-act races are the single-fault cases a certificate must tolerate, while schema transposition and key collision leave no discrete trace and fail validation instead [CAGE]. We add one fault of our own to that list, because it is the one this design is most exposed to: **identity drift**, where the same thing is written twice under two identities. Key collision is its mirror image, two things sharing one key, and collision at least fails validation; drift produces two individually valid records and no signal at all. §6.1 stage 0 is the response. And it has a *half-life*: same-entity staleness crosses the tolerated margin at roughly ten seconds in that setting, with system-level false-allow growing 1.8% to 7.1% as the freshness budget relaxes while the certified rate stays at zero. The validity horizon specified above is what this argument requires a provenance record to carry, and it is why the field is there rather than optional. **What we have not built is the other half: a demotion rule that keys off the horizon automatically**, so that an entry whose warrant has expired is demoted without waiting to be queried. We flag that as a mechanism change rather than assuming it.

**Validity condition** The validity condition records the state under which the entry was *true*, which is not the same as where it came from. Provenance answers *who said this and on what basis*; the validity condition answers *what had to hold for it to be so*. The distinction is load-bearing rather than taxonomic, in the system that motivated it, removing the source observation costs 5.2 points of success rate, and *randomising* it rather than removing it raises the rejection rate from 8.7% to 13.3% on one benchmark and 56.0% to 63.3% on another, so the gate is genuinely comparing states rather than checking that a field is populated [MemHarness]. The validity condition is also what makes transitive invalidation computable (§7.2): a derived entry whose supporting condition no longer holds is exactly the thing to withdraw.

**Version and conflict history** A unit's version order lets supersession retire a value without erasing it, and the retained conflict history is what makes that retirement auditable and, for the three reversible primitives, undoable. Both are specified in §7.2. They are stored on the unit rather than derived at read time because §6.3 keeps them off the query path deliberately: they are audit structures, consulted when a decision is questioned rather than on every serve.

**Confidence** One field cannot carry both quantities, and the reason takes a subsection of its own (§4.3).

### 4.2 Units assert; they do not warn, and one type must

Every field above describes a unit that makes a claim. A store built only from claims cannot represent *this path was tried and it failed*, and that is not a hypothetical gap: the external instances are now independent and specific. A skill card carries **avoidance cues** (anti-patterns and common confusions) as first-class content [SESA], and a long-horizon runtime retains rejected routes and a **falsified route** as reusable artifacts rather than discarding them [Argus]. Both are storing knowledge whose value is that it forecloses work, and our schema has nowhere to put it.

We therefore add a second unit kind, a **caution**, alongside the assertion. It reuses the node schema unchanged and differs in three respects, each of which was a blocking question before it was a specification. A caution is returned alongside an answer, never instead of one. A caution attaches to the evidence subgraph as a typed edge when its validity condition holds and it shares a binding path with what was retrieved. A caution is not an abstention trigger and it does not suppress: a store that can silently veto an answer has given itself an unfalsifiable power, and a consumer that receives both a claim and a suppression cannot tell which the system believes. Precedence is therefore fixed and dull: the answer is returned, the caution rides with it, and C3's three outcomes are untouched.

**It carries a count, not a confidence, and this is the constraint we would previously have got wrong.** "We are sure this failed once" is not "this will fail," and the two cannot share a scale. A caution reports how often and how recently the path failed **under a stated condition**, and it is deliberately kept off C3's coverage scale rather than calibrated onto it. That restraint is load-bearing rather than stylistic: §10 reports that our own attempt at a second calibrated channel failed twice and then failed a fairer test as well, so a design that quietly assumed a working second scale would have been assuming something we have now measured ourselves not to have.

**It ages by condition rather than by time, which is where the schema starts doing real work.** A path that failed under circumstances that no longer hold must stop warning, or the store accumulates advice that is true about the past and wrong about the present. This is the first unit kind for which `validity_condition` is load-bearing rather than bookkeeping: when the condition lapses the caution is **invalidated** (§7.2) and leaves the active view without being deleted, and if the condition recurs it is **re-promoted** under its original identity. Negative knowledge is thus the type where three separately motivated decisions (the validity condition, invalidation, and re-promotion) become one mechanism rather than three fields.

What we are not claiming. No experiment in §9 evaluates cautions; E2's design would have to be extended to cover them, and we mark that rather than implying coverage we do not have. The retrieval contract above is a specification, not a measured result.

### 4.3 Confidence is two quantities, not one

The natural design writes one `confidence` number per node. That is a category error, and the argument is about information states.

Confidence estimated *before* a reasoning step and *after* it are different quantities computed from different information: the first predicts whether the system can answer at all, the second whether the answer it actually produced is correct. Supervising both against one target creates a mismatch between what is estimated and what the estimate is trained to predict; matching each to its own information state cuts calibration error by half [CALIBER]. The same split applies to storage, where the two information states are write time and read time:

- **write-time confidence** $c_w$ on a node or hyperedge: given this content and its provenance, how likely is it to support future inferences? Supervised group-wise, over the population of items sharing its type and provenance class.
- **read-time confidence** $c_r$ on an assembled evidence subgraph and the answer built from it: is *this* inference correct, given what was retrieved? Supervised instance-wise, and carrying the conformal guarantee of §7.

Keeping both is cheap and buys a free diagnostic. A persistent divergence between an edge's $c_w$ and the $c_r$ of answers that used it indicates either that the knowledge has gone stale or that its write-time estimate was miscalibrated, either way, a demotion candidate for §6. Collapsing the two fields discards that signal.

We take one instruction from [CALIBER] and decline another. The position-target alignment principle transfers. Its instrument does not: it calibrates a *verbalized self-report*, which §1.1 places downstream of the exhibit and therefore bounded regardless of how well calibrated it becomes.

### 4.4 Where confidence may come from

The model constrains its own confidence fields. By the chain of §1.1, a value for $c_w$ or $c_r$ is admissible only if it is computed from information that does not already factor through the exhibit. In practice that permits four sources and excludes one large family.

Permitted: the primary record and its provenance; a mechanical check that consults something outside the model, such as executing code, running a query, or verifying a signature; an independent verifier a fabrication cannot cheaply pass; and a measured negative control, which is the only one of the four that quantifies a denominator directly.

Excluded: any functional of the generator's own output distribution, token log-probabilities, verbalized confidence, semantic entropy, self-consistency across resampled chains. This exclusion is scoped rather than absolute, and the scope matters. Treated as a readout of a latent decision variable, the answer-logit difference does satisfy the qualitative signatures of statistical decision confidence in well-specified single-step decisions, and abstention on it is near-optimal there [Computational Basis of Confidence]. The signatures break down in complex reasoning. That boundary is where a knowledgebase lives: multi-step composition, source-grounded retrieval where the failure is in the evidence and therefore invisible to the generator, and iterated re-encoding. Inside that regime the exclusion holds; outside it, generator-internal confidence is a legitimate signal and we do not pretend otherwise.

## 5. Why a hypergraph, and what a binary graph costs

Most knowledge graphs are binary. A triple store holds subject, predicate, object, and an $n$-ary relation is represented by *reification*: introduce a node standing for the relation and attach the participants to it with $k$ binary edges. Reification is expressive enough to represent anything, which is why the question is usually treated as settled. Expressiveness is not the issue. The issue is what the store can still *check* after the transformation, and reification moves the $n$-ary structure out of the type system and into a convention that nothing enforces.

This section makes the case that the three mechanisms in §7 are not implementable over a binary substrate without giving up the property that makes each of them worth having. Plain reification is the weakest version of that substrate and we do not rest the argument on it: §5.4 takes the case against quoted triples, named graphs, property graphs and singleton properties, concedes what each of them fixes, and isolates the one thing none of them enforces. §5.5 states what the hypergraph costs and §5.6 says when a binary graph is the better choice.

### 5.1 The relations that carry agent memory are irreducibly $n$-ary

Figure 1 sets the two encodings side by side. Three shapes recur, and none of them is a pair.

**A measurement binds an instrument, a subject, a value and a time.** Drop the instrument and the value is unattributable; drop the time and it cannot be superseded correctly. The four are jointly meaningful and individually inert. This is the canonical case and it generalizes to anything with a method: an extraction binds an extractor, a document, a span and a schema version. That signature does double duty, because it is also the **identity** of the extracted unit when the source supplies no stable key of its own: it is what makes re-extracting the same span with the same extractor produce the same unit rather than a second one (§6.1 stage 0).

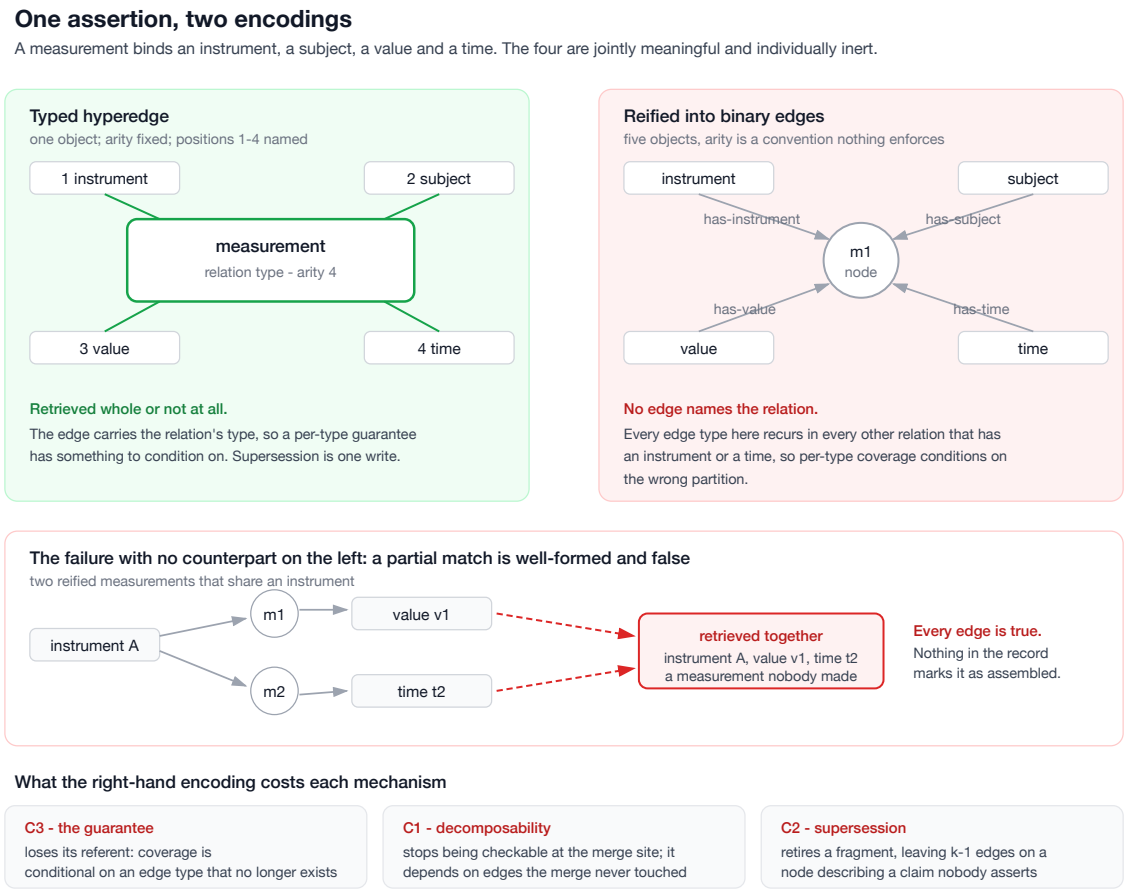


Figure 1: **one assertion, two encodings.** The same measurement as a single typed hyperedge and as a reification into binary edges. The right-hand encoding preserves the extension of the relation and discards the fact that these four participants belong together as one assertion, which is the information each mechanism in §7 consumes. The lower panel shows the failure with no counterpart on the left: a partial match across two reified relations that share a participant is syntactically valid, individually true edge by edge, and false.

**A claim binds a source, an assertion, a scope and a validity condition.** §4.1 requires every unit to carry a horizon, and a horizon is not a property of the assertion alone. "This holds until the policy is revised" is a relation between the assertion and the condition under which it stands. Represented as an attribute of the assertion node, it silently becomes a property of *every* use of that assertion, including uses whose scope the original source never covered.

**A merge binds its constituents, the policy that merged them and the reasoner version that licensed it.** §7.1 requires consolidation decisions to be re-checkable against the reasoner that will be asked to use the result, which means the merge itself is a first-class object with at least three participants. A binary edge between two merged entities cannot express which policy produced it or under what reasoner it was valid.

The third case needs one clarification, because it looks like a counterexample to the fixed arity the rest of this section relies on. A merge of three units and a merge of seven are the same relation type, so its arity cannot vary with the number of constituents. We represent it at arity three, with the first position bound to a single set-valued unit standing for the merged region. That keeps arity fixed, and it is an honest cost rather than a free move: a set-valued position does not give the plan-time selectivity §5.5 claims for the others, so merges are indexed by their policy and reasoner positions and the constituent set is scanned.

In each case the relation has a *type* and a fixed *arity*, and the participants occupy named positions. That is precisely the signature of a typed hyperedge, and it is what reification discards.

### 5.2 Four things a binary encoding cannot check

The costs below are not performance costs. Each is a property that §7 relies on and that reification makes unstateable.

**The type-conditional guarantee loses its referent.** §7.3 conditions coverage on the type of the edge being retrieved rather than marginally over a query distribution, and that is the whole mechanism: it is what keeps the guarantee from collapsing on the rare cell. Under reification there is no edge whose type names the relation. A measurement becomes a node plus four edges typed `has-instrument`, `has-subject`, `has-value` and `has-time`, none of which is the relation, and each of which appears in every other relation that has an instrument or a time. Conditioning coverage on those types conditions on the wrong partition. One can condition on the *reification node's* type instead, but the guarantee is over what retrieval returns, and retrieval returns edges.

