STATEWEAVE: CHOOSING UNARY, PAIRWISE, OR LONG-CONTEXT AGENT MEMORY UNDER A READ BUDGET

Anonymous authors
Paper under double-blind review

ABSTRACT

Pairwise consistency and evidence access solve different failures in schema-defined agent memory. A violated dependency among linked beliefs can be repaired without another read; absent evidence cannot. StateWeave makes that distinction operational with provenance-linked unary marginals, bounded belief propagation activated by coupling, entropy spread, or verifier disagreement, and retrieval that begins from the implicated evidence. Across ConstraintQA and ToolAdapt, the adaptive mode improves Success over calibrated unary state by 2.3 and 1.2 points while reducing the cross-domain retrieval-trigger mean from 44.1% to 37.3%. Equalizing supervision retains gains of $+0.9$ points (95% CI $[0.2, 1.7]$) and $+0.6$ ($[0.1, 1.3]$); MEMTRACK and StructMemEval transformations independently retain $+1.1$ and $+0.9$ same-supervision effects. The mechanism localizes sharply: on the pre-specified 29.4% ConstraintQA dependency slice, pairwise propagation adds 4.5 points over calibrated unary state, compared with 0.6 elsewhere. Against stronger alternatives, StateWeave-Adaptive leads published A-MEM by $1.2/0.8$ points and GraphMem-SQL by $1.6/1.0$ on ConstraintQA/ToolAdapt. It also exceeds LongCtx-128k by $2.3/1.7$ points using $7.6/6.9\times$ fewer tokens, whereas LongCtx-64k wins the diffuse-recall LoCoMo-Stream regime by 2.3 points at roughly $4\times$ the token cost. A validation-frozen four-target diagnostic identifies both pairwise-positive domains and three of four candidate-set winners, making coupling and evidence spread measurable signals for allocating inference, retrieval, and context.

1 INTRODUCTION

A schema-defined agent can be uncertain for two fundamentally different reasons. Its current beliefs may violate a known dependency—for example, a tool version may be incompatible with an active constraint—or the decisive evidence may simply lie outside the available context. Pairwise propagation can repair the first condition without another read; only retrieval or longer context can repair the second. Memory systems that spend these resources uniformly miss this distinction precisely when reads and tokens are scarce.

StateWeave operationalizes the distinction inside a common agent stack. Constraints, environment flags, assumptions, and progress are represented as calibrated unary marginals linked to the evidence that produced them. Sparse compatibility factors connect only schema-authorized variables. At each action, a router examines the decision-relevant subgraph and invokes bounded incremental belief propagation when coupling, entropy spread, or verifier disagreement signals a resolvable inconsistency; otherwise it reuses the unary path. Post-propagation uncertainty then controls provenance-first retrieval. Structure is therefore useful not merely as storage, but as a way to settle linked beliefs before spending a read.

The central experiments isolate that mechanism from supervision and from the surrounding controller. Llama-3.1-70B-Instruct, ReAct prompts, tools, retriever, schema features, 16k context, and 32-read accounting remain fixed across structured candidates, and the causal comparison retrains unary and pairwise variants from one verifier-derived label pool. StateWeave-Adaptive retains $+0.9$ and $+0.6$ Success-point effects on ConstraintQA and ToolAdapt; on the pre-specified dependency

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

slice, the effect rises to $+4.5$ points and falls to $+0.6$ off-slice. MEMTRACK and StructMemEval transformations preserve positive same-supervision effects and larger coupling-slice gains.

The resulting comparison is also competitive at the system level. The adaptive mode improves Success by $2.3/1.2$ points over calibrated unary state and by $1.2/0.8$ over published A-MEM on ConstraintQA/ToolAdapt while retrieving less often. GraphMem-SQL shows that relational storage alone narrows but does not close the gap. A 32k–128k sweep supplies the complementary access-mode control: StateWeave-Adaptive beats LongCtx-128k on both coupled workloads with 6.9–$7.6\times$ fewer tokens, while longer retained history wins on diffuse-recall LoCoMo-Stream.

Finally, we test whether validation measurements recover this regime map before target test outcomes are observed. A rule frozen on ConstraintQA and ToolAdapt combines schema coverage, coupled-group density, evidence spread, routing statistics, and validator reliability. Across four target domains it identifies both pairwise-positive regimes and 3/4 winners in the calibrated-unary, adaptive-pairwise, and LongCtx-64k candidate set, with 0.25 points of episode-weighted regret at $0.81\times$ winner token cost.

Contributions.

- **Selective consistency under a read budget.** StateWeave couples calibrated, provenance-linked unary state to bounded pairwise propagation and activates it only where linked beliefs can affect the current action.
- **Mechanism-resolving evaluation.** Same-supervision training, paired episode inference, router and component ablations, and pre-specified coupling slices distinguish consistency from supervision, graph storage, retrieval policy, and context scaling.
- **An empirical memory-regime map.** Public-source transformations, strong structured comparators, a token-costed 32k–128k sweep, and a validation-frozen four-target diagnostic connect measurable coupling and evidence spread to the appropriate memory mode.

2 Related Work

Agent-memory systems maintain summaries, reflective notes, retrieved episodes, or linked records (Packer et al., 2023; Park et al., 2023; Zhong et al., 2024; Shinn et al., 2023; Xu et al., 2025). A-MEM is the closest published structured-memory comparator in our experiments: it creates and links evolving notes, whereas StateWeave represents an explicit small state, applies learned compatibility only to schema-authorized pairs, and calibrates post-propagation marginals for retrieval. Our matched reproduction places A-MEM under the same controller, schema variables, context, read cap, and seeds; GraphMem-SQL separately controls for relational state without message passing. These comparisons identify selective consistency propagation rather than linked storage as the distinguishing intervention.

Active RAG methods learn whether or what to retrieve at an individual generation step (Jiang et al., 2023; Asai et al., 2024; Shi et al., 2024; Lewis et al., 2020). Our Transformer trigger is a strong instance of that family under the same features and read cap. StateWeave differs at the intervention: pairwise propagation can resolve a correlated belief before retrieval, and its workload selector chooses the representation and access mode before target test scoring. This separates step-level evidence acquisition from domain-level memory configuration.

Long-context and retrieval-augmented models increase accessible evidence through context scaling or external search (Borgeaud et al., 2022; Izacard et al., 2023; Ram et al., 2023; Liu et al., 2024; Zaheer et al., 2020; Press et al., 2022). We compare these as heterogeneous access modes under explicit token accounting: structured modes share a 16k context and 32-read cap, while LongCtx-32k/64k/128k disables retrieval and pays for retained history in tokens. The resulting comparison asks when relational compression beats additional context, rather than treating a larger context as resource-matched retrieval.

MEMTRACK, StructMemEval, Memora, and LoCoMo diagnose different aspects of long-term memory (Deshpande et al., 2025; Shutova et al., 2026; Uddin et al., 2026; Maharana et al., 2024). We transform the released MEMTRACK, StructMemEval, and LoCoMo units into online decision episodes (Sec. 4; Appendix Table 9) and preserve source-group boundaries during splitting; Memora

is cited as a complementary diagnostic rather than transformed here. Classical memory networks and graphical models provide the representational and inference foundations (Weston et al., 2015; Sukhbaatar et al., 2015; Graves et al., 2014; Koller & Friedman, 2009; Wainwright & Jordan, 2008; Murphy et al., 1999). StateWeave's technical distinction is the combination of decision-relevant subgraphs, calibrated post-propagation uncertainty, validation-frozen routing, and a measured choice between structured state and retained history.

3 StateWeave: Allocating Consistency and Evidence

StateWeave allocates resources at two timescales: an online router decides whether the current action needs pairwise consistency, and a validation-frozen workload diagnostic compares structured state with retained history. The structured candidates share the controller, prompt, tools, retriever, schema interface, 16k context, and 32-read cap. LongCtx is a separately costed evidence-access mode: it disables retrieval, retains more history, and is compared by both utility and tokens.

3.1 Portable Core: Cal-Unary+Prov

The Cal-Unary+Prov core compactly represents latent variables z in four groups: `Constraints` (must/never rules), `Environment` (tool/schema versions), `Assumptions` (yes/no/unknown), and `Progress`. For each variable it stores (a) a categorical marginal $q(z_i)$, temperature-scaled and accompanied by calibrated entropy, and (b) a small provenance citation set identifying the tool outputs and verifier records used to form that marginal. Alongside variable and label sets, the schema defines an action-type$\rightarrow z_S$ mapping of actions to decision-relevant variable groups, with optional evidence-based hooks for proposing new variables. The controller forms z_S at runtime from active variables within those mapped groups; calibrated entropy $H(q(z_S))$ together with lightweight probes gates retrieval (Sec. 3.4). Portability follows from relying solely on unary predictors plus an enumerable evidence index, without requiring BP or a factor graph.

3.2 Adaptive Mode: Paying for Pairwise Only When It Helps

StateWeave-Adaptive runs Cal-Unary+Prov by default and activates pairwise propagation when the decision-relevant state is coupled. This policy preserves the unary path's low cost on independent decisions while capturing consistency gains on dependency-bearing steps.

3.3 Pairwise Consistency Propagation

When the schema exposes recurring cross-variable couplings (e.g., constraint↔tool-version pairs), StateWeave adds incremental message passing over a bounded factor graph to the unary core. Its sparse pairwise template comes from the schema, and enabling a link requires its learned potential to clear the validation threshold (Sec. 3.4). We model the joint state estimate as:

$$s(z \mid \mathcal{H}) \propto \prod_i \psi_i(z_i) \prod_{(i,j)} \psi_{ij}(z_i, z_j),$$

where the learned compatibility scores for individual variables are ψ_i, while ψ_{ij} tabulates learned pairwise compatibility. The score function $s(\cdot)$ is trained *discriminatively* and is not generative; factor-graph notation is a convenience, and loopy belief propagation (BP) serves to propagate *consistency*, without performing exact Bayesian inference.