**Partial matches become well-formed and false.** Retrieve three of a measurement's four edges and the result is syntactically valid and semantically a chimera. Worse, retrieve edges from two different measurements that share an instrument and the store will assemble a composite that was never asserted, with nothing in the representation marking it as assembled. A typed hyperedge is retrieved or not retrieved; there is no partial state to misread. This is the failure mode that makes the difference concrete: the binary encoding admits subsets the source never licensed, and the admission is invisible because every individual edge in it is true.

**Decomposability cannot be stated at merge time.** §7.1's gate asks whether a merged region still decomposes into the inferences it was meant to support. An inference is an $n$-ary object, so the question is about hyperedges. Under reification the same question becomes "does this set of binary edges still admit the same set of valid groupings," which is not locally checkable at the merge site: whether a grouping survives depends on edges that the merge did not touch. The gate would have to re-derive the $n$-ary structure it was denied, over the whole region, on every merge.

**The lifecycle operates on fragments rather than claims.** §7.2 supersedes at the granularity of a claim, and retains conflict history so that supersession is reversible. A claim is $n$-ary. Superseding one binary edge of a reified claim leaves the other $k - 1$ edges pointing at a reification node that now describes a claim nobody asserts, and there is no single object whose version history records what happened. Supersession over a hyperedge is one write with one history entry.

Each of the four traces back to the same thing. Reification preserves the *extension* of a relation while

discarding the fact that a particular set of participants belongs together as one assertion. That is exactly the information the mechanisms consume.

### 5.3 Binding information, and why it is an analogy rather than a proof

There is a name for the missing quantity, and it comes from vision rather than from databases. Formalizing the binding problem, [Binding Information] defines **binding information** as the information a representation carries about *which features belong to the same object*, over and above the information it carries about the features themselves, and gives an information-theoretic probe for measuring it in a model's representations. Their setting is multi-object scenes and vision transformers, and their finding is that binding is a distinct and often deficient capability rather than a free consequence of good feature encoding.

**We use this as an analogy and label it as one.** Nothing about visual feature binding entails anything about knowledge graphs, and we do not claim a transfer of their results. What transfers is the *decomposition*: the observation that "which things belong together" is a separate quantity from "what the things are," that it can be measured, and that a representation can be strong on the second while being weak on the first. A binary encoding of an $n$-ary relation is precisely a representation that retains the features and discards the binding. Their formalization gives us a principled way to say which inferences a region supports are worth defending, by weighting each inference by the binding information its composition requires, and §7.1 uses that weighting in the consolidation objective. The weighting is the borrowed part. The claim that it is the right weighting for a knowledge store is ours, and it is a design commitment rather than a measured result.

### 5.4 What the modern alternatives fix, and what they do not

Plain reification is the weakest form of the binary encoding, and a practitioner would not propose it. Four better answers exist, and the argument above has to survive them rather than skip them. Each recovers something real. None recovers the same thing.

**Quoted triples (RDF-star)** make a statement addressable as the subject of another statement. This directly answers the identity half of §5.2: the relation now has something to name, so it can be annotated, dated and typed. What it does not give is arity. A quoted triple is still one binary statement, an $n$-ary relation is still $k$ of them, and nothing in the model requires that all $k$ be present or forbids a subset from being retrieved together. The chimera of Figure 1 is unaffected.

**Named graphs and quads** are the strongest alternative, and they genuinely answer §5.2's lifecycle objection: give the whole assertion a graph name, supersede the name, and you have one write with one history entry, which is exactly the property we claimed only a hyperedge provides. We concede that point. What a named graph is not is a *typed relation with named positions*. Membership is open by default, so nothing in the substrate prevents a graph that is missing a participant or carries an extra one, and the edges inside it remain `has-instrument` and `has-value`, so the type-conditional guarantee of §7.3 still has no relation-typed edge to condition on. A named graph gives you a container; it does not give you a signature.

**Labelled property graphs** give edges identity and properties, which makes an edge addressable and lets arity be recorded as an edge property. But edges remain binary, so an $n$-ary relation is still a hub node with spokes. This is reification with better ergonomics and better tooling, and the four objections apply to it in the same form.

**Singleton properties** give each statement instance its own predicate, which makes it addressable at the cost of an unbounded predicate vocabulary. That cost lands precisely where this design cannot afford it: per-type conformal calibration needs enough observations per cell, and a vocabulary in which every statement is its own type is a partition with one sample per cell.

**The common gap, stated so it can be argued with.** All four recover *addressability*, and one of them recovers *identity for the whole assertion*. None enforces *arity at write time*. The commitment in §6.1 stage 4, that a relation is written whole with all positions bound or is not written at all, is the one none of these substrates makes on your behalf, and it is what removes the partial state that §5.2's second and third objections exploit. A reader who needs only the lifecycle property should use named graphs,

which are cheaper and far better supported. A reader who needs the type-conditional guarantee and the decomposability gate needs the signature, and that is a hypergraph.

### 5.5 What the hypergraph costs

Four costs, stated so that a reader can decide against us.

**There is no canonical adjacency, so indexing is different work.** A binary graph has one natural index, the adjacency list, and query planners are built around it. A hypergraph is stored as an incidence structure: for each hyperedge, the participants and their positions; for each unit, the hyperedges it participates in and in which position. In practice we key incidence by `(relation-type, position)`, which recovers most of the selectivity a binary index would give, because the fixed arity of a typed relation means position is known at plan time. Adjacency queries that a triple store answers with one index probe take two here, one to the hyperedge and one back out.

**Subset queries are a trap.** A $k$-ary hyperedge has $2^k - 1$ non-empty participant subsets, and a query language that lets a caller ask about arbitrary subsets invites the planner to materialize them. We do not support arbitrary subset matching. A caller either matches a relation type with some positions bound, which is a single index probe, or retrieves the hyperedge whole. That restriction is what keeps the cost linear, and it is a real expressiveness loss relative to a triple store with a general pattern language.

**The tooling is thinner.** Binary graph stores are mature, and a hypergraph implementation gives up a large body of engineering: query optimizers, standard query languages, and off-the-shelf operators. We regard this as the most serious practical cost and the least interesting intellectually.

**Arity is a schema decision that is expensive to revise.** Adding a position to an existing relation type is a migration over every hyperedge of that type, whereas a binary store absorbs a new attribute as new edges with no change to the old ones. The hypergraph front-loads a decision that the binary encoding defers, and if the schema is genuinely unstable that is the wrong trade.

### 5.6 When a binary graph is the right choice

The argument above is conditional, and the condition is worth stating plainly, because a reader whose situation fails it should not adopt this design.

A binary graph is sufficient, cheaper and better supported when the relations really are binary, which is the common case for encyclopedic knowledge graphs built from subject-predicate-object triples; when nothing downstream needs a guarantee conditional on relation type; when consolidation is not gated on decomposability, either because the store does not consolidate or because content preservation is the accepted standard; and when there is a single writer, so that conflict is lossy rather than corrupting. Take away the three mechanisms of §7 and the hypergraph is an unforced complication.

What makes the hypergraph load-bearing here is not that it represents more. It is that each of the three mechanisms consumes the binding structure directly, and a substrate that discards it turns each mechanism into an approximation of itself.

## 6. Construction: how the store is actually built

This section walks the write path and the read path in order, names what each stage decides, and gives the cost of each. This construction is the part of the design most likely to be reused independently of the three mechanisms, because a store built this way is useful even to a reader who rejects §7.

### 6.1 The write path

Figure 2 gives the whole pipeline; the stages below name what each one decides.

**Stage 0, identity.** Before anything else the item is assigned an identity, and the rule is the one Stage 1 applies to type: **identity comes from the source's stable key, never from a name the model chose.**

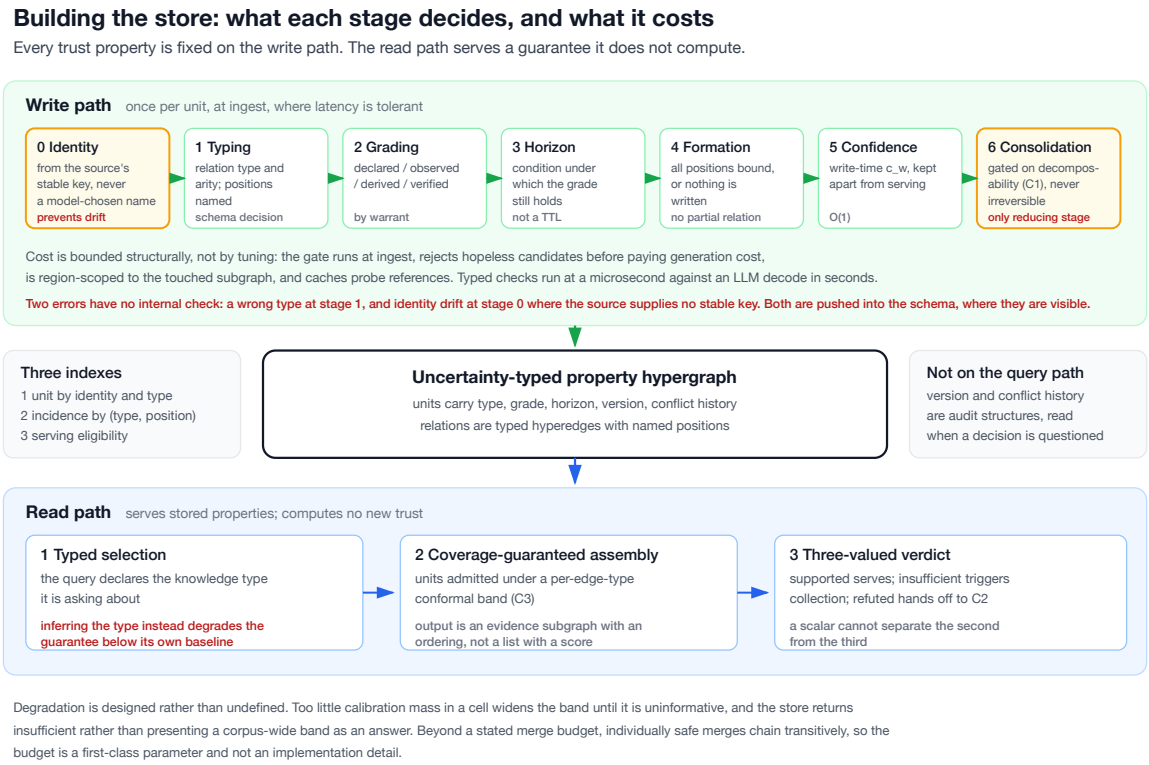


Figure 2: **building the store.** Every trust property is fixed on the write path, so the read path serves a guarantee it does not compute. Stage 6 is the only stage that reduces the store and the only one carrying a gate rather than a heuristic. Stages 0 and 1 are the two the store cannot check internally, which is why both identity and type are taken from the source's schema rather than chosen by the model.

A document identifier, a row primary key, a URI, an instrument serial with a timestamp: whatever the source already uses to mean *this same thing again*. Where the source offers no such key, the identity is the tuple §5.1 says an extraction binds, its extractor, document, span and schema version, which is stable under re-extraction of the same span by the same extractor. What is never admissible is an identity derived from the content, because a content-derived identity turns two writes of the same fact under different phrasings into two units, and **no primitive in §7.2 can bring them back together**: supersession is retire-on-conflict, and two entries that never met are never in conflict.

**Stage 1, typing and arity.** An incoming item is assigned a knowledge type and, if it is a relation, a relation type with a fixed arity and named positions (§5.1). This is the only stage that requires a schema decision, and it is deliberately front-loaded: §5.5 notes that revising arity later is a migration. Types come from the domain, not from the model; the store does not infer a relation type from text, because a mis-typed relation silently mis-conditions every guarantee downstream in §7.3.

**Stage 2, grading.** Every unit receives a position in the four-grade provenance lattice at write: declared, observed, derived or verified (§4.1). The important rule is that **the grade is assigned by the writer's warrant, not by the content's confidence**. A precise value from a model with no external source is `derived`, not `observed`, however confident it is. Absent an explicit warrant the grade defaults to `declared`, which is the conservative choice because declared content is what a multi-principal store accumulates silently and what, lacking a grade of its own, would otherwise be written as though it had been observed.

**Stage 3, horizon.** The unit records how long its warrant stands before it must be re-established. A horizon is not a time-to-live and it is not a decay rate. It is the condition under which the grade remains valid, which for a `declared` unit is typically the revision of a policy, for an `observed` unit the re-measurement interval of its instrument, and for a `derived` unit the validity of every input it was derived from. That last case is why derivation edges are retained rather than collapsed: §7.2's invalidation primitive needs them to find the derived entries whose supporting condition has lapsed.

**Stage 4, hyperedge formation.** A relation is written whole, with all positions bound, or it is not written. There is no partial relation and no placeholder participant, because a partially bound relation is exactly the chimera §5.2 exists to prevent. Where a participant is genuinely unknown, the correct representation is a different, lower-arity relation type, which forces the ambiguity into the schema where it can be seen instead of hiding it in a null.

**Stage 5, write-time confidence.** The unit carries $c_w$, the confidence in the content *given the source*, which §4.3 separates from serving-time confidence. Keeping the two apart is what lets a unit be simultaneously well-attested and currently unservable, which is the normal state of a superseded claim.

**Stage 6, consolidation.** The merge candidate is proposed and gated (§7.1). This is the only stage that can reduce the store, and it is the only stage with an irreversible failure mode, which is why it is the one carrying a gate rather than a heuristic.

### 6.2 The read path

Retrieval assembles an **evidence subgraph** rather than ranking units. Three stages.

**Stage 1, typed selection.** The query declares the knowledge type it is asking about. This is a real requirement rather than a convenience, and §10 records what it costs: when the type must be inferred from the query instead of observed, the guarantee degrades sharply, and the paper reports that as a limitation rather than working around it.

**Stage 2, coverage-guaranteed assembly.** Units and hyperedges are admitted under a per-edge-type conformal band (§7.3), so the guarantee attaches to the retrieved structure rather than to a score on the answer. The output is a subgraph with an ordering, not a list with a number.

**Stage 3, the three-valued verdict.** The subgraph resolves to **supported**, **insufficient** or **refuted**, and the three carry different remedies: supported serves, insufficient triggers collection rather than a lower-confidence answer, and refuted hands off to conflict governance (§7.2). Collapsing these to one scalar is the

failure §7.3 is built against, because a low score cannot distinguish a claim the store disbelieves from one it has nothing to say about.

### 6.3 Indexing and access paths

The store maintains three indexes, and the third is the one a binary graph does not need.

1. **Unit index** by identity and type, for direct lookup.
2. **Incidence index** keyed by `(relation-type, position)`, mapping a unit to the hyperedges it participates in *in that position*. Because typed relations have fixed arity, position is known at plan time, which recovers most of the selectivity a binary adjacency index would give (§5.5).
3. **Lifecycle index** over serving eligibility, so that the read path never walks superseded or revoked units and then filters them. Eligibility is a stored property, not a predicate evaluated at query time, which is what makes §7.2's primitives cheap to serve.

The version and conflict history are deliberately *not* indexed for serving. They are audit structures, read when a decision is questioned rather than on the query path.

### 6.4 What each stage costs

The design's practical objection is that structural checking is too expensive to run at write time. Measured against the cost of generation it is not close, and the reason is architectural rather than a matter of tuning.

Cost is bounded structurally in four ways. The consolidation gate runs at **ingest time**, which is far more latency-tolerant than query time. Its first stage rejects hopeless candidates before any generation cost is paid. The gate is **region-scoped** to the touched subgraph rather than the store. And probe references are cached across candidates. The typed checks surveyed in §1.2 run at a microsecond, at roughly 7% over generation already paid, and at one encoder forward pass respectively, against an LLM decode measured in seconds.

The read path's cost is dominated by retrieval and by the routing decision, not by the guarantee. That cost is measured as a fraction of **slot-held service under real concurrent execution**, rather than as isolated component latency: the arm is driven across an arrival-rate sweep and read at a matched achieved utilization, so a router's encode is charged for contending with the retrieval it is routing to. §9 reports what that measurement found, and it is the finding that most changed our own design.

### 6.5 Degradation, and what the store does when it is wrong

Three modes, each with a designed response rather than an undefined behaviour.

**Insufficient calibration data for a type.** The per-type band widens until it is uninformative. The store returns `insufficient` rather than a wide band presented as an answer, which is the correct behaviour and also the honest one, since a band covering the whole corpus carries no information whatever its nominal coverage.

**A merge that should not have happened.** The consolidation gate is validated only in the regime it is operated in, and §10 records the budget beyond which per-merge decisions chain transitively into one giant component. The response is operational: the gate is a per-merge criterion applied to a merge *set*, so the budget is a first-class parameter and not an implementation detail.

**A type assignment that was wrong.** This is the most damaging error the store admits, because it mis-conditions the guarantee rather than degrading it. There is no internal check that catches it, and we say so in §10. The mitigation is that types are assigned by the writer's schema rather than inferred, which converts a silent statistical error into a visible schema error. **Identity drift shares that property and has the same mitigation**, stage 0's stable key, and the same residual when the mitigation's precondition fails; §10 states it separately because the lifecycle depends on it.

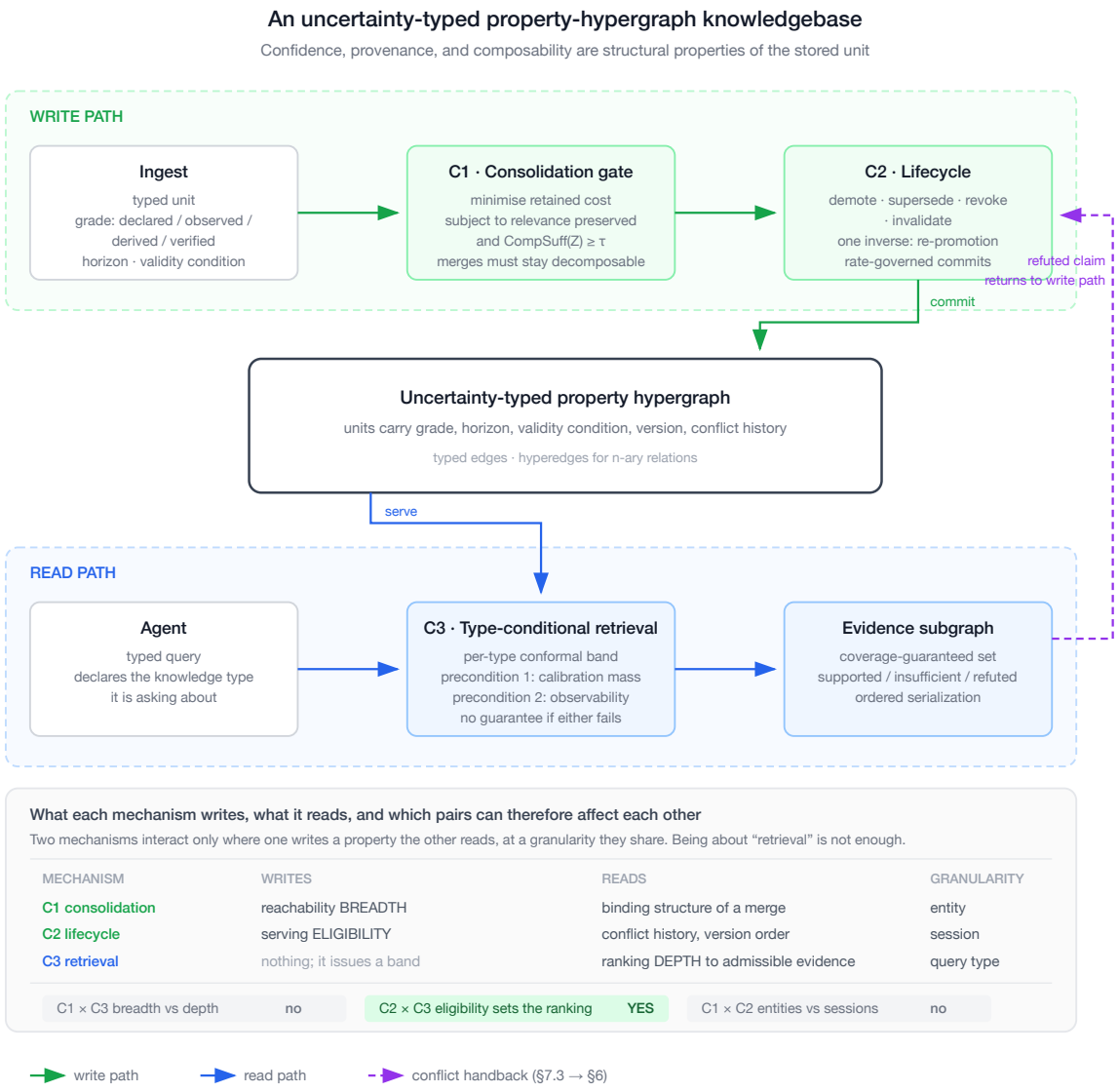


Figure 3: **the three mechanisms as specified.** The four-grade provenance lattice at ingest, C1's objective and its composability constraint, C2's four primitives and the single inverse transition three of them share, and C3's two preconditions with the rule that no guarantee is issued when either fails. The panel beneath records what each mechanism writes and what it reads, which is what decides whether any two of them can affect each other (§8).

## 7. Three mechanisms over the substrate

This section specifies the mechanisms. It states what each is designed to do and what would show it worthless; it does not carry our own measurements, which are collected in §9 so that a reader can see the specification and the evidence separately and judge how far apart they are. Figure 3 gives all three against the substrate, with the write and read sets that §8 turns into a composition criterion.

### 7.1 C1: consolidation gated on composability

**The objective.** Consolidation is a constrained optimization, not a compression ratio. Minimize the cost of the retained region $Z$ subject to two constraints: that the information $Z$ carries about the answer given the query is preserved, and that a composability predicate CompSuff($Z$) stays above a threshold $\tau$. CompSuff($Z$)**, compositional sufficiency, is the fraction of the inferences a region was meant to support that can still be assembled from $Z$ after the merge**, weighted by how much binding information each of those inferences requires. It measures whether the region still *composes*, which is a different property from whether its facts are still *present*, and §2 gives the evidence that the two come apart. Candidate inferences the region supports are enumerated and weighted by the binding information their composition requires (§5.3), and the top $B$ are defended. Defending the top $B$ *is* the declaration of what the region is for.

**Three questions, three instruments.** The gate answers three distinct questions, and the central design error we made and corrected was answering the second and third with one instrument.

*(a) Admissibility: is the structure still there?* A structural pre-filter over graph operations with no generation, rejecting candidates that break typed-path reachability for a defended path. It is near free and it prevents paying generation cost on hopeless candidates. **This stage is specified and then retired**, and the reason is architectural rather than empirical: it rejects candidates that *break* reachability, which is a compression criterion, while consolidation has the opposite failure mode, since merging bridges regions that were never connected and so *manufactures* paths rather than breaking them. The stage covers one direction of the lifecycle and not the one consolidation travels. We retire the defended-path subtraction and specify the predicate as the raw cross-product. Appendix E records this as one of our own defects rather than as a design choice made at the time.

*(b) Sufficiency: can the deployment reasoner assemble it?* Answered by counterfactual ablation of retained units against the answer, rather than by resampling the generator. It is an intervention measured against an outcome, so §4.4's exclusion of generator-internal signals does not apply to it [AttriMem].

*(c) Correctness: is the assembled answer right?* This is the question that needs an instrument carrying information the generator does not already have, and it is why (b) and (c) cannot share one. Verification must come from a referent *stochastically dependent on the truth of the claim* and not satisfiable by agreement with the claimant [Theorist Toolbox]. Both conditions follow from §1.1, since agreement is downstream and independent information is not. Our default instrument meets them at fixed 2x cost and without white-box access, by probing the **subgraph** twice in structurally asymmetric ways rather than by inspecting the answer twice.

**Decision.** CompSuff $\geq \tau$ accepts; otherwise repair and re-test, and failing that fall back to the fuller subgraph. Consolidation is never allowed to be the irreversible step: the fallback is always available because the pre-merge region is retained until the gate passes.

**What would show it worthless.** The gate is a per-merge criterion, and per-merge safety does not aggregate. §10 states the regime beyond which individually safe merges chain transitively, and §9 reports what the gate separated inside its regime, and Appendix F states the three criteria it was pre-registered against. The instruments for (b) and (c), the probe construction, and the signals we exclude are in Appendix D; parameters are in Appendix C. The claim this section defends is that composability must be *gated on a measured property*, and that survives any instrument meeting the two conditions above.

### 7.2 C2: conflict governed by four primitives

The literature discusses forgetting as one operation, which is why preservation and eviction read as contradictory goals. They are not, once the operation is decomposed by reversibility and audit obligation.

| Primitive | Effect | Reversible | Audit obligation |
|---|---|---|---|
| **Demotion** | hot/active tier → cold/archival tier | yes | cheap, rate-governed |
| **Supersession** | typed retire-on-conflict, history retained | yes | the retirement is typed and logged |
| **Revocation** | the only true delete (leakage, poisoning, legal) | **no** | the deletion is itself logged with provenance |
| **Invalidation** | transitive: a *derived* entry whose supporting condition no longer holds leaves the active view | yes | the trigger is recorded; the entry is never deleted |

**The three reversible primitives share one inverse, and revocation has none by design.** A unit superseded on conflict, or invalidated because its support lapsed, can have its warrant return: the conflicting evidence is itself retired, or the supporting condition holds again. Without a way back, such a unit can only be re-entered as a *new* one, which discards the identity and the contradiction history that made retirement auditable in the first place. We therefore specify **re-promotion** as a transition over the existing primitives rather than as a fifth primitive: demotion, supersession and invalidation are reversible, and a re-promotion returns the unit to the active view under its original identity with the re-entry logged exactly as the retirement was. The external precedent is direct, a revocable memory that returns archived precedents to active service when evidence reappears, with the return phase measured rather than asserted [TEPA].

**The inverse is what makes retirement worth having**, and this is the paper's largest measured effect rather than a design intuition; §9 reports the size and the schedule that isolates it. The architectural point is the one that survives the number: a lifecycle whose retirements cannot be undone is not a conservative design, it is a lossy one that has moved its losses somewhere harder to see.

**What the four primitives presuppose.** Every primitive above operates on a unit with a stable identity. Demotion moves that unit between tiers, supersession retires it against a conflicting successor, revocation deletes it, invalidation withdraws it when its support lapses, and re-promotion returns it under its original identity. None of them addresses **identity drift**, the case where one thing in the world is written twice under two identities and the two entries never meet. Drift is not a conflict, so supersession cannot see it; nothing has lapsed, so invalidation cannot; and there is no version order relating the two, so there is nothing to demote or revoke that would help. **Drift is therefore prevented at write time by §6.1 stage 0 rather than repaired at lifecycle time**, which is a design choice with a cost: §10 records what happens when stage 0's precondition fails.

**A naming hazard we inherited and now fix.** The external work whose result licenses re-promotion calls its own retire-on-conflict operation "revocation" [TEPA], and it is our **supersession**, not our delete. We keep our names because the distinction the table draws is between reversible and irreversible, and their operation is reversible.

**Rate governance is specified and not claimed.** A store can commit faster than it can verify, and the primitives above are governed by a rate limit on unverified commits. We specify the mechanism and do not report a measurement of it; §10 lists it among the things we did not test.

### 7.3 C3: type-conditional retrieval and a three-valued refusal

**Why one scalar cannot carry this.** The quantity a scalar confidence is usually taken to approximate is posterior uncertainty about the claim, and it does not: token-level entropy has essentially no relation to the exact posterior entropy of the state being asked about ($r \approx -0.001$), while the entropy of the *calibrated* posterior over declared states correlates at **0.742** [Semantic Map]. Token-level uncertainty is not posterior uncertainty. That result is about a calibrated posterior over declared states rather than about confidence in general, and we use it for exactly that: it establishes that the observable scalar and the quantity a consumer needs are different quantities, which is what forces the guarantee onto the structure instead.