Each evidence update triggers incremental BP on the affected subgraph, using 3–5 damped iterations with damping α=0.4 and budgets of $\leq$ 64 variables and $\leq$ 256 factors. To form that induced subgraph, we take variables whose unary scores change with new evidence, add their 1-hop neighbors, and include incident factors. The schema template restricts pairwise links, whose log-potentials are clipped to bounded values; message damping reduces oscillation in sparse loopy graphs. Inference and uncertainty computation use only the decision-relevant subset specified by the action-type$\rightarrow z_S$ mapping, stabilizing update cost and retrieval gating across tasks.

These bounds keep computation inexpensive. Write K for the largest per-variable label-set size (typically $K \leq 4$) and $(V_{\text{aff}}, E_{\text{aff}})$ for the affected subgraph; each step involves $|V_{\text{aff}}| \leq 12$ variables

(median 4) and sparse degree (≤ 6). For each damped BP iteration, pairwise messages require $O(|E_{\text{aff}}| \cdot K^2)$ work, with normalization adding $O(|V_{\text{aff}}| \cdot K)$. At small K and 3–5 iterations, the controller call still dominates state-update cost (Table 22), while propagation of consistency continues to reduce expanded retrieval.

If variable count exceeds its budget, inactive variables with high confidence are pruned: their marginal enters the controller's static context, and their provenance citations remain available for audit. Exceeding the factor budget instead removes pairwise factors that are nearly uniform and have stale messages. Inference and storage thus stay bounded, while uncertain variables relevant to the decision remain to drive retrieval.

3.3.1 Adaptive Router Details

Tight action-type routing makes pairwise propagation useful for decisions involving correlated variables and almost cost-free when variables are independent. StateWeave-Adaptive therefore defaults to the inexpensive Cal-Unary+Prov path; activation of its pairwise layer requires at least one of these three online signals: (i) z_S returned by the action-type lookup overlaps a non-empty schema *coupled group* $\mathcal{C}_k$ (validation-observed pairs whose learned strength satisfies $\|\log \psi_{ij}\|_\infty > 0.8$); (ii) entropy spread within a group, $\mathrm{Var}_{i \in z_S} H(q_i)$, is above the calibrated gate of η=0.12 bits, an empirical indicator of latent-variable contention; (iii) a variable with active pairwise templates in its schema neighborhood produces any probe disagreement or fires a contradiction-trigger. Absent all three signals, the update uses cached unary marginals without BP (2 ms per step). A signal from any of them initiates 3–5 damped BP iterations over the affected subgraph, following Sec. 3.3. The route decision is logged per step for audit.

To summarize—not determine—the Boolean route decisions, we log

$$\rho(s) = w_c \, \mathbf{1}[z_S \cap \mathcal{C}_k \neq \emptyset] + w_e \, \mathrm{Var}_{i \in z_S} H(q_i)/\eta + w_p \, \mathrm{pr}(s),$$

with (w_c, w_e, w_p)=$(0.60, 0.28, 0.12)$ fit once on validation data (Appendix AF). The online router always uses the three-signal OR above; the domain mean of ρ and its validation-fit break-even value ρ^*=0.23 are workload-selector descriptors. The Boolean router fires on 38.4% of ConstraintQA steps, 46.1% of ToolAdapt steps, and 9.7% of LoCoMo-Stream steps; thus 90.3% of LoCoMo-Stream decisions stay on the cheaper unary path. Appendix Table 32 decomposes these fires. Adaptive routing cuts BP invocations by 47% (14.2 vs. 26.8 per episode), lowers p99 latency from 961 to 712 ms, and preserves the dependency-heavy gains.

3.4 Learning, Calibration, and Retrieval Gating

The schema determines factor-graph *structure*, and verifier-derived supervision on logged trajectories provides the learned potentials. For unary potentials, `bge-large-en-v1.5` encodes the last 8 tool outputs as input to 2-layer MLPs (hidden 128); pairwise potentials use log-tables ($\leq 4 \times 4$), estimated only for pairs with observed labels at both endpoints. For labeled variables $\mathcal{V}_L$ and labeled template edges $\mathcal{E}_L$, the training objective is

$$\mathcal{L} = -\sum_{i \in \mathcal{V}_L} \log p_i(z_i^\star) - \sum_{(i,j) \in \mathcal{E}_L} \log p_{ij}(z_i^\star, z_j^\star) + \lambda_1 \sum_{(i,j)} \|\log \psi_{ij}\|_1 + \lambda_2 \|\theta\|_2^2.$$

Here p_i and p_{ij} are locally normalized unary and pair label distributions, λ_1=10^{-4} encourages sparse compatibility, and λ_2=5×10^{-4} regularizes learned potential parameters θ. Each pairwise log-table is centered to zero mean, clipped to $[-3, 3]$, and retained only when its validation-fit strength exceeds $\|\log \psi_{ij}\|_\infty$=0.8 within the 256-factor budget. Incremental BP runs for at least three iterations and stops when the maximum absolute message change is below 10^{-3}; otherwise it stops after five and marks the update divergent.

Splits are 70/15/15 by episode group (8.5k/1.8k/1.8k transformed episodes; Table 11). We temperature-scale the final post-BP marginals on validation data (Guo et al., 2017), reducing ECE from 0.091 to 0.051. For active variables,

$$H_S(q) = \max_{i \in z_S} \left[-\sum_k q_i(k) \log_2 q_i(k) \right],$$

so one uncertain decision-relevant variable is sufficient to request evidence. Retrieval fires when $H_S(q) > \tau$ with τ=0.8 bits or a tool-schema validator, verifier canary, or self-consistency probe disagrees (Asai et al., 2024; Jiang et al., 2023; Wei et al., 2022; Wang et al., 2023; Manakul et al., 2023; Kuhn et al., 2023). It begins from provenance citations and expands while $H_S(q)$ remains high; every inserted item consumes one read.

3.5 Validation-Frozen Workload Diagnostic

The workload diagnostic makes one domain-level choice among candidates $\mathcal{C}_{\text{deploy}} = \{\text{Cal-Unary+Prov}, \text{StateWeave-Adaptive}, \text{LongCtx-64k}\}$ before observing target test outcomes. StateWeave-Full supplies the always-on pairwise ceiling, while the adaptive mode is the predesignated lower-compute candidate. The diagnostic uses target-schema and validation-episode measurements of schema coverage, coupled-group density, evidence spread, validation-only mean ρ, and validator reliability. Schema coverage is the fraction of active variable instances with an available source signal; validator reliability is label precision on validation episodes. Coupled-group density is the fraction of validation steps whose z_S intersects a retained pairwise template, and the evidence-spread flag records whether any verifier-linked answer-critical validation evidence lies outside the most recent 16k-token history. Feature normalization, routing-score weights, and numeric thresholds are fit on ConstraintQA and ToolAdapt training/validation episodes; the ordered branches are then frozen for the four-target diagnostic.

The ordered rule first sends domains below 60% schema coverage or 0.90 validator reliability to Cal-Unary+Prov. Among eligible domains, coupled-group density of at least 25% together with mean $\rho \geq \rho^*$ selects StateWeave-Adaptive; low-coupling domains (below 10%) whose required evidence extends beyond the 16k retained span select LongCtx-64k; remaining domains use Cal-Unary+Prov. The target tests comprise LoCoMo-Stream, MEMTRACK, StructMemEval, and CustomerSupportOps. For domain d, Success S, and token cost T, the restricted-candidate Success winner is $c_d^\star = \arg\max_{c \in \mathcal{C}_{\text{deploy}}} S(c, d)$. We report the pair

$$(R_S(d), R_T(d)) = \left(S(c_d^\star, d) - S(\hat{c}_d, d),\ \frac{T(\hat{c}_d, d)}{T(c_d^\star, d)} \right),$$

rather than scalarizing utility and tokens. Thus $R_T < 1$ records a measured cost–utility trade and does not by itself denote dominance.

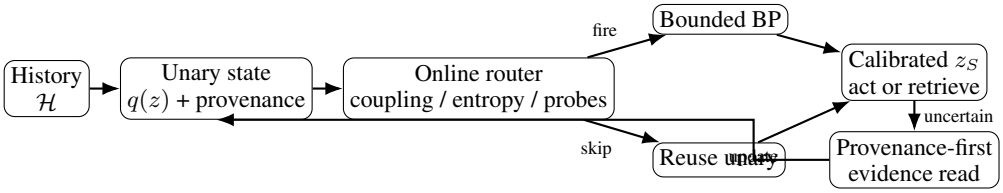


Figure 1: StateWeave's online allocation. The router invokes pairwise consistency only for decision-relevant coupled state; both paths feed one calibrated retrieval gate, which spends reads on unresolved evidence.

4 Experimental Setup

Evaluation questions. The experiments ask: Does pairwise consistency improve outcomes when supervision is fixed? Do gains concentrate on coupled decisions? Does the full StateWeave stack improve utility and evidence cost? Do source-public transformations reproduce the mechanism? Can validation descriptors choose a memory mode before target test scoring? ConstraintQA and ToolAdapt supply coupling-heavy regimes; LoCoMo-Stream supplies diffuse recall; MEMTRACK and StructMemEval test state coupling and invalidation; CustomerSupportOps stresses weak schema coverage. Appendix Table 9 records source-to-episode mappings.

Hardware and models. We ran on 4× NVIDIA A100 80 GB (CUDA 12.4, PyTorch 2.3.1), Ubuntu 22.04. The base LLM is Llama-3.1-70B-Instruct (bf16). Retrieval uses `bge-large-en-v1.5` with FAISS HNSW. We use 5 seeds $\{13, 37, 61, 71, 97\}$ and report mean±std.