**Per-edge-type coverage.** The guarantee is conformal and conditional on the type of the edge being retrieved rather than marginal over a query distribution, for the reason §2 gives: a marginal guarantee under-covers the rare cell, which is the cell an agent reaches for when the common path has failed. The cost is larger prediction sets and a requirement for per-type calibration data, and we take that trade deliberately rather than incidentally.

**Two preconditions, both of which can fail.** The guarantee holds when the type is *observed*, and it needs enough calibration mass in the cell being served. Both are real constraints rather than formalities, and §10 reports what each costs when it fails, including the case where the guarantee degrades below the marginal baseline it was meant to improve on. We report that as a limitation rather than presenting the mechanism as unconditional.

**Three outcomes, not one score.** Retrieval returns **supported**, **insufficient** or **refuted**, with separately certified budgets. The distinction that matters is between the second and the third: insufficiency is a statement about the store's coverage, refutation is a statement about the claim, and a single scalar confidence cannot express both because a low value is ambiguous between them [SURE-RAG; Two Axes of Abstention]. The three carry different remedies. Supported serves. Insufficient triggers collection rather than a hedged answer. Refuted hands off to conflict governance (§7.2), which is the point at which C3 and C2 meet and the reason §8 predicts those two to interact.

**Aggregating without breaking the guarantee.** Coverage conditional on edge type does not compose automatically across a subgraph containing several types. The serving rule is that a subgraph inherits the weakest per-type guarantee among the edges it contains, which is conservative and cheap, and which is why the evidence subgraph is returned with an ordering rather than a single band.

### 7.4 Multi-principal access, and why delivery is part of the contract

A store written by humans, software and agents at once has to answer a question a single-writer store never faces: what a given principal is entitled to see, and in what form. Two commitments follow from §4.

First, eligibility is a **stored property** rather than a query-time predicate (§6.3), so that a principal's view is a projection of the graph rather than a filter over it. Second, and less obviously, **delivery is part of the access contract**. Knowledge present in a store contributes nothing by itself, and a memory system that adds material and relies on the model choosing to use it inherits the model's compliance failure rate; the controlled evidence for this is that facts held only in a summarization channel were absent from 106 of 108 forced compactions while the same facts injected from an externalized store arrived at 138 of 138 [Delivery Not Storage]. A design that treats retrieval as complete once the material is in context has not finished the job.

## 8. Composition: what each mechanism reads and writes

This section states the criterion introduced in §1.2 against the model it applies to. The three mechanisms operate on the one store defined above, and the natural expectation is that they interact: all three touch what a query eventually gets back. Two of the three pairs do not, and the exceptions are not random. Which pairs can affect each other is decided by a property of this model rather than by anything the mechanisms do internally, so the criterion is stated here, against the substrate, rather than only in the section that measures it.

**Characterise each mechanism by what it writes and what it reads.** The store exposes several distinct properties, and a mechanism touches only some of them. Consolidation changes which units are reachable from which. Lifecycle governance changes which units are eligible to be served at all. Calibrated retrieval writes nothing, and reads one specific thing: how far down a returned ordering the admissible evidence sits.

| Mechanism | Writes | Reads | Addresses units at the granularity of |
|---|---|---|---|
| **C1** consolidation (§7.1) | reachability **breadth**: which units are reachable from which | binding structure of a candidate merge | the **entity** |
| **C2** lifecycle (§7.2) | **serving eligibility**: which versions the store will return | conflict history and version order | the **session** |
| **C3** calibrated retrieval (§7.3) | nothing in the store; it issues a band | the returned ordering, specifically the **depth** at which admissible evidence sits | the **query type** |

The criterion is a read-write dependency, not a topic overlap. Two mechanisms over one store can affect each other only if one writes a property the other reads, at a granularity they share. Being about "retrieval" is not enough, and this is what makes the criterion worth stating rather than assuming. Applied to the three pairs, and derivable from §7.1 through §7.3 without reference to any measurement, it gives:

- **C1 and C3 do not interact.** C1 writes breadth; C3 reads depth. Adding reachable units that rank below the gold evidence enlarges what is reachable without changing how far one must go to contain the answer. The written property and the read property are different properties of the same ordering.
- **C2 and C3 do interact.** C2 writes serving eligibility; C3 reads the ordering that eligibility determines. This is a direct write-to-read edge and it is the one pair where a lifecycle policy is legible to a coverage guarantee.
- **C1 and C2 do not interact.** Here the properties are unrelated *and* so are the granularities: C1 fuses entities, C2 demotes sessions, and session identifiers are never merged. A unit's session membership survives any amount of entity fusion.

§9 reports what happened when each pair was run over one store. All three predictions hold, the positive one with a closed form.

Stated honestly: we did not derive this table before running the experiments. The three pairs were measured first, two returned nulls that we initially read as disappointments, and the property assignments are our reconstruction of why. What makes it more than a description of three outcomes is that the assignments are readable off §7.1 through §7.3 without consulting §9, so a reader can check the criterion against mechanisms we have not tested, and that it discriminates rather than only predicting failure. The criterion is falsified by a pair that writes and reads the same property at a shared granularity and still fails to interact. Three instances in one system is not a law, and we do not present it as one.

Two consequences follow, and they are why the criterion is worth stating as design guidance rather than as an observation about our particular three mechanisms.

**The first is a test plan.** Name the property each mechanism writes and the property each reads, and the pairs worth integration-testing are the ones whose sets intersect. The rest can be skipped, and skipping them is not a gap in rigour: running a pair whose sets do not intersect spends a run confirming an absence the specification already implies.

**The second is less comfortable, because it bounds what end-to-end evaluation can tell you.** A mechanism whose writes no other mechanism reads cannot be validated end to end at all. C1 sits in exactly that position with respect to both C2 and C3, so a downstream task metric would report a null for it whether

the consolidation gate worked perfectly or did nothing whatever. That is why C1 carries its own instrument measuring reachability directly (§7.1). The general form is worth stating plainly: **wherever no read path crosses what a mechanism writes, end-to-end numbers cannot distinguish a mechanism that works from one that is inert.**

This is the same read-write vocabulary §4.3 uses to separate $c_w$ from $c_r$: the model's fields are defined by which side of the store's boundary computes them, and so are its mechanisms.

## 9. What we built, and what measuring it changed

This section is deliberately short. The paper's contribution is the design in §4 to §8; the measurements matter here insofar as they changed it, and two of them changed it a great deal. The full experimental record, the apparatus, and a catalogue of thirteen measurements of ours that were computed over the wrong population are in Appendices A, B and E.

| Experiment | Status | Outcome |
|---|---|---|
| **E4a** headroom for type-aware routing | run | **KILLED.** The oracle router is dominated on latency and throughput in all five seeds (throughput never below 7x and tightly spread; p95 13x on average but ranging 1.6-28.8x across seeds, so throughput is the stable axis) while holding a small unestablished accuracy edge; concurrent work reaches the same conclusion with a certification framework [RouteGuard] |
| **E4b** a cheap learned router | run | **KILLED on cost**, conditional on E4a |
| **E4c** the taxonomy carries information | run | **fired on one corpus**, not the other |
| **E1d** label-free predicate vs a learned gate | run | **all three criteria fail to fire**; the predicate separates from its control by 16× the detectable effect, with a bounded regime |
| **E2** governed conflict state | run | **two-sided.** Re-promotion worth +53.8 points under reversal; retirement *without* an inverse buys +1.3 and is unmotivated |
| **E3** type-conditional calibration | run (3 of 4 clauses) | coverage needs a label the query does not carry (**16× set size** without it); the **insufficiency channel is unfinished** and fails a fair comparison |
| **E5** does C1 interact with C3 | run | **no interaction.** Consolidation moves 83.7% of per-question scores and moves the coverage band by 0.1% |
| **E7** does C2 interact with C3 | run | **interaction, with an identity, tested at $k = 1$ only.** Retained stale knowledge inflates the required set by +0.500 [+0.385, +0.615] |

| Experiment | Status | Outcome |
|---|---|---|
| **E10** does C1 destroy C2's supersession | run | **damage is real and does not propagate.** 15.6% of updates collapse structurally; the lifecycle is unaffected |
| **E1a–c** re-specified admissibility stage, **E4d** token-axis routing | not run | held, and named in §10 |

Four things in that table are worth stating in prose, because each altered the design rather than scoring it.

**The inverse transition is the largest clean effect in the paper, and it is a correction rather than a mechanism.** Running four lifecycle policies through a drift schedule in which a value is asserted, superseded, retracted and reasserted, the arms are indistinguishable until the retraction. Asked which dated statement is in force, keeping everything answers correctly 3.8% of the time, overwriting 5.1%, and supersession 5.1%. Retirement alone is therefore worth **1.3 points**, which is nothing. Adding the inverse takes the same store to **59.0%**, a gain of **53.8 points**. This is why §7.2 specifies re-promotion as part of the lifecycle rather than as an enhancement to it: a retirement primitive without an inverse is not a conservative design.

**The composability gate separates from its control, and its label-free claim is narrower than we first stated.** The predicate separates from its closest published control by sixteen times the smallest effect the design could detect, and none of its three pre-registered kill criteria fired. But the label-freedom claim holds against corpus shift and not against a held-out query category, so we narrow it rather than report it as passed.

**The type-conditional guarantee holds only when the type is observed, and that is expensive.** An inferred-type router costs **16x** the evidence of an oracle-typed one at the same worst-type coverage, and always returning the widest band costs about **21x**. The refusal channel is reported as unfinished: our replacement for a scalar confidence does not yet beat a fair comparison, and §10 says so.

**The premise that type-aware routing pays did not survive, and this is the result that most changed the paper.** We built the design partly on the expectation that knowledge types prefer different retrieval backends and that routing per query would therefore beat any single backend. Under real concurrent load it does not. Across five seeds, always serving the simplest backend beats the oracle router on latency and on throughput in every one, with a throughput margin never below **7x**. The type structure is real and reproducible on dialogue; exploiting it costs more than it returns. We report this as a result because a design paper that only reported its surviving premises would be making exactly the error §1.1 describes.

The comparison is fair because of a baseline we adopted from the routing literature rather than one we chose: the router is measured not only against every single backend and a fixed policy but against a **latency-matched random mixture**, on the reasoning that a random mixture of two endpoints traces the line between them in expectation, so appearing on a Pareto frontier is not by itself a result [TRACE-ROUTER]. Holding our own router to that standard is a large part of why this premise did not survive.

**The instrument behind that last result is itself a contribution.** Routing cost is measured as a fraction of slot-held service under real concurrent execution, rather than as isolated component latency, which is the axis the surrounding literature defers. Under eight-way concurrency the static backends show essentially no contention penalty while the routers run four to seven times longer, because a router's encode contends with the very retrieval it is routing to. Appendix B gives the apparatus and the operating point.

## 10. What we do not claim

**The type must be observed, and inferring it is not a workaround.** §7.3's guarantee is conditional on a label the query does not carry. Inferring it puts worst-type coverage below the marginal baseline the mechanism exists to beat, so the honest statement is that the mechanism works where the type is known out of band and degrades below its own baseline where it is not.

**The refusal channel is unfinished.** Three outcomes are better specified than one scalar, but our instrument for separating insufficiency from refutation does not yet beat a fair comparison. We report the channel as designed and not as demonstrated.

**Identity drift is prevented, not detected, and the prevention has a precondition.** §6.1 stage 0 takes identity from the source's stable key, and where a source supplies one that closes the problem at write time. Where it does not, the fallback identity is the extraction signature, and that is stable under re-extraction of the same span but *not* across a re-phrasing, a re-segmentation, or a change of extractor. In those cases the same fact can enter twice under two identities, and we have **no mechanism that detects it and none that repairs it**: the four lifecycle primitives all presuppose the identity they would need to reconcile (§7.2), and consolidation merges on decomposability rather than on sameness, so the gate is not an identity resolver and should not be mistaken for one. Entity resolution is a mature field and the right repair is presumably to adopt one of its methods at stage 0, which we have not done. We flag it as the gap in the lifecycle rather than presenting the four primitives as complete.

**The composability gate does not itself compose.** It is defined per merge and applied to a merge *set*, and per-merge safety does not aggregate: beyond roughly a thousand merges, individually safe choices chain transitively into one giant component, at which point the gate performs worse than the surface-similarity control it otherwise beats by an order of magnitude. A set-level criterion is the obvious repair and we have not built it.

**Rate governance is specified and untested.** §7.2 governs the rate at which the store commits faster than it can verify. We did not measure it.

**Composition is a reconstruction, not a prediction.** The criterion in §8 was reconstructed after the three mechanism pairs were measured, not registered before. What earns it more than heuristic status is that the property assignments are readable off §7 by someone who has not seen §9, and that it discriminates rather than predicting nulls uniformly. The criterion is falsified by a pair that writes and reads the same property at a shared granularity and still fails to interact.

**The evaluation is one machine and a small corpus.** Everything in §9 comes from one machine. Cross-machine variation is untested. The two corpora we built are small, the accuracy intervals are underpowered, and where a null is reported it bounds the effect rather than establishing its absence.

**The artifact is withheld.** Appendix B states what that costs a reader, which is that nothing in §9 is independently reproducible from this paper alone.

**And our own numbers were wrong thirteen times.** Turning the paper's standard on its own instruments found thirteen measurements computed over the wrong population, nine of which had inverted a conclusion we had already written down. Appendix E lists them with the number each changed. We include this not as a confession but because a reader deciding whether to trust the design should know the error rate of the process that produced its evidence, and because the procedure that now runs over every number before it is published is itself part of the contribution.

## 11. Conclusion

Confidence and provenance are computed at answer time in most agent-facing knowledgebases, and §1.1 argues that this places a ceiling on them that no amount of engineering removes: information about whether a claim is true enters through the primary record, and every later stage is a rendering of that record. A score computed from the rendering cannot carry information the record did not.

The response this paper specifies is to move those properties into the store. A unit carries its type, its provenance graded on a four-valued lattice with a validity horizon, the condition under which it held, its version and conflict history, and a confidence separated into write-time and serving-time quantities. Relations are typed hyperedges rather than binary edges, because the relations that carry agent memory are irreducibly $n$-ary and every binary encoding we examined recovers addressability without enforcing arity at write time, which is the property the three mechanisms consume. On that substrate, consolidation is gated on whether a merged region still decomposes into the inferences it supported, conflict is preserved through four lifecycle

primitives of which three share a single inverse, and retrieval returns a coverage-guaranteed evidence subgraph with a three-valued verdict rather than a scalar.

**The most portable result is the smallest one.** Two mechanisms over one store can affect each other only where one writes a property the other reads, at a granularity they share. It is checkable from specifications before anything is built, it discriminates rather than predicting failure uniformly, it carries a stated falsifier, and it bounds what end-to-end evaluation can establish about a mechanism whose writes nothing reads. We expect that to transfer to stores built on entirely different mechanisms, and it is the part of this work we would most like to see used elsewhere.

**Where we would go next**, in the order we would do it. The consolidation gate's admissibility stage is specified for the wrong direction of the lifecycle and is currently retired; re-specifying it for the path-manufacturing direction that consolidation actually travels is the first item, and the experiments held on it are registered in Appendix F. The gate is a per-merge criterion applied to a merge set, and per-merge safety does not aggregate, so a set-level criterion is the repair that would extend it beyond the regime §10 bounds it to. The validity horizon is currently a stored field that nothing acts on automatically; a demotion rule keyed off horizon expiry would close that loop without waiting for a query to notice. The lifecycle's four primitives all presuppose the identity they operate on, so identity drift is prevented at write time and neither detected nor repaired afterwards; adopting an entity-resolution method at stage 0 is the repair, and it is the gap we would close first if the store were carrying sources without stable keys. The refusal channel separates supported from insufficient from refuted by design but not yet by measurement, and it is the mechanism we would most like to be wrong about, because a store that can say *I have nothing on this* is worth more than one that returns a low number. Beyond those, the type-observability requirement is the sharpest limit on the retrieval guarantee and the one most likely to decide whether this design is practical, and the latency instrument needs a second machine before its numbers should be trusted as anything but internally matched.

## Appendix A: Review provenance

### A.1 Why this appendix exists

§1.1 argues that a claim's evidential value depends on how often the procedure would have produced it anyway. That applies to a design paper's literature review as much as to its experiments: a review that reads until it finds support has a denominator too, and it is usually unreported. This appendix reports ours.

### A.2 Method

Papers reached the review through a graph over the corpus rather than through keyword search. Concepts were located by querying a persistent knowledge graph built over the full paper set, which repeatedly surfaced relevant work that reading-driven passes had missed, including two papers that changed load-bearing design decisions and that no author on this project had read. Framework names read off graph node labels were treated as leads only.

Two rules were fixed in advance and held:

1. **A verdict requires a full-text read.** No claim in this paper is supported by an abstract, a summary, or a node label.
2. **Every reviewed paper receives a verdict class** (SUPPORT, QUALIFY, NEUTRAL, or NOVELTY-THREAT) recorded whether or not it helps us, and cross-domain citations are labeled as analogies rather than proofs.

Reviews were conducted in dated passes as the corpus grew, with each pass reading the new arrivals against the current design rather than against the original thesis.

### A.3 Aggregate

Across nine dated passes, **63 papers were read in full and assigned verdicts**:

| Verdict | Count | Meaning |
|---|---|---|
| SUPPORT | 36 | strengthens a claim or supplies a mechanism |
| QUALIFY | 17 | forces a scope restriction, a re-framing, or an added control |
| NOVELTY-THREAT | 4 | publishes something we had treated as ours |
| NEUTRAL / precedent | 6 | supplies a component or an architectural precedent |
| **REFUTE** | **0** | no paper contradicted a claim of ours outright |

Papers carrying a dual verdict are counted once, under the more consequential class. **The count does not cover every citation. Sixty-three papers were read and carry a verdict row. Fifty-seven are cited in this edition**, two of them statistics references from outside the corpus. Reading more than one cites is normal and not a defect, but the direction of the gap has to be stated rather than left for a reader to infer from a bibliography, because this appendix exists to make the review's own denominator visible. The earlier, longer edition of this paper cited sixty-nine. Read scope is recorded per paper rather than claimed uniformly, and where a read was partial the entry says so and flags any claim that depends on unread material. The most recent pass exists to close such gaps: it revisited the one paper previously cited for its mechanism with none of its effect sizes, read the section that had been left, and upgraded the verdict accordingly.

The zero is the number most in need of context, and three things should be said about it. It is a genuine result: seventeen papers forced qualifications, two forced a positioning change, one corrected an overreach of ours about generator-internal confidence, one narrowed a gap claim we had overstated (§3), and one required us to report power floors on our own null results (§9), so the process was not merely confirmatory. It is also a weak signal on its own: the reviews were conducted by the authors, papers arrive through a relevance filter that selects for topical proximity, and results sharing benchmarks or models are correlated, so verdict counts saturate rather than accumulate [Phantom Evidence]. And a later pass sharpened that last caveat rather than repeating it: the dominant confounder across correlated results is shared *item difficulty*, not shared lineage, with a measured lineage excess of $-0.009$ $[-0.064, +0.047]$ [One Human, N Agents]. We report the count and decline to lean on it.

### A.4 What the review changed

The design-relevant outcome is not the tally but the specific changes. The load-bearing ones:

| Change | Forced by |
|---|---|
| The correctness leg of the gate was rebuilt: self-consistency is structurally unable to answer it | [Prior Laundering]; [Reliability Scales Inversely]; [Reasoning Denoiser]; [Theorist Toolbox] |
| Abstention went from one signal to three, with the refuted branch handed to conflict governance | [SURE-RAG]; [Two Axes of Abstention] |
| Confidence split into write-time and read-time fields | [CALIBER] |
| Delivery became part of the access contract | [Delivery Not Storage] |
| A latency-matched random mixture became a required baseline | [TRACE-ROUTER] |
| "Never use generator-internal confidence" was scoped rather than asserted | [Computational Basis of Confidence] |
| The latency-novelty claim was narrowed three times, then stated once precisely | [AgentKVShift]; [ACM]; [SwiftMem] |
| Positioning moved from conceptual priority to measurement | [Supra Cognitive Modes]; [ACM] |
| The C1 gap claim was narrowed: label-freedom is not the differentiator, post-merge decomposability is | [Ontology-Guided Dedup]; [AdaMM]; [SESA] |
| An arm was added to E1 to test that distinction instead of asserting it, with a fidelity control as the crux | [ThinkReset] |

| Change | Forced by |
|---|---|
| Abstention was re-grounded as a soundness requirement rather than a coverage trade | [CAGE] |
| Every pre-registered null now reports a power floor, which moved the E4 refutation onto different evidence | [Temporal Leakage] |
| "Report composition at the operating point" became one stated principle rather than four separate remarks | [One Human, N Agents]; [DFOT]; [Temporal Leakage]; [SPIKE-Bench] |
| C1 acquired a formal target, and lost the right to call query-agnosticism a strength: sufficiency is task-relative, so the defended path set had to be recognised as a *declared* relevance variable | [Source-Side Sufficiency] |
| "Keeping history" was dropped as a C2 claim once two systems published it, and C2 re-scoped onto revocation, rate governance, multi-principal contracts, and confidence-linked conflict | [HiGram]; [Eigenius] |
| Per-group conditional coverage was dropped as a C3 claim, and a hard per-type calibration floor was adopted in its place | [Conformal Fairness Audit] |
| The provenance marker went from binary to a four-grade lattice, with a slot for *declared* content that had nowhere to live | [Eigenius] |
| Calibration was separated from discrimination, so the conformal layer no longer implicitly claims both | [Certified Deferral] |
| Coverage-grid granularity became a stated criterion for the confidence layer, not only calibration error | [Sparsity Pitfalls]; [Certified Deferral]; [LoBoost] |
| The per-type coverage requirement stopped being an assertion and became two reported diagnostics, with a policy for rare types | [LoBoost] |
| A generation-budget control was added, and applied against our own abstention probe rather than only cited | [Shift Monitor] |
| The evidence subgraph acquired a canonical serialization order, because information equivalence does not imply measurement equivalence | [Semantic Map] |

## Appendix B: Apparatus, corpora, and artifact statement

### B.1 The literature graph behind the design

Concepts and grounding derive from a knowledge graph over a continuously growing arXiv corpus, refreshed on a weekday schedule. At the time of writing the graph holds approximately **17,800 nodes and 20,700 edges over 739 papers in 383 communities**; the counts change daily and no claim in the paper depends on their exact values.

### B.2 Models, hardware, and corpora

Reported because §10 commits to reporting them, and because the latency numbers in §9 are meaningless without a machine.

| Component | What was used |
|---|---|
| Dense retrieval (B2, B3) | `sentence-transformers/all-MiniLM-L6-v2` |

| Component | What was used |
|---|---|
| Lexical retrieval (B1) | Okapi BM25, $k_1 = 1.5$, $b = 0.75$, no learned component |
| Graph construction (B4) | triples extracted from paragraph prose by `gpt-oss` served locally through ollama; **never from any benchmark's gold supporting-fact labels**, which would make B4 an oracle retriever rather than a backend |
| Generation and abstention judging | the same local `gpt-oss`, at a generation budget calibrated per dataset (§9) |
| Hardware, all latency and throughput numbers | MacBook Pro (Mac18,4), Apple M1 Max, 10 cores, 64 GB unified memory, macOS 14.4 |

**A consequence of that hardware worth stating rather than hiding: every latency figure in §9 is a single-machine, unified-memory measurement.** The contention it reports is real, and the harness sustains 19,148 requests per second against a no-op executor, so the plateaus are backend properties rather than harness ceilings. But the absolute milliseconds are not portable, and the ratios are the transferable quantity.

| Corpus | Split and size as used | Role |
|---|---|---|
| MuSiQue-Ans | dev; 2,364 paragraphs, 150 questions with gold | multi-hop retrieval, and the corpus where reachability harm is sparse |
| LoCoMo | 1,571 dialogue units, 1,977 questions all with gold | carries C1 and C3; its retrieval channel is ranking-bound and its reachability channel is not |
| LongMemEval | pooled; 940 sessions, 500 questions, 78 labelled knowledge-update items | the only substrate that can host C2 |
| 2WikiMultihopQA | dev; evaluated and **rejected** | rejected during selection after a task family scored 60.7 F1 from paragraph titles alone |

**Three LoCoMo question counts appear in this paper and they are nested, not inconsistent. 1,977** is the full set, every question carrying gold. **1,531** is the four answerable families (single-hop 841, multi-hop 281, temporal 320, inference 89) and is what E4a and E5 use; the remaining **446 unanswerable** questions are not discarded but are the set A6 tests abstention on. **989** is E1d's held-out evaluation half of the full set, the other half having trained the learned arm. Question sets are hash-verified between runs, and the preserved smaller question file from an earlier LLM-bound sample is retained alongside the full set so that a change in $n$ cannot be mistaken for a change in result.

**B.2a Model, versions, and how the latency numbers were taken**

| | |
|---|---|
| Generation, extraction, judging | `gpt-oss`, **20.9B parameters, MXFP4 quantization**, served by **ollama 0.32.14** on the machine below |
| Training cutoff | **not published for this build.** We commit in §10 to reporting it and cannot: the local model card does not carry one. What this costs is stated there |

| | |
|---|---|
| Latency percentiles | over requests within a run and then **across runs**: 500 requests per configuration, 8 service slots, arrival-rate sweep, **five seeds (0-4)**, read at the point nearest achieved utilization 0.80 with throughput taken at that same point. The table reports mean ± sd across seeds |
| Repetitions | **five runs per configuration, seeds 0-4.** Percentiles are computed within each run and then aggregated across runs, so the ± is a between-run spread and not a within-run one |

**What this design supports, and what it still does not.** The comparison is between configurations driven to the same achieved utilization, each seed in its own session on the one machine below, which is what the domination claim needs. It now carries a between-run variance estimate; §10 reports that estimate together with the two things it still cannot support, namely variation across *machines*, which is untested, and an exactly matched utilization, which the arrival-rate sweep can approach but not hit.

### B.3 Artifact statement: the ledger and harness are not released

Every number in this paper traces to an internal results ledger, and the discipline that ledger enforces is doing real work: it is what caught the defects in Appendix E, and where it disagrees with a findings file the ledger wins while the code wins over both. **We are not releasing it, nor the latency harness, nor the extraction and merge scripts.** The reason is mundane rather than principled: the harness and the ledger are entangled with a personal research pipeline that also ingests the author's mail, and the corpora are derived artifacts of third-party benchmarks whose redistribution terms we have not cleared. Neither obstacle is interesting and neither is insurmountable, which means this is a cost we chose to impose rather than one forced on us.

What that costs the reader, stated plainly rather than left implicit. Nothing in §9 is independently reproducible from this paper alone. A reader can check internal consistency, follow every argument, and see which claims we withdrew and why, but cannot re-run a single measurement or confirm that a reported interval came from the procedure we describe. For the negative results that matters less, since a refutation that is wrong in our favour would be a strange error to make; for the positive ones, in particular E7's identity and E1d's separation, it matters more, and those two should be read as claims supported by a described procedure rather than as verified results.

### B.4 Two honest notes about the design instrument

First, the graph covers the paper corpus only, not this project's own reasoning artifacts, so orienting on a design question requires reading the design documents as well as querying the graph, neither alone is sufficient. Second, a single-analyst review of a graph this size has blind spots by construction; a second lens would likely surface a temporal/forgetting axis we treat only as a volatility tier. **This is the same caveat that Appendix A.3 attaches to its zero REFUTE count, and the two should be read together:** a review with one lens and no refutations is exactly the pattern a missed refutation would produce, which is why A.3 reports what the qualifications actually changed rather than resting on the count.