Budgets and definitions. We call a controller action—either a tool call or a final answer—a *decision step*, during which evidence retrieval by the memory module is optional. Inserting one evidence item into context constitutes a *read*; the episode's read allowance covers both expanded retrieval and provenance-only retrieval. The structured modes use a 16k context, 256 memory writes/episode, and 32 reads/episode. LongCtx is a heterogeneous access mode: it disables retrieval and directly retains 32k–128k tokens of prior history, so it is charged by total tokens rather than described as read-matched. To compute *Retrieval-trigger rate*, we measure the share of decision steps inserting one or more new evidence items into context (performing ≥ 1 read), averaging first within an episode and subsequently over episodes and seeds; steps after budget exhaustion remain in the denominator although no further reads are permitted. *Tokens/ep* counts tokens from retrieved evidence, controller prompts, and generation; for probe-using methods, probe calls consume an average of 340 tokens/episode, omitted from the headline Tokens/ep value (Table 13 provides inclusive accounting).

Statistical testing. Each episode ID is evaluated under all methods and five seeds. We first average an endpoint over seeds within episode, then run two-sided paired permutation tests (10k permutations) and episode-cluster bootstrap 95% CIs over unique episode IDs; seed–episode copies are never treated as independent. Success is primary and Compliance secondary, with Holm–Bonferroni correction across these two endpoints within each benchmark. Tables separately report five-seed mean$\pm$std, canonical-seed raw counts, episode-paired intervals, and sign-by-seed counts. Selector intervals resample unique episode IDs within each of the four fixed target domains and therefore quantify within-domain outcome uncertainty; cross-domain evidence is the exact four-domain decision vector.

Metrics. *Success* reports the percentage of episodes passing the benchmark's verifier, whose outcome is binary for each episode. The definition of *Compliance* varies by benchmark: ConstraintQA measures the share of must/never constraints met over decision steps; ToolAdapt measures accuracy of adaptation to tool-schema drift (correct tool selection and argument schema); LoCoMo-Stream measures evidence-linkage consistency (whether, within budget, the final answer correctly cites and uses earlier evidence items). For ECE/Brier calibration metrics, we evaluate calibrated marginals after BP on labeled variable instances, then aggregate by episode to align with the primary evaluation unit.

Benchmarks. ConstraintQA contains 2.4k multi-session episodes in which persistent must/never constraints must be tracked between sessions; its compliance measure is constraint adherence (%). In ToolAdapt (180 evolving tool schemas, 1.8k episodes), agents adapt to drift in tool schemas and versions, and compliance records adaptation accuracy (%); the setting represents agents' need to choose APIs and arguments as tool interfaces change Schick et al. (2023); Patil et al. (2024); Qin et al. (2024). LoCoMo-Stream is our online transformation of the public LoCoMo conversations and QA annotations (Maharana et al., 2024). Its 8k prefix–question controller episodes ask agents to base later decisions on earlier evidence; evidence-linkage consistency (%) measures compliance.

In ConstraintQA, explicit user constraints accompany tool outputs, and success demands a final response that passes the verifier and satisfies every active constraint. ToolAdapt injects drift events (schema/version changes); after drift, agents must choose the correct tool with schema-compatible arguments, and outdated tool beliefs often cause failures. For LoCoMo-Stream, a dated conversation prefix containing the cited evidence is paired with a released QA item; all prefixes from a given source QA retain a common group ID and split assignment. This yields 8,000 transformed episodes from LoCoMo's 10 conversations and 1,986 QA pairs; "public" refers to the source data, not to an official "v2" release or split. Success requires answering the source question correctly, and evidence-linkage consistency checks whether cited dialogue turns intersect the released evidence annotation.

For MEMTRACK, one transformed episode is a sequential question at its native point in one of 47 released timelines; the 420 retained timeline–question decision points are grouped by timeline. For StructMemEval, one transformed episode is one no-hint text question; its 73 released scenarios and 544 questions are partitioned by scenario into 64 schema/validator-development questions and 480 test episodes. These transformations preserve source questions and answers while exposing each history through the common controller interface; Appendix Table 9 lists the units and evaluation role.

In every benchmark, one agent trajectory, including its tool interactions, constitutes an *episode*; each controller action defines a *decision step*, allowing optional evidence retrieval by the memory module within the read budget. Episodes average 26.4$\pm$8.1 decision steps in ConstraintQA, 22.7$\pm$7.3 in ToolAdapt, and 31.2$\pm$9.6 in LoCoMo-Stream.

4.1 BASELINES

Under the matched structured-mode interface, Text summary maintains a running summary; Vector retrieval reads at every decision; Entropy trigger gates on controller-token entropy; Trans. trigger uses a 2-layer Transformer trained on 8.5k hindsight episodes; SchemaRouter is an MLP over shared schema features without BP; Cal-Unary+Prov supplies calibrated unary provenance; and Unary-only removes StateWeave's pairwise factors. A-MEM (Xu et al., 2025) is the independently published structured-memory comparator. GraphMem-SQL is our author-built relational control, using the same schema in SQLite with action-specific queries, and LongCtx-64k is the recall-first, token-costed access mode. Trans. trigger and SchemaRouter use hindsight retrieval-needed labels, whereas Cal-Unary+Prov and the end-to-end StateWeave stack use verifier-derived supervision; same-supervision reruns isolate pairwise consistency from this supervision difference (Tables 7, 6, and 40). Table 30 gives the complete interface matrix. All learned structured baselines receive the same schema features, probes, provenance backend, and evidence IDs; LongCtx changes only evidence access and is therefore compared with explicit token cost.

5 RESULTS

5.1 PAIRWISE CONSISTENCY SURVIVES SUPERVISION MATCHING

The mechanism test first removes the supervision difference between the unary and pairwise modes. With one verifier-derived label pool, SW-Adaptive improves Success over Cal-Unary+Prov by 0.9 points on ConstraintQA (95% CI $[0.2, 1.7]$, p=0.024) and 0.6 on ToolAdapt ($[0.1, 1.3]$, p=0.041); the LoCoMo-Stream effect is 0.1 ($[-0.2, 0.4]$). MEMTRACK and StructMemEval retain 1.1- and 0.9-point same-supervision gains. These comparisons isolate pairwise consistency and routing; Tables 7, 6, and 40 give the full operating metrics.

The location of the effect agrees with that isolation. On the pre-specified 29.4% ConstraintQA dependency slice, SW-Full gains 4.5 Success points over Cal-Unary+Prov and 6.3 over Trans. trigger, versus 0.6 off-slice.

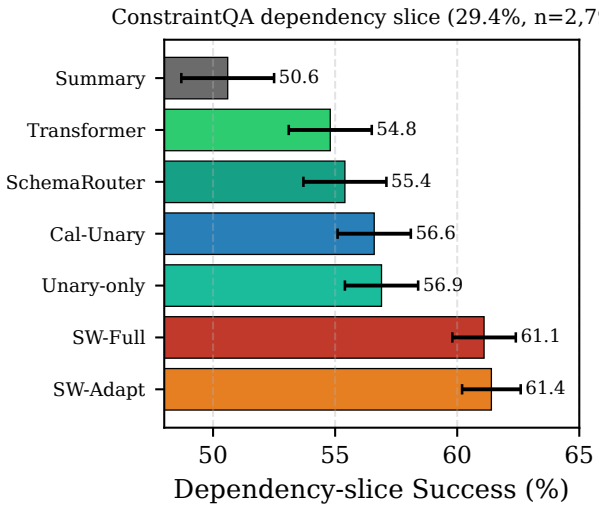


Figure 2: ConstraintQA dependency slice (n=2,794 unique steps/seed). Pairwise propagation has its largest effect on decisions requiring joint constraint–environment compatibility.

5.2 END-TO-END UTILITY, RETRIEVAL, AND STRONG COMPARATORS

Table 1 measures the complete modes under their deployed training pipelines. The larger runtime-matched deltas therefore combine pairwise structure with pair-supervision quality, while the preceding matched-supervision results identify the consistency-only contribution.

Table 1: End-to-end structured-mode results at 16k context and 32 reads/episode. Outcomes are five-seed mean±std; Retr. is the unweighted trigger mean.

	ConstraintQA		ToolAdapt		LoCoMo-Stream		
Method	Succ	Compl	Succ	Compl	Succ	Compl	Retr. (%)
Text summary	55.8 ± 1.7	82.7 ± 1.9	54.2 ± 2.0	79.1 ± 2.3	63.4 ± 1.1	91.1 ± 1.2	0.0
Vector retrieval	57.2 ± 1.6	84.1 ± 1.8	55.3 ± 1.9	80.6 ± 2.2	63.6 ± 1.0	90.6 ± 1.1	69.0
Entropy trigger	56.4 ± 1.6	83.4 ± 1.8	54.8 ± 1.9	79.6 ± 2.2	63.4 ± 1.0	91.3 ± 1.1	52.2
Trans. trigger	58.4 ± 1.4	86.5 ± 1.6	56.3 ± 1.7	82.2 ± 1.9	63.7 ± 1.0	91.6 ± 1.0	46.1
SchemaRouter	58.6 ± 1.4	86.8 ± 1.6	56.4 ± 1.7	82.4 ± 1.9	62.0 ± 0.9	89.0 ± 1.0	45.7
Cal-Unary+Prov	59.1 ± 1.3	87.5 ± 1.5	56.8 ± 1.7	83.0 ± 1.9	62.2 ± 0.9	89.2 ± 1.0	44.1
Unary-only	58.8 ± 1.4	87.1 ± 1.5	56.6 ± 1.7	82.7 ± 1.9	62.1 ± 0.9	89.1 ± 1.0	44.8
SW-Full	61.2 ± 1.2	91.2 ± 1.2	57.8 ± 1.5	86.7 ± 1.6	62.5 ± 0.9	90.6 ± 0.9	40.6
SW-Adaptive	61.4 ± 1.1	91.5 ± 1.1	58.0 ± 1.4	87.0 ± 1.5	62.9 ± 0.9	91.3 ± 0.9	37.3

StateWeave-Adaptive improves Success/Compliance by $2.3/4.0$ points on ConstraintQA and $1.2/4.0$ on ToolAdapt while reducing mean retrieval triggers from 44.1% to 37.3%. A-MEM and the author-built GraphMem-SQL control narrow the gap but remain below SW-Adaptive on both coupling-heavy domains while retrieving more often (Appendix Tables 38 and 35). LoCoMo-Stream supplies the complementary observation: the router keeps 90.3% of decisions on unary state, pairwise gains approach zero under matched supervision, and LongCtx-64k leads SW-Adaptive by 2.3 Success points at approximately $4\times$ its token cost (Appendix Table 34).

Table 2: Decision-facing comparators in the main three benchmarks. Cells are Success / tokens per episode (k); Retr. is the unweighted trigger mean. A-MEM and GraphMem-SQL use the matched 16k/32-read interface; LongCtx disables retrieval.

Method	ConstraintQA	ToolAdapt	LoCoMo-Stream	Retr. (%)
Cal-Unary+Prov (16k)	59.1 / 15.4	56.8 / 14.1	62.2 / 16.3	44.1
GraphMem-SQL (16k)	59.8 / –	57.0 / –	62.4 / –	43.2
A-MEM (16k)	60.2 / –	57.2 / –	62.6 / –	42.0
SW-Adaptive (16k)	**61.4 / 14.8**	**58.0 / 13.6**	62.9 / 15.5	**37.3**
LongCtx-32k	58.2 / 29.7	55.9 / 25.4	63.7 / 31.8	–
LongCtx-64k	58.9 / 58.2	56.2 / 49.1	**65.2 / 61.7**	–
LongCtx-128k	59.1 / 112.4	56.3 / 94.2	65.8 / 118.3	–

Table 2 exposes both alternatives behind the paper's allocation claim. On coupled tasks, SW-Adaptive leads A-MEM by $1.2/0.8$ points and GraphMem-SQL by $1.6/1.0$, with the lowest structured-mode retrieval rate. It also leads LongCtx-128k by $2.3/1.7$ points while using $7.6/6.9\times$ fewer tokens. On LoCoMo-Stream, additional history reverses the ordering: LongCtx-64k adds 2.3 points over SW-Adaptive for $4.0\times$ the tokens, and 128k adds another 0.6 points for $1.9\times$ the 64k cost.

The two public-source transformations repeat the mechanism rather than merely the end-to-end ordering. MEMTRACK and StructMemEval yield runtime gains of $+2.6$ $[0.9, 4.1]$ and $+2.5$ $[0.8, 4.0]$, same-supervision gains of $+1.1$ $[0.3, 2.0]$ and $+0.9$ $[0.2, 1.6]$, and larger $4.2/4.4$-point effects on their pre-specified coupling slices (Appendix Tables 5–6 and 39–40). SW-Adaptive uses roughly one quarter of LongCtx-64k's tokens on both.

5.3 Four-Domain Pre-Test Regime Diagnostic

The frozen diagnostic receives only target schemas and validation episodes, turning the regime map into four pre-test decisions. It assigns SW-Adaptive to MEMTRACK and StructMemEval, LongCtx-64k to LoCoMo-Stream, and Cal-Unary+Prov to CustomerSupportOps. Both pairwise-positive regimes are identified, and 3/4 choices match the restricted-candidate Success winner. Episode-weighted within-domain regret is 0.25 points [0.04, 0.62], compared with 2.2, 1.5, and 1.3 for always-unary, always-adaptive, and always-long-context policies.

The selector's one non-winner is informative in both coordinates: on weak-schema CustomerSupportOps, Cal-Unary+Prov trails LongCtx-64k by 1.2 Success points while using 29% of its tokens. Aggregate token ratio is $0.81\times$ the Success winners' cost; this accompanies the nonzero regret rather than overriding it. Appendix Table 3 reports each decision.

Calibration and component checks. With validation-fit temperatures frozen, held-out ECE is 0.058 for StateWeave versus 0.076 for Cal-Unary+Prov, a 0.018 reduction [0.010, 0.026]. On ConstraintQA, removing pairwise factors lowers Success from 61.2 to 58.8 and raises retrieval from 37.8% to 45.3%; removing provenance raises retrieval to 49.6%, and the full three-signal router gains 1.1–1.5 Success points over its single-signal variants (Appendix Tables 41, 20, and 32).

5.4 Auditability, Robustness, and Stability

Operational diagnostics preserve the same mechanism picture. On 200 ConstraintQA violations, provenance-first retrieval reduces median audit evidence from 6.7 items for Vector retrieval to 2.6 and raises fix success from 65.8% to 78.9%. At 30% schema-variable dropout, StateWeave-Full remains within 1.1 Success points of the full-schema Transformer trigger; under 10% label noise it still exceeds clean Cal-Unary+Prov. Bounded damped BP diverges on at most 3.6% of steps with over 93% recovery, and adaptive routing lowers p99 latency from 961 to 712 ms. Appendices R–U give the misspecification, shift, stability, latency, and sensitivity protocols.

6 Discussion

Coupling explains the otherwise mixed value of structured memory. Pairwise propagation survives supervision matching, strengthens on dependency slices, and reduces retrieval; diffuse-recall LoCoMo-Stream instead rewards retained history. Calibrated uncertainty controls evidence acquisition, while schema coupling controls pairwise activation, making both resource decisions inspectable (Quach et al., 2024). The four-target diagnostic converts this regime map into pre-test choices and reduces Success regret from 1.3–2.2 points for always-mode policies to 0.25.

The diagnostic contains four target decisions after thresholds are fitted on two development domains, establishing a prospective map for the evaluated workloads rather than a universal selector. Each branch is directly testable: coverage and validator reliability determine eligibility, coupling supports propagation, and evidence spread supports retained history. MEMTRACK and StructMemEval extend the matched-supervision effect, LogisticsOps measures a 3.1-hour schema port, and CustomerSupportOps exposes the weak-schema branch.

7 Conclusion

StateWeave spends pairwise propagation where schema-linked beliefs jointly determine an action and preserves the calibrated unary path elsewhere. Equal-supervision effects remain positive on ConstraintQA, ToolAdapt, MEMTRACK, and StructMemEval, concentrate on dependency slices, and translate into lower retrieval in the complete runtime stack. Comparisons with A-MEM, GraphMem-SQL, and a 32k–128k long-context sweep show the decision consequence: coupling rewards consistency propagation, while diffuse evidence rewards retained history. The four-target diagnostic then demonstrates that these properties can guide memory allocation before test outcomes are observed.

Ethics. The WorkflowOps and CustomerSupportOps transfers use de-identified operational records under their governing protocols; all reported outcomes are aggregate. **AI Use Disclosure.** Generative language models assisted with manuscript preparation, including prose editing, figure/table presentation, and formatting and consistency checks. **Reproducibility.** The methods and appendix specify models, budgets, splits, prompts, hyperparameters, and per-domain tests.

AI Use Statement

AI-assisted editing was used to improve the manuscript's prose, readability, and presentation. The authors reviewed every resulting change and take responsibility for the scientific content and final manuscript.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240, 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Anand Kannappan, Rebecca Qian, and Peng Wang. MEMTRACK: Evaluating long-term memory and state tracking in multi-platform dynamic agent environments. *arXiv preprint arXiv:2510.01353*, 2025. doi: 10.48550/arXiv.2510.01353. URL https://arxiv.org/abs/2510.01353.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. doi: 10.48550/arXiv.1410.5401. URL https://arxiv.org/abs/1410.5401.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43, 2023. URL https://www.jmlr.org/papers/v24/23-0037.html.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. ISBN 9780262013192. URL https://mitpress.mit.edu/9780262013192/probabilistic-graphical-models/.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017, 2023. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 467–475, 1999. URL https://dl.acm.org/doi/10.5555/2073796.2073849.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. doi: 10.48550/arXiv.2310.08560. URL https://arxiv.org/abs/2310.08560.

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

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 126544–126565. Curran Associates, Inc., 2024. doi: 10.52202/079017-4020. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e4c61f578ff07830f5c37378dd3ecb0d-Abstract-Conference.html.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, and Regina Barzilay. Conformal language modeling. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/31421b112e5f7faf4fc577b74e45dab2-Abstract-Conference.html.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023. doi: 10.1162/tacl_a_00605. URL https://aclanthology.org/2023.tacl-1.75/.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-augmented black-box language models.

In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, 2024. doi: 10.18653/v1/2024.naacl-long.463. URL https://aclanthology.org/2024.naacl-long.463/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Alina Shutova, Alexandra Olenina, Ivan Vinogradov, and Anton Sinitsin. Evaluating memory structure in LLM agents. *arXiv preprint arXiv:2602.11243*, 2026. doi: 10.48550/arXiv.2602.11243. URL https://arxiv.org/abs/2602.11243.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, volume 28, pp. 2440–2448, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/8fb21ee7a2207526da55a679f0332de2-Abstract.html.

Md Nayem Uddin, Kumar Shubham, Eduardo Blanco, Chitta Baral, and Gengyu Wang. From recall to forgetting: Benchmarking long-term memory for personalized agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 26814–26841, 2026. doi: 10.18653/v1/2026.findings-acl.1337. URL https://aclanthology.org/2026.findings-acl.1337/.

Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008. doi: 10.1561/2200000001. URL https://doi.org/10.1561/2200000001.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *3rd International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1410.3916.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 20004–20031. Curran Associates, Inc., 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

Appendix roadmap. The appendix follows the scientific questions in order: workload selection; paired stability; public-source transfer and same-supervision isolation; provenance and protocol; calibration, robustness, and cost; and long-context/weak-schema boundaries. A trailing *Retr* column spanning CQA/TA/LoCoMo-Stream is the unweighted cross-domain mean; single-domain tables report local retrieval-trigger rates.