## Appendix C: Gate parameters

Collected for reproducibility. The gate's criterion is §7.1 and its mechanism is Appendix D.

| Parameter | Meaning | Notes on setting |
|---|---|---|
| $\tau$ | Sufficiency threshold on CompSuff | Sets the accept/repair boundary; swept in E1, and its choice interacts with $B$ |
| $M$ | Probes evaluated per candidate subgraph | Sampling budget; bounds Stage-B cost directly |
| $k$ | Maximum relation-path length enumerated | Hop cap; enumeration grows sharply in $k$, so this is the main cost control |
| $B$ | Defended-path budget (top-$B$ by binding information) | Defending every path is unaffordable and wrong; trivial chains carry no binding |
| reasoner-version | Identity and version of the gating reasoner | Recorded in provenance; a change invalidates prior decisions and triggers re-gating (§7.1) |
| $N_{\text{abl}}$ | Ablated variants per attribution estimate | Roughly one batched forward pass at the value used [AttriMem] |

Cost is bounded structurally rather than by tuning alone: the gate runs at ingest time, which is more latency-tolerant than query time; Stage A avoids paying generation cost on hopeless candidates; the gate is region-scoped to the touched subgraph; and probe references are cached.

> **Not included, and a lettering note.** The internal decision log: superseded mechanisms, dated open questions, and the running record of what each review pass changed: is maintained separately and is not part of the paper. Appendix lettering matches the design document: A, B, C here, and **Appendix D is the gate mechanism**, split out of §7.1 on 2026-08-19 with its parameters left in Appendix C above.

## Appendix D: Composability gate mechanism

The instruments behind §7.1's three questions. Separated from the body because they implement the criterion rather than argue for it. Parameters are in Appendix C; what would show the whole gate is worthless is §7.1.

### D.1 Stage B, sufficiency by counterfactual ablation

Following context-attribution, we estimate each retained unit's contribution as $\phi(z, y; c) \approx F(y \mid c) - F(y \mid c \backslash z)$ over ablated variants, at a cost of roughly one batched forward pass [AttriMem]. This is an intervention measured against an outcome, not a functional of the generator's output distribution, so the exclusion of §4.4 does not apply to it.

### D.2 Stage C, correctness needs information the generator does not have

The peer-prediction constraint sharpens what the referent has to be: a scoring statistic independent of the agent's own signal elicits the prior mean, and a constant report weakly dominates honesty, so the referent must be *stochastically dependent on the truth of the claim* while not being satisfiable by agreement with the claimant [Theorist Toolbox]. Both conditions follow from §1.1: agreement is downstream, independent information is not.

Our default instrument meets them at fixed 2× cost and without white-box access. Rather than a second, stronger model, we use the **same** model under two structurally asymmetric probes engineered to fail differently: a *completion-pressured* probe ("answer this multi-hop probe from $Z$"), which under slot-filling pressure will confabulate for absent content, and a *grounding-first* probe ("list what $Z$ supports, without being told the question"), which reports only what is grounded and so misses non-salient content. Their disagreement

is independently informative, and critically, **neither failure mode is observable by resampling either probe** [ExtractConf].

**The load-bearing condition is easy to lose and worth stating plainly: the grounding-first probe reads the subgraph, not the first probe's answer.** Two probes both inspecting the generated answer would be pure downstream processing and would add exactly zero evidence, however they were prompted. The design works because the second probe re-touches the source. An instrument that loses this property is not a cheaper version of this gate; it is not a gate at all.

### D.3 What we exclude, and the four independent reasons

Self-consistency is retained only for Stage B, where the question genuinely is about the deployment reasoner's own behaviour. For Stage C it is excluded on four independent grounds: it is neutral by construction with respect to the belief that generated the samples [Prior Laundering]; sample-disagreement detectors are functionals of the output distribution and blind to the dominant error term [Reliability Scales Inversely]; it fails empirically to separate informative from noisy reasoning [Reasoning Denoiser]; and it is a bad purchase even where it helps slightly, at five times the cost for a negligible gain over log-probability [ExtractConf]. An LLM-judge substitute is not a repair: judge panels accept deliberately fabricated work at high rates [Theorist Toolbox], and by §1.1 a judge touching only the exhibit cannot add evidence in principle.

### D.4 Decision, repair, and fallback

CompSuff $\geq \tau$ accepts. Otherwise **repair**, meaning add back the minimal edge set covering the failed probes' paths, and re-test; failing that, fall back to the fuller subgraph. The gate runs at ingest time and is region-scoped to the touched subgraph, which is what keeps a two-probe instrument affordable; Appendix C gives the cost argument in full.

**Stage A is specified but retired.** The defended-path subtraction described in §7.1(a) is not part of the mechanism as we now specify it, because it is inert on one corpus and harmful on the other (§9), and every reported E1d number was computed without it. The stage is recorded here rather than deleted because Appendix E row 11 is about the fact that we specified it and never ran it, and that entry needs its referent to remain readable.

## Appendix E: Thirteen denominators we got wrong ourselves

*This appendix is the experimental-rigour record of the work behind §9. It is placed here rather than in the body because this is a design paper, but it is not an afterthought: it is the reason to believe §9 at all.*

**Figure 4 is the taxonomy at a glance and the list below is the detail**: the figure names each defect and what class of population it got wrong, the list says what it changed and what replaced it. Read the figure to see that they are one error; read the list to check any single one.

This paper's contribution is that every claim should arrive with its denominator measured. The honest test of that commitment is what happens when the discipline is turned on our own instruments, so we report it: thirteen of our own measurements were wrong, and each was found by the standard we set for others. Ten changed a number we had already written down; two changed what a number was being compared *against*; and one changed neither, because the stage it concerns had never been executed at all. Defects 10, 11 and 13 are the three of a different kind, and they are discussed together after the list rather than in the order they appear in it.

**1.** The routing kill criterion compared a **point estimate** to a threshold

- *What it did:* Returned KILL at $n = 60$ and GREENLIGHT at $n = 1{,}531$ on identical data, while the underlying quantity never moved
- *Now:* Tests the lower bound of a cluster-bootstrapped interval, and reports pick stability so that mutual information over a tied table cannot masquerade as structure

**2.** Abstention was scored as a **rate**


Figure 4: **one error, thirteen costumes.** Every entry is a statistic computed over the wrong population; what differs is which population. Nine inverted a conclusion rather than blurring one, all thirteen were silent, and nine were found after this section was first written.

- *What it did:* Ranked the graph backend best on the unanswerable set while it refused 88% of *everything*; informedness ranked it near worst
- *Now:* Scored against a matched answerable control set, and reported as informedness

**3.** Routing overhead was measured against **end-to-end latency**

- *What it did:* The denominator included queue wait, so the same configuration read 68.9% at 36% utilization and 10.9% at 96%: the criterion could be passed by adding load
- *Now:* Measured against slot-held service, with a regression test that holds work fixed and varies only queueing

**4.** Utilization was summed from **stage timings**

- *What it did:* Planning and dispatch hold the slot without appearing in any stage, so the fastest backends appeared to plateau at 0.72 no matter how hard they were driven
- *Now:* Measured as slot occupancy, wall-clock from acquire to release; the same backend then reached 0.90

**5.** A consolidation experiment was scored through **top-$k$ retrieval**

- *What it did:* Merging is a *construction* intervention; the corpus was already known to be *ranking*-limited, so every arm returned 0.0809 (including zero-merges and merge-all), and the kill criterion fired on a dead channel
- *Now:* Scored on manufactured non-gold reachability, which has no top-$k$ cutoff; verified non-circular against the predicate that selects the merges

**6.** A criterion was labelled **"under shift"** when neither compared arm can see the query distribution

- *What it did:* Both arms are label-free, so their contrast is a property of the evaluation set alone; the in-distribution and cross-corpus conditions returned identical curves
- *Now:* The shift claim now rests only on the comparison that has shift sensitivity

**7.** An aggregate averaged a **working regime together with a destroyed one**

- *What it did:* Area between curves was taken across the whole budget sweep including a saturated tail where every arm has collapsed the graph; it reported "not separated" with a *negative* estimate
- *Now:* Restricted to the regime the gate is operated in, where the same contrast is **+84.1**, sixteen times its detection threshold. **Worse than first recorded:** the same restriction also *reverses the sign* of a second contrast on the held-out-category condition, from +52.5 to **-6.3**, so the defective aggregate was not merely widening an interval but reporting the wrong arm ahead

**8.** Router quality was scored as **classification accuracy**

- *What it did:* Accuracy is dominated by the easy majority class while the mechanism's job is minority coverage, and it is blind by construction to an asymmetric loss: routing a hard question to an easy band voids the guarantee, the reverse only wastes set size. Adding class priors (the *correct* discriminant) raised accuracy 10.5 pp and cut multi-hop coverage **19.4 pp**, and across three classifiers a 27 pp accuracy gain never restored coverage
- *Now:* Scored as worst-type conditional coverage at a stated operating point, with the accuracy-optimal arm reported beside it so the divergence is visible rather than assumed away

**9.** A signal's worth was read off **AUC**

- *What it did:* AUC averages over every operating point; an abstention gate needs one. The ablation channel reached AUC 0.308 [0.267, 0.349] (genuinely informative, saliency control passing at four times random), while delivering **0.0% true-positive rate** at the 8.7% false-positive rate the gate runs at, because the distributions overlap almost completely in that tail
- *Now:* Reported as TPR at a *stated* operating point with the FPR swept and shown whole, and resolution checked before the signal is credited at all

**10.** An experiment's **comparison structure** contradicted the design it was testing

- *What it did:* §7.3 commits to *factorized* abstention, on the stated grounds that one threshold conflates a wrong answer with an inadmissible question; we then evaluated the second factor with a **substitution** bar that assumed a single axis and required the challenger to win at the incumbent's own operating point. Two experiments were run against the wrong question before anyone asked the right one
- *Now:* Re-run as a **conditional** test (does the channel carry information *inside* the incumbent's decision cells), which the signal fails (AUC 0.448 and 0.411, both intervals spanning 0.5). The verdict is unchanged and now rests on a fair comparison

**11.** A ledger entry recorded a measurement **without the corpus it was measured on**

- *What it did:* The selection ledger recorded two candidate predicates as scoring identically. That is true of MuSiQue, where they agree to within 0.14%, and false of LoCoMo, where they agree on 45.6% of candidates and LoCoMo carries the headline. Because the entry read as a settled equivalence, nobody checked which predicate the code actually ran, and it ran neither the one §7.1 specified nor by choice: `score_decomposability` fell through to the raw cross-product it was written to improve on. **Every E1d number is a cross-product number**
- *Now:* The subtraction is retired and C1 is specified as the cross-product (§7.1), which is what every reported number already used. The ledger entry now carries the corpus it was measured on. **A ledger is only a control if its entries carry the scope they were measured at**

**12.** An arm was compared against a control **that had been given a different budget**

- *What it did:* Consolidation arms were compared against merge-all, which performs five times as many merges. The statistic then answers *how many* merges were made rather than *which ones were chosen*, which is the only question a gate can be credited for. It reached a headline: the gate was reported as bounding manufactured reachability by **8×** (+6.9% against +56.3%). Budget-matched, the arms are +6.9% gate, +4.6% random and +5.6% anti-gate, so the ordering is real but the effect is a fifth of what was claimed and the gate's advantage over merge-all is **entirely budget**
- *Now:* **The 8× headline is WITHDRAWN.** Every C1 arm now carries a matched-budget control, which is what caught the same confound in E10 before it reached a report: random selection at 300 merges collapses updates at exactly the gate's rate (3/77 against 3/77), so the gate does not bound the collapse either. **A control that is not budget-matched is not a control**, and two of this paper's claims died to that sentence

**13.** A statistic whose **only job was to select a baseline** was read as a comparison against it

- *What it did:* A4's all-gold@10 micro-average exists to decide which single backend is `best_single`, and it puts B1-verbatim at 68.1% against B2-chunked's 63.2%. §9's domination paragraph read those two numbers as always-B1 against the *oracle* and concluded the oracle was dominated on accuracy as well as on latency and throughput. Both numbers were correct; the population one of them named was wrong. The claim reached the abstract, §2, §9's results table and §10, and it survived two rounds of review because nothing in it looks false.
- *Now:* The refutation rests on latency and throughput alone, where the oracle loses on each of them in every seed and by never less than 7x on throughput, and its accuracy edge is positive and unestablished (+0.036 and +0.013, neither clearing its MDE). **This is the one entry none of our own checks would have caught.** The consistency procedure verifies counts against tables, figure numbers against prose, and cited experiments against defining sections, and every one of those tests passes on a sentence that names the wrong population for a correct number.

Three things connect them, and the pattern is more useful than any individual fix.

**All thirteen reduce to one error, including the three that presented as something else: a statistic computed over the wrong population.** A point estimate over a distribution that has a spread; a rate over a set that excludes the controls; a ratio over a denominator that includes unrelated work; a sum over the wrong interval; a construction change scored through a population the corpus had already bottlenecked; a contrast labelled by a condition its population cannot vary with; an average taken over a population in which the effect cannot exist; a router scored on a population dominated by the class it was not built to serve; a signal credited on an average over operating points rather than the one it runs at;

a challenger scored against a population of comparisons the design had already ruled out; an equivalence recorded over one corpus and relied on for another; an arm scored against a control drawn from a population five times the size; and a baseline-selection statistic read as a comparison against the baseline it selected. This is the operating-point principle above, which we found in four external literatures before noticing we were violating it in thirteen places of our own. The three-way split at the head of this section describes what *changed* when each was fixed; this is what they have in common underneath.