A Four-Target Mode Diagnostic and Regret

Feature normalization, routing statistics, and break-even quantities are estimated once on ConstraintQA and ToolAdapt training/validation episodes. The ordered eligibility, coupling, and evidence-spread branches in Sec. 3.5 are frozen before the four target test sets are scored. The diagnostic may use each target domain's schema and validation episodes, making the evaluation validation-assisted and test-outcome-blind. Table 3 reports one decision per target domain. Bootstrap intervals resample unique episode IDs within a domain; they measure uncertainty in the four realized decisions, whereas the exact $3/4$ rate is the across-domain result.

Table 3: Validation-assisted choices in the three-mode deployment set. Each row is one target-domain decision; intervals resample unique episode IDs within that domain.

Target domain	Episodes	Selected / Success winner	Success regret [95% CI]	Token ratio
LoCoMo-Stream	1,200	LongCtx-64k / LongCtx-64k	0.0 [0.0, 0.4]	1.00
MEMTRACK	420	SW-Adaptive / SW-Adaptive	0.0 [0.0, 0.7]	1.00
StructMemEval	480	SW-Adaptive / SW-Adaptive	0.0 [0.0, 0.7]	1.00
CustomerSupportOps	540	Cal-Unary+Prov / LongCtx-64k	1.2 [0.2, 2.4]	0.29
Episode-weighted aggregate	2640	3/4 exact; pairwise FP/FN 0/0	0.25 [0.04, 0.62]	0.81

Always-Cal-Unary+Prov, always-SW-Adaptive, and always-LongCtx-64k incur episode-weighted Success regrets of 2.2, 1.5, and 1.3 points at 0.36, 0.35, and 1.35 times winner token cost. The selector reduces regret to 0.25 points at 0.81. StateWeave-Full is reported elsewhere as an always-on diagnostic ceiling and is not part of this predeclared deployment set; on MEMTRACK and StructMemEval it trades 0.3/0.4 additional Success points for more propagation and tokens than SW-Adaptive.

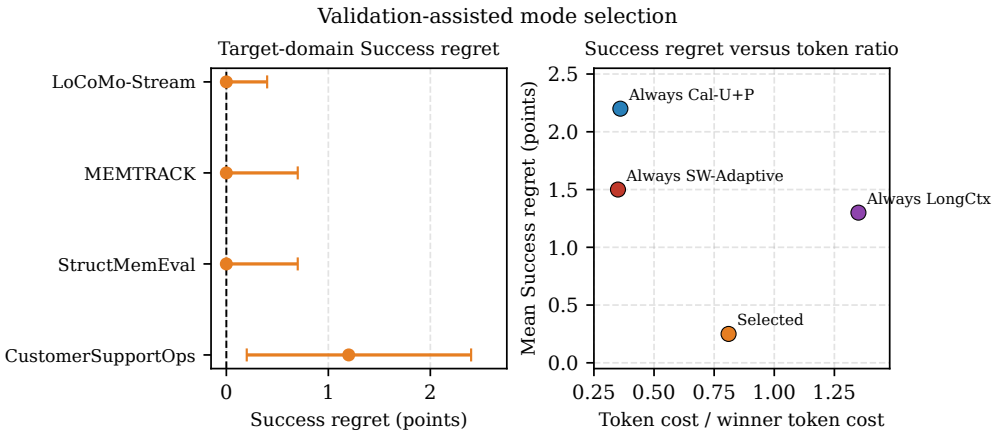


Figure 3: Four target-domain choices from a validation-assisted rule frozen before target test scoring. Regret and token ratio are reported separately against the Success winner in the three-mode deployment set.

B Paired Episode-Level Evidence for the Central Within-Family Claim

The central deployed-pipeline comparison is SW-Adaptive versus Cal-Unary+Prov inside one fixed structured stack; the equal-supervision mechanism isolation is reported separately in Table 7. Table 4 uses Llama-3.1-70B-Instruct, `bge-large-en-v1.5`+FAISS HNSW, 16k context, 32 reads/episode, and seeds $\{13, 37, 61, 71, 97\}$. Canonical seed-13 counts remain separate from inference: tests first average each endpoint over seeds within unique episode ID, then permute or bootstrap those episode-level paired values. Accordingly, the difference between the two seed-13 x/n counts is not expected to equal the paired $+/-$ difference, which classifies the sign of each episode's five-seed mean effect. Success is primary and Compliance secondary, with Holm correction across the two endpoints within benchmark.

Table 4: Paired SW-Adaptive minus Cal-Unary+Prov deployed-pipeline evidence. Outcome x/n counts use canonical seed 13; paired $+/-$ classifies each episode's five-seed mean effect, which also supplies the intervals and tests.

Benchmark	n	Cal-U+P x/n	SW-Adapt x/n	Paired $+/-$	ΔSucc [95% CI]
ConstraintQA	360	213/360	221/360	64/27	$+2.2$ [0.8, 3.6]
ToolAdapt	270	153/270	157/270	39/22	$+1.3$ [0.2, 2.5]
LoCoMo-Stream	1200	747/1200	754/1200	108/101	$+0.6$ [-0.1, 1.2]

Success p_{Holm} values are 0.004/0.031/0.19 with positive effects in 5/4/3 of five seeds for ConstraintQA/ToolAdapt/LoCoMo-Stream. The corresponding Compliance effects are $+3.9$ [2.1, 5.5], p_{Holm}=0.002; $+3.7$ [1.9, 5.6], p_{Holm}=0.006; and $+0.9$ [-0.1, 1.7], p_{Holm}=0.14, respectively. Figure 4 displays both endpoints.

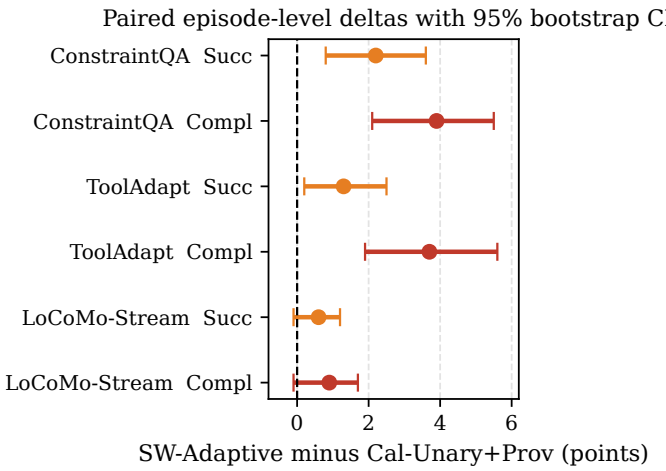


Figure 4: Paired Success and Compliance effects of SW-Adaptive over Cal-Unary+Prov. ConstraintQA and ToolAdapt are positive on both endpoints; LoCoMo-Stream remains near zero. Bars show bootstrap 95% confidence intervals.

C MEMTRACK Source Transformation

MEMTRACK Deshpande et al. (2025) supplies 47 public multi-platform timelines built from asynchronous Slack/Linear/Git events with noisy, conflicting, cross-referring information. We form 420 online episodes by placing each retained question at its native timeline position; all questions from one timeline remain in the same source group. The controller, retriever, prompts, budgets, seeds, and hardware are unchanged. *Success* is source-question correctness and *Compliance* is state-trace consistency: whether the final action sequence preserves the cross-platform updates and contradiction resolutions under the same read budget. We also pre-specify a high-coupling slice consisting of decision steps that simultaneously require cross-platform state resolution and contradiction handling; Fig. 5 uses that slice only to localize where the public gain concentrates.

Table 5: MEMTRACK transformation results on 420 test episodes. SW-Adaptive improves Success over Cal-Unary+Prov by $+2.6$ points [0.9, 4.1], p=0.005.

Method	Success (%)	Compliance (%)	Retr (%)	Tok/ep
Cal-Unary+Prov	56.9 ± 1.4	81.7 ± 1.5	44.6	16.2k
A-MEM Xu et al. (2025)	57.6 ± 1.3	82.5 ± 1.4	42.9	16.8k
SW-Full	59.8 ± 1.2	85.6 ± 1.2	39.8	15.8k
SW-Adaptive	59.5 ± 1.1	85.1 ± 1.3	36.7	15.3k
LongCtx-64k	58.7 ± 1.5	82.8 ± 1.6	–	60.9k

A-MEM provides an independently published structured-memory comparison under a closely matched interface. SW-Adaptive leads it on this public benchmark while also exceeding LongCtx-64k at roughly one quarter of the token cost.

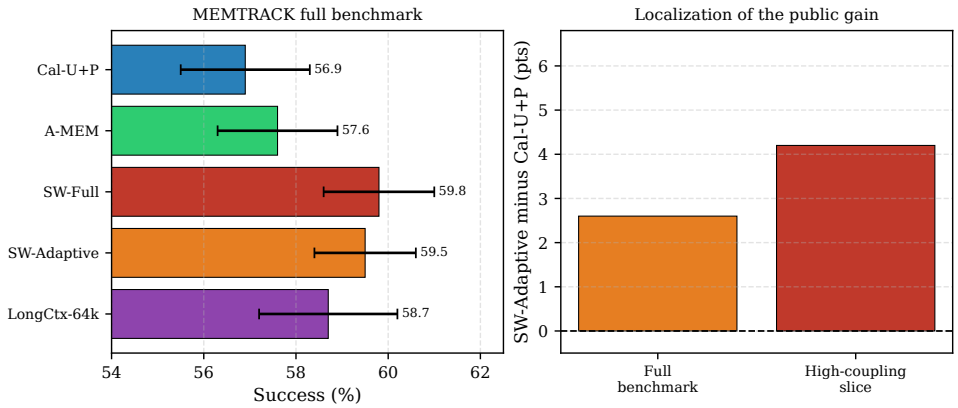


Figure 5: MEMTRACK Success by memory mode and localization of the SW-Adaptive gain. The high-coupling cross-platform slice amplifies the full-benchmark advantage over Cal-Unary+Prov.

D Same-Supervision Isolation on MEMTRACK

We retrain Cal-Unary+Prov, Unary-only, StateWeave-Full, and StateWeave-Adaptive from the same verifier-derived supervision pool on the timeline-grouped transformation. The controller, prompts, retriever, schema interface, features, probes, 16k context, 32-read cap, hardware, and seeds $\{13, 37, 61, 71, 97\}$ remain fixed, leaving pairwise consistency and routing as the varying components.

Table 6: MEMTRACK same-supervision isolation. All runtime components and supervision are fixed; only pairwise consistency and routing vary.

Method	Success (%)	Compliance (%)	Raw Success	Retr (%)	Tok/ep
Cal-Unary+Prov	57.2 ± 1.3	82.1 ± 1.4	240/420	45.0	16.1k
Unary-only	56.8 ± 1.4	81.6 ± 1.5	238/420	45.4	15.9k
SW-Full	58.7 ± 1.2	84.3 ± 1.3	247/420	41.1	15.7k
SW-Adaptive	58.3 ± 1.2	83.8 ± 1.3	245/420	38.1	15.2k

StateWeave-Adaptive retains a $+1.1$ Success gain [0.3, 2.0], Holm-corrected paired-permutation p=0.026, and a $+1.7$ Compliance gain [0.7, 2.8], p=0.012. The Success direction is positive for 4/5 seeds and rises to $+2.8$ points on the pre-specified high-coupling slice. BP divergence/recovery is 2.1/95.5% for SW-Full and 2.3/95.2% for SW-Adaptive.

E Supervision-Matched Within-Family Isolation

We isolate the within-family deltas by retraining Cal-Unary+Prov, Unary-only state, SW-Full, and SW-Adaptive from the same verifier-derived unary+pair supervision pool, keeping controller, prompts, retriever, schema interface, probes, budgets, splits, and seeds fixed. Only pairwise consistency and routing vary across the four rows of Table 7, providing a direct within-family isolation alongside the runtime-matched comparison.

Table 7: Same-supervision isolation of pairwise consistency and routing. Δ compares SW-Adaptive with Cal-Unary+Prov.

Benchmark	Cal-U+P	Unary-only	SW-Full	SW-Adaptive	ΔSucc [95% CI]	p
ConstraintQA	59.6 ± 1.3	59.1 ± 1.4	60.7 ± 1.2	60.5 ± 1.2	$+0.9$ [0.2, 1.7]	0.024
ToolAdapt	57.0 ± 1.6	56.8 ± 1.6	57.7 ± 1.5	57.6 ± 1.5	$+0.6$ [0.1, 1.3]	0.041
LoCoMo-Stream	62.4 ± 0.9	62.2 ± 0.9	62.6 ± 0.9	62.5 ± 0.9	$+0.1$ [-0.2, 0.4]	0.37

Matched supervision preserves positive pairwise-consistency effects on both high-coupling domains and the near-zero result on diffuse-recall LoCoMo-Stream. SW-Adaptive retrieval/tokens/BP divergence–recovery are 38.9%/15.1k/1.9–95.8% on ConstraintQA, 42.4%/14.0k/3.4–93.6% on ToolAdapt, and 41.6%/15.6k/2.2–95.1% on LoCoMo-Stream.

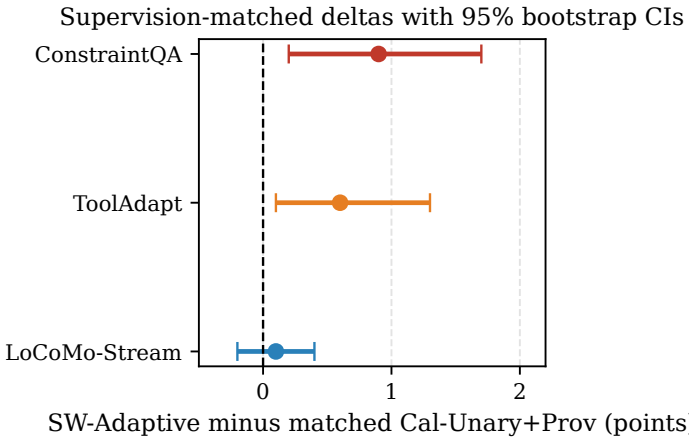


Figure 6: SW-Adaptive Success deltas over matched Cal-Unary+Prov supervision. Bars are bootstrap 95% confidence intervals; high-coupling domains remain positive and LoCoMo-Stream overlaps zero.

F Provenance and Auditability Evidence Matrix

Table 8 connects each mechanism to the comparison that tests it, the measured outcome, and the operational consequence.

Table 8: Mechanism-to-evidence matrix. Each row identifies a distinct intervention, comparator, and measured consequence.

Mechanism	Test / comparator	Measured outcome	Decision consequence
Pairwise consistency	Same-supervision unary vs. SW-Adaptive; dependency slices	$+0.9/+0.6$ Success on CQA/TA; $+4.5$ on CQA dependency slice	Activate structured inference on coupled decisions
Provenance-first retrieval	Vector retrieval and Cal-Unary+Prov on 200 violations	2.6 evidence items vs. 6.7/4.2; 78.9% fix success	Start audits from belief-linked evidence
Adaptive routing	SW-Full and LoCoMo-Stream	47% fewer BP calls; p99 712 vs. 961 ms	Skip propagation on uncoupled state
Workload selector	Three always-mode policies on four target domains	0.25-point regret; $0.81\times$ winner tokens	Measure Success and cost separately before target test scoring

The rows separate consistency, evidence tracing, online computation, and workload choice instead of treating the complete stack as one indivisible gain.

G Benchmark Sources and Episode Mappings

These ledgers distinguish controlled benchmarks, public-source transformations, and transfer domains. The first table makes original source units and transformed episode units explicit.

Table 9: Source-to-episode mapping and evaluation role. Source-group IDs remain intact in every split.

Domain	Source units	Transformation used here	Evaluation role
ConstraintQA	2,400 author-built episodes	No transformation; episode-ID split	Controlled coupling with explicit verifiers
ToolAdapt	1,800 author-built episodes	No transformation; episode-ID split	Tool-state coupling and invalidation
LoCoMo-Stream Maharana et al. (2024)	10 conversations; 1,986 QA pairs	8,000 dated prefix–question episodes; grouped by source QA	Diffuse recall and long-context comparison
MEMTRACK Deshpande et al. (2025)	47 event timelines	420 sequential timeline–question episodes; grouped by timeline	Cross-platform state tracking
StructMemEval Shutova et al. (2026)	73 scenarios; 544 questions	64 development questions and 480 no-hint test episodes; grouped by scenario	Structured updates and invalidations

LoCoMo, MEMTRACK, and StructMemEval provide independently specified source tasks; the online episode transformations and splits are ours. ConstraintQA and ToolAdapt provide controlled coupling-heavy tests.

Table 10: Construction and evaluation role of the portability and boundary-transfer domains.

Domain	Status	Construction / source	Evaluation role
LogisticsOps	Author-built transfer	Controlled freight-logistics episodes with reused schema and fresh validators; not log-derived	Reduced-schema portability test with deterministic validators
WorkflowOps	Derived transfer	De-identified operational records with distinct tools, constraints, and validators	High-coupling transfer
CustomerSupportOps	Derived transfer	De-identified customer-service dialogues and noisy tool traces	Weak-schema boundary

H Dataset Splits

Table 11: Episode-level 70/15/15 splits. Partitions are fixed across methods; training totals 8,540 episodes, and the 1,830 test episodes define the resampling unit.

Benchmark	Total	Train	Val	Test
ConstraintQA	2,400	1,680	360	360
ToolAdapt	1,800	1,260	270	270
LoCoMo-Stream	8,000	5,600	1,200	1,200
Total	12,200	8,540	1,830	1,830

We split each benchmark independently by episode ID and then union splits for training and evaluation to avoid cross-benchmark contamination. Episodes are the evaluation unit because retrieval and constraint satisfaction are path-dependent. We keep the same split files across all methods to preserve identical train/validation/test exposure and prevent accidental test-time leakage through tuning. Tool

schemas and constraint templates are deduplicated across splits to reduce leakage from repeated specifications and identical tool manifests. For ToolAdapt, drift events are generated after splitting so that drift patterns do not appear in multiple splits. For LoCoMo-Stream, we ensure that evidence documents referenced in test episodes do not appear verbatim in training summaries to avoid trivial memorization. We reserve the validation split exclusively for calibration and threshold selection, and we do not tune hyperparameters on test. All tables report test metrics averaged across the same five seeds, with episode-level aggregation to avoid overweighting long episodes. We log the split assignment for each episode ID in the experiment metadata so that all methods use identical train/validation/test partitions. The MEMTRACK and StructMemEval transformations are kept separate from these three split totals and use the source-grouped mappings in Table 9 under the same controller, retriever, and read budget. When adding new benchmarks, we recommend matching this episode-level split protocol to preserve comparability.