All thirteen were silent. None produced an error, a crash, or an implausible number. Ten of them each produced a *plausible* number that was wrong; **defect 10** produced a defensible number against an indefensible alternative, which is quieter still, because nothing about the number itself is off; **defect 11** produced numbers that were correct for the predicate that ran and wrong for the one we had described, which is quieter again, because the numbers are not the thing that is wrong; and **defect 13** produced two correct numbers and named the wrong population for one of them, which is quieter still. That is precisely the failure mode this paper argues confidence estimates suffer from and precisely why it argues they must be structural. An instrument that fails loudly needs no denominator; the ones that need it are the ones that fail quietly.

Nine of the thirteen inverted a conclusion rather than blurring one, and they are worth naming so the count can be checked. Defect 1 flipped a verdict with sample size. Defect 2 flipped a ranking. Defect 3 flipped a pass into a fail. Defect 5 produced a printed refutation of one of this paper's own contributions from a channel in which no arm could differ from any other. Defect 7 turned "not separated" into a contrast sixteen times its own detection threshold. And defects 8 and 9 each inverted an *arm ranking*: the accuracy-optimal router was the coverage-worst one, and the signal with the healthiest AUC was the one that does nothing where it is used. Defect 12 turned a budget difference into an 8× mechanism effect that does not exist, and defect 13 turned a baseline-selection number into a domination claim on an axis where the oracle was in fact ahead. In every case the statistic we would have reported ordered the arms backwards, or credited a mechanism for an effect it did not produce, rather than merely blurring a real one. Measurement defects are not conservative by default, and treating them as noise that averages out is the assumption that lets them survive.

**Defects 5 through 13 were found after this section was first written**, which is the most useful thing about them, and defect 7 alone moved a headline contrast from "not separated" to sixteen times its own detection threshold.

**Defects 10, 11 and 13 are the ones we would most like to have caught earlier, and they are a different kind.** Ten of the thirteen are statistics computed over the wrong population, defect 12 among them. Defect 10 is an *experiment* compared against the wrong alternative: the design in §7.3 had already committed to factorized abstention, and the evaluation asked a substitution question anyway. Defect 10 is the only entry that did not change a conclusion (the channel fails either way), and that is precisely why it is worth recording. A defect that leaves the answer intact still leaves the warrant wrong, and a reader has no way to tell the two apart from the outside.

Defect 11 is worse than defect 10 in one specific way, and it is the reason this row exists rather than a footnote in §5.5. Eleven of the others were caught by re-examining a number; defect 13 is the second exception, since both of its numbers were correct too. This one could not have been, because every number it touches is correct: the experiment ran a well-defined predicate and measured it accurately. What was wrong was the sentence in §7.1 claiming which predicate that was. No amount of checking our arithmetic would have surfaced it, and what did surface it was running the *other* predicate head to head and finding it loses to random selection (§9). **A specification is not validated by an experiment that silently ran something else**, and the only general defence we can offer is the one this row demonstrates: execute the specified path and report what it does, rather than inferring from a ledger that it must be equivalent.

**A fourteenth of a different kind, deliberately not numbered with the thirteen.** An earlier edition of this work reported a p95 latency column that we later could not reproduce from anything we had retained: the figures appeared in no result file, no log and at no point on any archived load curve, while the throughput column beside them matched an archived run exactly, so that table's two columns had come from different places and only one had been kept. Those numbers are superseded and a design paper does not need the archaeology, but §10 argues that a reader deciding whether to trust the design should know the error rate of

the process that produced its evidence, and a number with no retained provenance is the sharpest available datum about that rate. It was silent, it reached the abstract, and it sat in the measurement the central verdict rested on, so it meets every test this section applies. It is separated from the thirteen because it is a *provenance* failure rather than a statistic computed over the wrong population: no consistency check reaches it, and the defence is retaining the artifact rather than checking the arithmetic. The count is thirteen of one kind, plus one of another.

We do not claim these are the last thirteen, and we have direct evidence that they are not: **nine of them arrived after this list was first written.** Three came from one experiment, two more from the experiments that replaced it, one from auditing how another had been *compared*, one from asking which code path a specification actually reaches, and one from adding a matched-budget control to a comparison that had never had one. We claim only that they were found by a procedure we have written down, and that a paper reporting none would be making the less credible claim.

What now runs over every number before it is reported, so that a reader can judge whether a thirteenth would be caught: a consistency check over the manuscript that verifies every stated count against the table it counts, every figure's numbers against the prose, and every experiment cited against a section that defines it; a requirement that any arm compared against a differently-sized control carries a matched-budget control before the comparison is reported at all; and the operating-point rule, that a signal is credited at the point a gate runs rather than on an average over points it never reaches. The first of those exists because of defects 2, 8 and 12; the second because of defect 12 alone; the third because of defect 9.

**A fourth blind spot is open, and we found it the same way we found defect 13, by someone reading the appendix rather than by a check.** None of those remedies compares a *protocol* row against the protocol actually run. When the latency sweep went from one seed to five, Appendix B.2a's replacement row landed directly above a surviving row that still read "one run per configuration," and a paragraph still asserting the single-run design was a limit we would rather name. Both were true of the previous protocol and false of the row beside them, and both sat in the appendix a reader opens precisely to verify the seed count. A stated count checked against its own table cannot see this; a superseded row is internally consistent with everything except reality. We have fixed the row and have no check that would have caught it.

Per §1.3 and §1.1, results carry the **selection ledger**: the number of configurations tried before the reported one, so that a reported likelihood ratio can be read as severity rather than as diagnosticity alone [Phantom Evidence]. Pre-registered kill criteria are the antidote to selection, not a decoration, and Appendix F collects them so they can be checked against what we report. Four instrument-discipline precedents we follow: reporting the errors a method *introduces* alongside those it repairs, as with a checker that repairs 197 selection errors and introduces 54 for a net gain [CALVER]; validating an estimator against planted ground truth before deploying it where the answer is unknown, with a null at zero dose and a monotone dose-response above it [Temporal Leakage]; running a **parallel negative-control stream inside every cell** of a factorial, so that a cell counts as a detection only if the control does *not* fire, together with a script that re-derives every published statistic from raw data [Shift Monitor]; and testing stability in **both** directions, since "a measurement that changes with every harmless rewrite is unstable, while one that never changes is uninformative", pairing paraphrase and distractor invariance against omission and contradiction *responsiveness*, where withholding relevant evidence must degrade the estimate and contradictory evidence must move it in the correct direction [Semantic Map].

## Appendix F: Pre-registered kill criteria, and the experiment register

Collected so they can be checked against what we report rather than reconstructed from the prose. This appendix is also the **register** for the experiment identifiers used in §9's table: each entry below states what the experiment tested and the criterion that would have killed it.

The composition experiments are not in the criteria list, for the reason given below, so their designs are stated here instead. **E5** asks whether consolidation moves the coverage band that type-conditional retrieval issues (C1 against C3). **E7** asks whether retained stale knowledge inflates the evidence set that retrieval must return (C2 against C3). **E10** asks whether consolidation destroys the supersession lifecycle depends on (C1

against C2). **A4** is the backend-selection micro-average that decides which single backend is `best_single`, and **A6** is the abstention arm that tests C3's inadmissibility branch.

**What "fixed in advance" does and does not cover here.** The E1 to E4 criteria were fixed before their experiments ran, and that is the claim this list makes. Three entries are not clean cases and say so in place: E4b's denominator was re-specified on 2026-08-06 after the original was found to be the wrong population, E4c's stability clause was added on 2026-08-04, and E1d's criteria were added on 2026-08-05 when the arm itself was added. **The composition experiments E5 to E10 are absent from this list on purpose: they were designed after the mechanisms they compare, so any criterion for them would be post-hoc, and presenting one as pre-registered would be the exact error Appendix E collects.** Their design is stated above and their outcomes are in §9.

**E4a — Types have different optimal backends (H4a)**

- *Killed if:* An oracle router fails to Pareto-improve on the best single backend. **Stops the thesis, not just the experiment.**
- *Outcome:* **FIRED** on both corpora (§9). Reported as the result. **Basis re-pointed 2026-08-05b:** it fires on the §9 *domination* measurement (always-B1 beats the oracle on latency and throughput at once, in every one of five seeds, with a throughput margin never below 7x; the oracle's accuracy edge is positive and unestablished), not on the accuracy interval, whose power floors are MDE +0.042 (LoCoMo, 0.57 power at the observed lift) and +0.050 (MuSiQue). The interval bounds the effect above; it does not establish absence

**E4b — A cheap learned router captures that headroom (H4b)**

- *Killed if:* R-learned is dominated by any single backend; **or routing exceeds 20% of slot-held SERVICE p95** (re-specified 2026-08-06: the original end-to-end denominator included queue wait, so the same configuration scored 68.9% at 36% utilization and 10.9% at 96%, and a criterion passable by adding load is not a criterion); **or** it fails to beat a latency-matched random mixture
- *Outcome:* Moot on accuracy (conditional on E4a). **FIRES on latency:** under real execution the encoder router spends **58.9–71.5% of slot-held service on routing at every one of seven operating points**, against a 20% budget; backends with no router read 0.0–0.9%. The LLM router was already killed at ~93%

**E4c — The type taxonomy carries information**

- *Killed if:* Low mutual information between taxonomy label and oracle-best backend; misroute cost $\geq$ routing benefit; a high default-bucket rate; **or low pick stability under cluster resampling** (added 2026-08-04: MI over a tied or unstable table is tie-breaking noise)
- *Outcome:* **FIRED on MuSiQue** (0.21 stability, 1 tied family). Not fired on LoCoMo (0.96)

**E4d — Offline training does not bias the router**

- *Killed if:* Warm-started selection drifts toward the expensive backend without an accuracy gain
- *Outcome:* not yet run

**E3 — Per-type conformal coverage is practical**

- *Killed if:* Per-type coverage requires unrealistic calibration data; set sizes inflate to uselessness; or abstention is uncorrelated with true insufficiency
- *Outcome:* **Precondition 1 run (§9). FIRES at per-conversation granularity** (69% of cells starved at $\alpha = 0.05$; the band degenerates to the whole corpus) and **does not fire at type granularity**, where pooling instead under-covers the two hardest types by 20+ points. The guarantee has a measured granularity window. **Precondition 2 (observability) also run: it PASSES (condition number 3.72) while the guarantee it guards fails**, routing by inferred type puts multi-hop at 59.2% against 70.5% for no typing at all, so precondition 1's result is an oracle result and §7.3 now requires the type out of band. **Third clause run and it FIRES:** the structural insufficiency signal is uncorrelated with true insufficiency (AUC 0.495 [0.460, 0.525]) while the model's self-report reaches 61.4% TPR at 8.7% FPR, so on this instrument the channel C3 proposed to replace is the one that

works. **Three further runs bound the result.** Query-local conformal, the obvious way to avoid needing the type at all, reaches nominal worst-type coverage in **no configuration of 25** without starving, so the sixteenfold cost of out-of-band typing is not an artifact of our mechanism choice. Two rebuilt structural channels then failed a substitution bar (one uninformative at any decomposition, one informative but delivering **0.0% TPR** at the operating point), and the second failed a **complementarity** bar as well, carrying no information inside either of the self-report's decision cells. The channel is unfinished on a fair comparison, not a mismatched one

**E2 — Rate-governed eviction is sufficient**

- *Killed if:* Governance fails to prevent cascade or unbounded growth; or provenance retention breaks the latency budget
- *Outcome:* **RUN 2026-08-12 (3,120 generations), rebuilt around a drift-and-reversal phase schedule after [TEPA] showed clean-update benchmarks cannot separate the arms. Two-sided.** Re-promotion is worth **+53.8 points** under reversal (59.0% vs 5.1%), confirming §7.2's transition the day it was specified. But a pre-registered kill **fires**: plain supersession beats keeping-everything by only **+1.3 points** there, so retirement without an inverse is unmotivated, while in the forward phases supersession reaches **100%** against 91% and 87%. A lifecycle state repairs the store's self-description and not the reader's behaviour: stale-serve on the value probe is ~36% for every arm alike. **The growth half of this criterion remains untested**: the design carries two versions per item by construction. A separate C2 write-policy probe is also complete (§7.2): supersession beat overwrite on all three probes including current value, and overwrite fabricates rather than abstains when it lacks history; that tests the primitives, not this criterion

**E1a — The composability gate earns its cost**

- *Killed if:* The gate is no better than binding-info-only at matched budget; or LRU matches the full-graph upper bound
- *Outcome:* not yet run

**E1b — CompSuff measures composability, not artifact**

- *Killed if:* CompSuff fails to predict held-out multi-hop accuracy; the proxy is uncorrelated with correctness on a labeled slice; or a shortcut baseline (answer-only, length-only, evidence-surface) reproduces it
- *Outcome:* not yet run

**E1c — The gate carries evidential value**

- *Killed if:* Its measured false-pass rate under target-absent conditions leaves $\log \Lambda \leq \log k_{\text{eff}}$ too small to support the decisions it gates
- *Outcome:* not yet run

**E1d — A label-free structural predicate is worth preferring to a learned utility gate (added 2026-08-05b)**

- *Killed if:* The decomposability predicate fails to separate from a **fidelity/reconstruction control** under query-distribution shift, that control being the objective family ThinkReset [2607.28642] tested and abandoned; **or** the learned gate's advantage under shift is no larger than in distribution, leaving no shift penalty for label-freedom to exploit. **Refutes C1's framing rather than qualifying it.** An in-distribution loss to the learned gate is *consistent* with the claim and is not a kill
- *Outcome:* **RUN 2026-08-07, re-run 2026-08-18. NO CRITERION FIRES, AND THE SECOND IS NARROWED RATHER THAN PASSED** (n=989). Predicate separates from the fidelity control by **+84.1 [80.5, 87.7]**, 16× its MDE, and on the held-out category by +81.9 [69.8, 96.5]; the first criterion does not fire on any condition. The second is **instantiation-dependent and we report the split**: across corpora the learned gate goes from 1.34 [0.12, 3.14] ahead to **11.62 [9.15, 14.26] behind**, the predicted penalty; across a held-out query category it goes from 1.34 ahead to **6.27 [4.89, 7.78] further ahead**, no penalty at all. Read as a conjunction over both instantiations the criterion would fire, so **C1's label-freedom claim is narrowed to corpus shift** (§7.1) rather

than reported as passed. **Gating is not vacuous**, which is the precondition for the separation above to mean anything: the predicate rejects candidate merges rather than admitting them all, and a gate that never fires would separate from any control trivially. **Limitation found: the predicate does not compose**, beyond ~1,000 merges safe choices chain into a 932-entity component and the ordering inverts

**Run order: E0 (instrument and harness floor) → E4 → E3 → E2 → E1.** This front-loads the kill signal. E4a can end the thesis, so it runs before effort is spent on the mechanisms that presuppose it, and E1 (the most engineering-heavy) runs last.

The front-loading worked, and it cost us the headline claim. E4a fired on the second corpus before any effort went into E4b, E4d, or the answering runs those presuppose, which is what the ordering was for. We record the consequence rather than soften it: the routing claim is the one the design was organized around, and it did not survive its own pre-registered test. What survives is the type-conditional *phenomenon* on dialogue, the instruments, and the three mechanisms C1–C3, whose evaluations do not depend on routing. Whether routing pays on the token and latency axes (never measured on one stack, and where the oracle is already marginally cheaper in retrieved context on both corpora) is a separate live question, not a rescue of this one.

## References

Citation keys are the short names used in the text. Author lists and titles are taken from arXiv metadata for every entry; the five that predate this project's download queue were read off the papers themselves, as were the two statistics references, which are not part of the corpus. Lists of more than four authors are given as the first three and *et al.* Every arXiv entry corresponds to a paper held in the project corpus and, where a verdict is recorded, an Appendix A row.


- [**ACM**] Gaurav Dadhich (2026). Agentic Context Management: Solving Agent Memory and Cost by Treating Them as Lifecycle and Architecture Problems. arXiv:2607.21503
- [**AdaMM**] Zhoujin Tian, Yao Tian, Hao Zhang et al. (2026). Beyond Retrieval: Analytic Memory for Multimodal Agents. arXiv:2607.29440
- [**AgentKVShift**] Nilesh Prasad Pandey, Jason Kong, Lanxiang Hu et al. (2026). AgentKVShift: Efficient KV Cache Reuse for Agentic Memory Systems. arXiv:2607.21604
- [**AREX**] Shuqi Lu, Chaofan Li, Kun Luo and et al. (24 authors) (2026). AREX: Towards a Recursively Self-Improving Agent for Deep Research. arXiv:2607.21461
- [**Argus**] Boxiu Li, Zimo Wen, Yijia Fan and et al. (2026). Argus: A General-Purpose Agentic Runtime for Long-Horizon Reasoning. arXiv:2608.05144
- [**AttriMem**] Qinfeng Li, Yuntai Bao, Xinyan Yu et al. (2026). AttriMem: Attribution-Guided Process Feedback for Agent Memory Learning. arXiv:2607.21106
- [**Binding Information**] Lianghuan Huang, Yihao Li, Saeed Salehi et al. (2026). Formalizing the Binding Problem. arXiv:2606.03976
- [**CAGE**] Blaise Delattre, Cong Wang and Yang Cao (2026). CAGE: Certified Authorization under Typed-Return Uncertainty for Tool-Using Agents. arXiv:2607.29190
- [**CALIBER**] Conor Finlay, Joshua Kurien, Saurabh Dash et al. (2026). CALIBER: Calibrating Confidence Before and After Reasoning in Language Models. arXiv:2606.24281
- [**Calibrated Selective Fact-Checking**] Dekun Yang (2026). Calibrated Selective Fact-Checking via Evidence Chain Evaluation. arXiv:2607.18240
- [**CALVER**] Omatharv Bharat Vaidya, Connor Thomas Jerzak, Zayne Rea Sprague et al. (2026). When Many Answers Are Valid, Voting Fails: Symbolic Verification for Best-of-K Causal Reasoning in LLMs. arXiv:2608.03506
- [**Certified Deferral**] Jianru Shen (2026). Provable Limits and Certified Deferral for Verbalized Uncertainty in Small Language Models. arXiv:2608.05064
- [**Compaction Survey**] Ashwin Gerard Colaco and Nada Lahjouji (2026). What to Keep, What to Forget: A Rate–Distortion View of Memory Compaction in LLMs and Agents. arXiv:2607.08032
- [**Composition Collapse**] Zhe Yu, Wenpeng Xing, Yunzhao Wei et al. (2026). Composition Collapse: Stable Factual Knowledge Does Not Imply Compositional Reasoning. arXiv:2605.26789

- [**Computational Basis of Confidence**] Dharshan Kumaran, Viorica Patraucean, Maks Ovsanikov et al. (2026). The Computational Basis of Confidence in Large Language Models. arXiv:2607.12447
- [**ConfidenceBench**] Matthew ffrench-Constant, Daniel Yang, Xinmeng Huang and Sanyam Kapoor (2026). ConfidenceBench: Evaluating Confidence Calibration in Large Language Models. arXiv:2607.20526
- [**Conformal Fairness Audit**] Lujia Zhong, Xinkai Wang, Shuo Huang and Yonggang Shi (2026). When Is a Conformal Guarantee Fair? Auditing Silent Subgroup Under-Coverage in Alzheimer's Disease Longitudinal Prediction. arXiv:2608.04254
- [**Delivery Not Storage**] Swapnanil Saha (2026). Delivery, Not Storage: Cue-Anchored Working Memory as a Harness Property for Coding Agents. arXiv:2607.20972
- [**DFOT**] Zongheng Guo, Tao Chen, Tianli Li and et al. (2026). When Derived Measurements Mislead: Quantifying and Mitigating LLM Over-Trust with Privileged-Modality Reliability Evidence. arXiv:2607.28421
- [**DynaKRAG**] Yaqi Wu, Xiaolei Guo, Chenyu Zhou et al. (2026). DynaKRAG: A Unified Framework for Learnable Evidence Control in Multi-Hop Retrieval-Augmented Generation. arXiv:2607.06507
- [**EAR**] Ganesh Senrayan, Moyuru Yamada, Ishan Jindal and Kiran Purohit (2026). Exploratory and Assimilating Reflection: Reflective Recall Cycle for Long-term Memory. arXiv:2607.17879
- [**Eigenius**] Hans-Martin Will, Allen L. Brown Jr. and Matthew Fuchs (2026). Eigenius: A Typed Knowledge-Graph DBMS with Epistemic Stratification and Institution-Mediated Reasoning. arXiv:2608.04457
- [**Eviction Cascade**] Ruicheng Ao, Jing Dong, Gan Luo and David Simchi-Levi (2026). Service-Induced Congestion in Memory-Constrained LLM Serving. arXiv:2606.15555
- [**ExtractConf**] Nitesh Kumar (2026). Beyond Logprobs: A Multi-Signal Confidence Engine for LLM-Based Document Field Extraction. arXiv:2606.24420
- [**Governed Shared Memory**] Yanki Margalit, Nurit Cohen-Inger, Erni Avram et al. (2026). Governed Shared Memory for Multi-Agent LLM Systems. arXiv:2606.24535
- [**HiGram**] Xiawei Yue, Boran Wang, Xiaoqing Zhang et al. (2026). Hierarchical Graph Memory for LLM Agents with Path-level Localization and Rewrite. arXiv:2608.05095
- [**LMEB**] Xinping Zhao, Xinshuo Hu, Jiaxin Xu et al. (2026). LMEB: Long-horizon Memory Embedding Benchmark. arXiv:2603.12572
- [**LoBoost**] Vagner Santos, Victor Coscrato, Luben Cabezas et al. (2026). LoBoost: Fast Model-Native Local Conformal Prediction for Gradient-Boosted Trees. arXiv:2602.22432
- [**MemFail**] Ishir Garg, Neel Kolhe, Dawn Song and Xuandong Zhao (2026). MemFail: Stress-Testing Failure Modes of LLM Memory Systems. arXiv:2605.26667
- [**MemHarness**] Rong Wu, Daocheng Fu, Licheng Wen and et al. (2026). MemHarness: Memory Is Reconstructed, Not Replayed. arXiv:2607.28272
- [**One Human, N Agents**] Cesare Zavattari, Alessandro Tommasi and Giuseppe Prencipe (2026). One Human, N Agents: Audit-Budget Allocation for LLM Agent Fleets under Miscalibrated, Correlated Confidence. arXiv:2607.28317
- [**Ontology-Guided Dedup**] Vaibhav Dangaich, Kevin Lewis and Kundeshwar Pundalik (2026). An Ontology-Guided, Deduplication-Aware Extraction Layer for Knowledge Graph Construction from Heterogeneous Documents. arXiv:2607.28662
- [**Phantom Evidence**] Yukiyasu Kamitani and Ken Shirakawa (2026). Phantom Evidence: How and Why Generative AI Manufactures False Positives in Science. arXiv:2607.25991
- [**Prior Laundering**] Ali Siahkoohi and Sina Alemohammad (2026). Prior laundering: learned priors with inherited, undetectable overconfidence. arXiv:2607.21721
- [**Quiet Failure**] Muhammadjon Tursunbadalov and Mustafojon Tursunbadalov (2026). A Quiet Failure in Calibrated Virtual Screening: Marginal Conformal Prediction Under-Covers the Minority Class, and a Class-Conditional Fix Recovers It. arXiv:2607.06605
- [**Reasoning Denoiser**] Junlin Fang, Do Nguyen-Thanh, Xiaogang Xu et al. (2026). Reasoning Denoiser: Denoising Reasoning Traces for Hallucination Detection in Large Reasoning Models. arXiv:2607.22098
- [**Recoverability Collapse**] Pieter van Rooyen (2026). First-Order Recoverability Collapse in Self-Referential Information Decoders. arXiv:2606.24861

- [**Reliability Scales Inversely**] Kushal Chakrabarti (2026). Reliability Scales Inversely: Bigger Language Models Compound Mistakes Faster. arXiv:2607.18292
- [**Retain-or-Consolidate**] Qingcan Kang, Mingyang Liu, Shixiong Kai et al. (2026). Retain or Consolidate? Budget-Dependent Operator Selection for Language Agent Memory. arXiv:2607.17545
- [**RouteGuard**] Anchen Sun and Kaiqi Yang (2026). RouteGuard: Certifying Routing Gain in LLM Multi-Agent Systems When Complementarity Is Not Enough. arXiv:2608.07583
- [**Semantic Map**] Matthew F. Dixon (2026). Calibrating Semantic Uncertainty from Observable Language-Model Probabilities. arXiv:2607.17447
- [**SESA**] Zenghuang Fu, Zhaoyang Li, Qiuyuan Ai et al. (2026). Self-Play Meets Skill Evolution: Self-Evolving Search Agents that Pose, Solve, and Remember. arXiv:2607.29468
- [**Shift Monitor**] Jun Wen Leong (2026). Online Shift Detection and Conformal Adaptation for Deployed Safety Classifiers. arXiv:2606.11949
- [**Short-Term-to-Long-Term Transfer**] Taewoon Kim, Vincent Francois-Lavet and Michael Cochez (2026). Short-Term-to-Long-Term Memory Transfer for Knowledge Graphs under Partial Observability. *Reinforcement Learning Journal*. arXiv:2605.22142
- [**Source-Side Sufficiency**] Joss Armstrong (2026). Source-Side Sufficiency for the Information Bottleneck: Exact Reduction and Finite-Block Equivalence. arXiv:2604.26744
- [**Sparsity Pitfalls**] Elena Merdjanovska, Omar Zaidan and Andreas Rücklé (2026). Evaluation Pitfalls and Sparsity Limitations in LLM-based Confidence Estimation. arXiv:2608.04899
- [**SPIKE-Bench**] Shu Quan et al. (2026). A Blind Spot in Alignment: Quantifying Biosecurity Risks in Large Language Models. arXiv:2608.02684
- [**Supra Cognitive Modes**] Joshua Tobkin and David Yang (2026). Supra Cognitive Modes: A Routed Architecture for Agent Memory. arXiv:2607.19096
- [**SURE-RAG**] Jingxi Qiu, Zeyu Han and Cheng Huang (2026). SURE-RAG: Sufficiency and Uncertainty-Aware Evidence Verification for Selective Retrieval-Augmented Generation. arXiv:2605.03534
- [**SwiftMem**] Anxin Tian, Yiming Li, Xing Li et al. (2026). SwiftMem: Fast Agentic Memory via Query-aware Indexing. arXiv:2601.08160
- [**Sycophancy Dissociation**] Anthony Baez, Sheer Karny and Pat Pataranutaporn (2026). Dissociating the Internal Representations of Sycophancy in LLMs. arXiv:2607.07003
- [**Temporal Leakage**] Zeyu Zhang and Bradly C. Stadie (2026). Temporal Leakage in LLM Backtesting: Measurement, Validation, and Adjusted Scores. arXiv:2608.02985
- [**TEPA**] Yan Zhou et al. (2026). TEPA: Revoking Stale Memories for Conflict-Robust Language Agents. arXiv:2608.07429
- [**Theorist Toolbox**] Moran Koren (2026). Theorist Toolbox: Tools for Agent-Based LLM-assisted economic theory. arXiv:2606.22337
- [**ThinkReset**] Fei Ding, Yongkang Zhang, Runhao Liu et al. (2026). ThinkReset: Learnable Intermediate Interface Construction for Bounded-Context Long-Horizon Reasoning. arXiv:2607.28642
- [**TRACE-ROUTER**] Ritik Raj, Souvik Kundu, Sarbartha Banerjee et al. (2026). TRACE-ROUTER: Task-Consistent and Adaptive Online Routing for Agentic AI. arXiv:2607.22465
- [**Two Axes of Abstention**] Benedikt J. Wagner (2026). Two Axes of LLM Abstention: Answer Correctness and Question Answerability. arXiv:2607.08456