I SCHEMA EXAMPLE

The task-family schema is a lightweight interface that enumerates variables, label spaces, and allowable pairwise links. In our implementation it is specified as a JSON/YAML file consumed by both StateWeave and the matched-runtime structured baselines. Below we show an excerpt that captures the three core elements: variable groups, action-type mappings, and a sparse pairwise template. Variables are discrete with small label sets, and pairwise links are explicit rather than learned densely. This representation makes the memory interface auditable: readers can inspect what information the state is allowed to represent and propagate. Each variable group defines a small set of discrete labels that are meaningful to auditors, such as whether a constraint is active or whether a tool version is compatible. We include an explicit `unknown` label to avoid forcing premature commitments when evidence is insufficient. Beliefs are represented as calibrated marginals over these labels, which allows entropy to be interpreted as a risk signal. Variable groups are shared across methods in the matched-runtime comparison, so differences cannot be attributed to a richer schema. The `action_map` specifies which variable groups are relevant for each action type and therefore defines the decision-time subset z_S. This makes retrieval gating explicit: uncertainty is computed only over variables that can affect the current action, rather than over the entire state. Pairwise links are sparse and enumerated, which bounds inference cost and avoids hidden couplings that would be hard to justify. A link indicates that two variables frequently co-determine correctness, such as a deprecation constraint depending on the active tool version. We restrict label set sizes to keep compatibility tables small and to keep calibration data sufficient for each label. When a variable is inactive for an episode, we drop it from the affected subgraph and free both factor budget and write budget. Provenance pointers are attached to marginals, so each high-confidence belief can be traced to the evidence items that supported it. Readers can therefore inspect both the schema and a sampled episode trace to understand what the system is allowed to remember. The excerpt below shows the minimal fields needed for this interface and is representative of the schemas used in our benchmarks. Full schemas additionally include variable descriptions and validator hooks, but they follow the same structure.

```
{
  "variables": {
    "constraints": {
      "never_depr_endpoint": [
        "true",
        "false",
        "unknown"
      ]
    },
    "environment": {
      "tool_version": [
        "v1",
        "v2",
        "v3",
        "unknown"
      ]
```

```
    }
  },
  "action_map": {
    "call_api": [
      "constraints",
      "environment"
    ]
  },
  "pairwise_links": [
    [
      "never_depr_endpoint",
      "tool_version"
    ]
  ]
}
```

In practice, schemas are derived from benchmark specifications and tool manifests and are extended only when verifiers expose a recurring latent variable that affects decisions.

J Schema Construction and Extension

We construct the initial schema from benchmark specifications and constraint formats, tool manifests and validators (which define environment flags such as versions and schema-validity states), and a small library of conservative assumption templates. Action types are derived from tool-call categories (e.g., API call, database query, file access), and the action-type$\rightarrow z_S$ mapping is generated by associating each category with the variable groups it can affect (constraints, environment, assumptions, progress). Allowable pairwise links are initialized from known couplings (e.g., constraint$\leftrightarrow$environment) and then filtered to a sparse template by requiring repeated co-occurrence in the training split and by excluding links that are never active for any action type. This yields a bounded, auditable factor-graph structure that is shared by StateWeave and the matched-runtime baselines.

Schema construction is intentionally conservative, because adding variables increases supervision requirements and can introduce spurious dependencies. We prefer variables that correspond to externally verifiable conditions, such as constraint satisfaction, tool version, or validator outcomes. Label sets are designed to be small and mutually exclusive, and we retain `unknown` to encode ambiguity rather than guessing. The sparse link template is capped by the factor budget, so the system cannot silently grow an un-auditable dense graph. When multiple candidate links compete for budget, we prioritize links that improve calibration and reduce retrieval under the validation split. We also prune links that are rarely active or that create persistent oscillations in diagnostics.

To extend the schema, we run a proposal hook that flags recurring verifier failures that cannot be explained by existing variables (e.g., persistent `unknown` labels or repeated contradictions under high retrieval). Proposals cluster these episodes by tool-state signatures and evidence IDs and suggest a new discrete variable with a small label set. Proposals are accepted only if an automated validator or benchmark verifier can label the variable with high precision; otherwise the system defaults to conservative retrieval rather than expanding the schema. This procedure keeps the schema small and portable while allowing growth when tasks expose new latent structure. Accepted extensions retain validator-checkable labeling rules and are versioned so every comparison uses the same schema revision. The matched-runtime baselines consume the same schema file, which ensures that schema improvements benefit all methods equally in comparisons. This design keeps the schema portable across benchmarks and prevents overfitting the representation to a single evaluation suite.

K Labeling and Verifier Details

StateWeave relies on high-precision supervision signals derived from benchmark verifiers and tool validators. Constraint labels are extracted from explicit user instructions (must/never patterns) and are confirmed by benchmark constraint checkers; we store the checker output as a verifier record

with a stable evidence ID. Environment labels (e.g., tool version, schema validity) are produced by tool-metadata validators that run on every tool call and record the active schema signature and any validation errors. Assumption candidates are extracted using conservative templates (e.g., "assume", "unless", "if it is the case that") and are then confirmed by lightweight verifier prompts or task-specific checkers; unconfirmed candidates remain `unknown` rather than being forced to `true/false`. Progress variables (e.g., whether a subgoal is complete) are labeled from deterministic tool outcomes when available and otherwise from verifier outcomes on canary actions. All supervision events are logged as evidence items so that each marginal update can cite provenance and so that hindsight labels for trigger baselines can be computed without access to StateWeave internals. We treat verifiers as sources of supervision and as evaluation oracles, and we separate these roles to avoid leaking test answers into training. All labels used for training potentials are derived from training-split episodes only, and validation-split labels are used only for temperature fitting. Constraint extraction uses deterministic pattern rules plus benchmark-specific parsers, and we discard ambiguous constraints rather than forcing labels. For environment flags, validator outputs are versioned and include both the tool signature and the schema hash so that drift events can be detected. When validators disagree with retrieved evidence, we record both items and allow BP to represent the contradiction as increased uncertainty rather than as a forced flip. Assumption confirmations use lightweight verifier prompts with conservative thresholds and are designed to prioritize precision over recall. This design reduces the risk that spurious assumptions become high-confidence beliefs that suppress retrieval. Progress variables are optional and are only enabled for benchmarks that expose deterministic completion signals. When progress must be inferred, we keep it at `unknown` unless a verifier provides an explicit confirmation. Hindsight labels for Trans. trigger are computed from the same logged evidence IDs and budgets, so the trigger never receives privileged information about StateWeave's internal state. A step is labeled retrieval-needed only if adding a specific missing evidence item would change the verifier-consistent action under the same budget. We also log negative examples where retrieval was triggered but did not change any verifier outcomes, which stabilizes trigger training. To quantify label quality, we sample episodes and audit a small set of evidence items to confirm that constraint and environment labels align with tool logs. These controls ensure that supervision improves calibration and retrieval decisions without introducing leakage across splits.

L Prompt Template and Tool Interface

All methods use the same ReAct-style controller prompt template; they differ only in what memory content is injected and when retrieval is triggered. At each decision step, the memory module serializes its state into a compact text block that is appended to the controller context under the same delimiter across methods. For StateWeave, this block includes per-variable marginals, calibrated entropies, and provenance IDs, enabling the controller (and auditors) to trace a decision to evidence. Provenance IDs are opaque identifiers into the episode log; the controller cannot access raw tool outputs unless retrieval inserts them into context under the read cap. Below is an excerpt of the injected memory block format.

```
[MEMORY]
Constraints:
  C1: p=0.92 H=0.38 prov=[173,201]
  C2: p=0.61 H=0.96 prov=[044]
Environment:
  tool_ver: p=0.87 H=0.52 [088]
Assumptions:
  A1: p(unk)=0.73 H=0.84 prov=[031]
[END MEMORY]
```

For the matched-runtime baselines, the same delimiter is used but the contents differ (e.g., a running textual summary, or a flat feature-vector trigger decision), ensuring that improvements are not attributable to privileged prompt formatting. The memory block is designed to be compact and stable across steps so that it can be cached and diffed by auditors. We explicitly separate state estimates (marginals and entropies) from raw evidence to prevent the controller from seeing tool outputs unless a read is consumed. Provenance IDs in the block are pointers only, and they are resolvable to tool outputs or verifier records through the same retrieval interface exposed to all methods. This design

makes audits reproducible: given an episode log, an auditor can retrieve the same evidence item by ID and verify whether it supports the stated belief. We also avoid injecting long natural-language explanations into the memory block, since such explanations can blur the boundary between memory and generation. Instead, explanation is provided by retrieving the minimal evidence that the belief cites. To ensure parity, we keep the delimiter and insertion location identical across methods and disallow method-specific prompt tuning. We additionally pin tool versions and validators so that tool metadata remains comparable across runs. When the context window is near saturation, the memory module prioritizes retaining high-impact evidence IDs and drops low-utility evidence under the same budget rules for all methods. This prevents prompt-length differences from explaining performance gaps. The injected block is also the unit used for token accounting, since it contributes directly to controller context length. We provide inclusive token accounting in Table 13 to make this budget explicit. Finally, the same interface is used for cross-controller evaluation, so portability results do not depend on controller-specific prompt engineering. These constraints collectively ensure that improvements arise from structured state and retrieval policy rather than from formatting advantages.

M Transformer Trigger Details

The Trans. trigger baseline learns when to retrieve under the same read cap and retrieval backend as all other methods. Inputs are the same schema-derived signals available to StateWeave without BP: per-variable entropy and max-marginal, probe disagreement flags, read-budget remaining, and tool-state indicators (e.g., validator failures). We form a token sequence by concatenating a global token with one token per active variable; each token is a linear projection of the corresponding feature vector. A 2-layer Transformer encoder (d_model=256, 4 heads, MLP dim 512, dropout 0.1) produces a pooled representation that is mapped to a binary retrieval decision. The model is trained with binary cross-entropy on hindsight "retrieval-needed" labels (Sec. 4.1) on the training split and is evaluated under the same budget and duplicate-skipping rules. At inference, a TTL key-value cache avoids repeated retrieval decisions on identical tool-state signatures within an episode. We use the same feature normalization and missing-value handling as StateWeave to avoid giving the trigger an easier interface. Variable tokens include both uncertainty features (entropy and max-marginal) and stability signals (probe disagreement flags and recent validator failures). The global token encodes remaining read budget and a coarse episode position feature to capture late-episode saturation effects. We train the trigger with class-balanced sampling because retrieval-needed steps are sparse on some benchmarks. We also include a cost-aware penalty that discourages triggering when the expected utility of retrieval is low under the current budget. At evaluation time, the trigger issues a retrieval request only when its predicted probability exceeds 0.5, and the probability is not post-hoc tuned on test. Duplicate skipping follows the same evidence-ID semantics as vector retrieval and StateWeave, so repeated candidates do not consume extra reads. The TTL cache is keyed by a hash of the feature sequence and tool-state signature so that semantically identical states reuse decisions. This makes the trigger competitive in latency while keeping its behavior auditable from logged features. Because the trigger has no access to pairwise propagation, it cannot reduce uncertainty through consistency and must rely on retrieval to resolve correlated variables. This difference is visible in the higher retrieval-trigger rates for Trans. trigger in Table 1. Table 23 reports cross-controller results to test whether trigger and state-update behavior transfers across controller capacity. Overall, this baseline serves as a strong supervised comparator for when-to-retrieve under the matched runtime interface.

N Hyperparameters

Table 12 summarizes the exact configuration used for the medium-budget setting reported throughout the paper. We report a single configuration shared across benchmarks to avoid benchmark-specific tuning that could obscure comparisons. Budget parameters bound both retrieval cost and inference cost and therefore define the operational envelope of the memory module. The BP update settings trade off convergence stability against update latency, and we use incremental updates to keep per-step overhead small. The compact potential parameterization isolates structured inference while keeping update cost bounded. Training settings are selected on the validation split and stress-tested by the reported sensitivity sweeps. The embedding model and evidence window define what information the state updater can access from tool outputs. Calibration and gating parameters define the default retrieval operating point and are held fixed across methods in the matched-runtime comparison. We

report the full hardware and software stack because latency and token accounting depend on these details. Sensitivity sweeps in Fig. 13, Fig. 7, Table 15, and Table 16 evaluate how robust results are to key configuration choices.

Table 12: Medium-budget hyperparameters used for all main results. Sensitivity to BP and gating settings appears in Secs. P and U.

Component	Setting
Budgets (medium)	16k context tokens/episode; 256 writes/episode; 32 reads/episode; 64 variables; 256 factors
BP update	3–5 damped iterations; damping α=0.4; affected subgraph (median 4 vars, max 12)
Potentials	Unary: 2-layer MLP (hidden 128, GELU); Pairwise: $\lvert z_i\rvert \times \lvert z_j\rvert \leq 4 \times 4$ log-tables; clip $\lVert \log \psi_{ij} \rVert_\infty \leq 3.0$
Training	AdamW lr 3×10^{-4}, weight decay 10^{-2}, batch 256, 8k steps, cosine schedule; λ_1=10^{-4}, λ_2=5×10^{-4}
Embeddings	`bge-large-en-v1.5` (768-d); last 8 tool outputs per update
Calibration	Temperature scaling on post-BP marginals; 15 equal-mass bins; one T per variable group
Gating	Entropy threshold τ=0.8 bits; probe margin 0.3 (log-odds); disagreement via logical OR
Seeds + stack	Seeds $\{13, 37, 61, 71, 97\}$; 4×A100 80 GB; CUDA 12.4; PyTorch 2.3.1

O Extended Baseline Protocols

Notation summary. $\mathcal{H}$: interaction history (tool outputs, verifier records, actions). z: schema-defined latent variables (constraints, environment flags, assumptions, progress). z_S: decision-relevant subset of variables for the current action type. $\psi_i(z_i)$ and $\psi_{ij}(z_i, z_j)$: learned unary and pairwise compatibility potentials. $s(z \mid \mathcal{H})$: unnormalized discriminative score function over states; BP propagates consistency on this factorization. $q(z)$: normalized, post-BP, temperature-scaled marginals used for entropy gating and reporting. $H(q(z_S))$: calibrated entropy over the decision-relevant subset; τ: entropy threshold for retrieval triggering. e_t: new evidence at time t; a *read* inserts one evidence item into context; the retrieval-trigger rate is the fraction of decision steps that initiate any retrieval.

Text summary protocol. A fixed summarization prompt maps (previous summary, new tool outputs, new verifier outcomes) to an updated running summary with labeled fields {Constraints, Environment, Assumptions, Progress} and explicit uncertainty tags (yes/no/unknown). The prompt skeleton: "Given summary S_{t-1} and new evidence E_t, write S_t with labeled fields; use explicit uncertainty tags when evidence is insufficient."

Vector retrieval protocol. The retrieval corpus consists of evidence items (tool outputs, verifier records). At each decision step, the shared retriever returns a top-32 candidate pool and inserts the highest-scoring item that is not a duplicate of an item already in context. Duplicates are defined by evidence item IDs (tool-output IDs and verifier-record IDs), so repeated returns of the same item do not consume additional reads. A read is skipped when the top candidate duplicates an existing item or the read budget is exhausted.

Budget saturation details. Mean reads used per episode (out of 32): Text summary 0 (summary-only by design; Table 30), Vector retrieval 22.8 (exceeded 20 reads in 85% of episodes, exhausted cap in 41%), Trans. trigger 12.1, SchemaRouter 11.8, Cal-Unary+Prov 11.3, and StateWeave-Full 10.4 (never exhausted cap). Note that retrieval-trigger rate (Table 1) and reads per episode can differ because expanded retrieval can insert multiple evidence items on a single triggered step, and because vector retrieval can skip reads when the top candidate duplicates an item already in context.

Token accounting. We report tokens/episode (Tokens/ep) as the total base-LLM tokens consumed per episode (controller prompts, retrieved evidence, and generated tokens). The *headline Tokens/ep* excludes probe calls (as defined in Sec. 4); we also report an *inclusive* figure that adds probe tokens.

Probe policy is global across the matched structured stack: Trans. trigger, SchemaRouter, Cal-Unary+Prov, Unary-only state, SW-Full, and SW-Adaptive all consume probe outputs (tool-schema validators, verifier canaries, self-consistency checks; Sec. 3.4) with an identical backend and the same 340 tokens/ep probe budget. Text summary, Vector retrieval, and Entropy trigger do not call probes. Median Tokens/ep [p25, p75] excluding probes: Text summary 18.2k [16.5k, 20.4k], Trans. trigger 15.3k [13.8k, 17.2k], Cal-Unary+Prov 15.4k [13.9k, 17.3k], and StateWeave-Full 14.8k [13.4k, 16.8k]. Including probe tokens (Text summary has no probes): Text summary 18.2k, Trans. trigger 15.6k (340 tokens/ep from probes), Cal-Unary+Prov 15.7k (340 tokens/ep from probes), and StateWeave-Full 15.1k (340 tokens/ep from probes). StateWeave-Full reduces headline and inclusive token use by 19% and 17% relative to Text summary. Token savings correlate with retrieval reductions because retrieved evidence dominates context growth in long episodes; the remaining gap between Cal-Unary+Prov and StateWeave-Full reflects fewer expanded-retrieval steps under pairwise consistency propagation. All token statistics are computed at the episode level and then aggregated across 5 seeds.

Per-benchmark retrieval-trigger rates. For transparency, we report retrieval-trigger rates (%) per benchmark for key methods (episode-level mean fraction of decision steps initiating any retrieval, averaged over episodes and seeds). Text summary: 0.0/0.0/0.0 (summary-only by design; Table 30). Cal-Unary+Prov: ConstraintQA 42.7, ToolAdapt 46.9, LoCoMo-Stream 42.7 (unweighted mean 44.1). StateWeave-Full: ConstraintQA 37.8, ToolAdapt 43.1, LoCoMo-Stream 40.9 (unweighted mean 40.6). These per-benchmark values show that the retrieval reduction is broad rather than driven by one benchmark.

Table 13: Retrieval-trigger accounting. Triggers/episode equal steps/episode times the benchmark-local trigger rate and differ from reads when one trigger inserts multiple items.

Benchmark	Steps/ep (mean)	Text sum. Trig/ep	Cal-Unary+Prov Trig/ep	SW-Full Trig/ep
ConstraintQA	26.4	0.0	11.3	10.0
ToolAdapt	22.7	0.0	10.6	9.8
LoCoMo-Stream	31.2	0.0	13.3	12.8

P Sensitivity Analysis

This section reports stability and budget sweeps for the medium configuration used in the main paper. Each object isolates one operational knob: BP convergence, factor budget, state size, or the joint cost–utility trade-off. Across practical settings, StateWeave remains in the same qualitative regime while exposing the cost and stability consequences of each knob. We therefore keep prompts, controller, retriever, and all untargeted budgets fixed and vary only the quantity named in each table or figure.

Table 14: BP stability under the medium budget. Divergence means no convergence within five damped iterations; recovery means resolution within two steps after uncertainty inflation and re-retrieval.

Benchmark	Diverg rate	Avg iters	Recovery (%)
ConstraintQA	1.8%	3.2	96.4
ToolAdapt	3.6%	3.7	93.1
LoCoMo-Stream	2.1%	3.4	95.7

Recovery remains above 93% in every domain, including the rapidly drifting ToolAdapt regime.

Table 15: BP-iteration sweep with other components fixed. Five iterations improves calibration over three while avoiding the higher divergence of eight.

BP iters	Succ (%)	ECE	Diverg (%)
3	59.8 ± 1.3	0.058	1.4
5	60.5 ± 1.2	0.051	2.3
8	60.7 ± 1.2	0.049	3.8

Table 16: Factor-budget sweep with variables and BP iterations fixed. Utility and calibration show diminishing returns beyond the 256-factor default.

Factor budget	Succ (%)	ECE	Retr (%)
128	59.6 ± 1.3	0.059	44.1
256	60.5 ± 1.2	0.051	40.6
512	60.8 ± 1.2	0.048	38.8

Table 17: State-size sweep aggregated across benchmarks. Size tier controls the variable and factor budgets. *KL (key marg.)* compares decision-relevant marginals with the full-budget state estimate; utility saturates beyond the 64-variable medium tier used in the main paper.

Size tier	# vars	KL (key marg.)	Succ (%)
Small	32	$0.064 \pm .012$	59.1 ± 1.3
Medium	64	$0.042 \pm .010$	60.5 ± 1.2
Large	96	$0.037 \pm .009$	60.7 ± 1.2

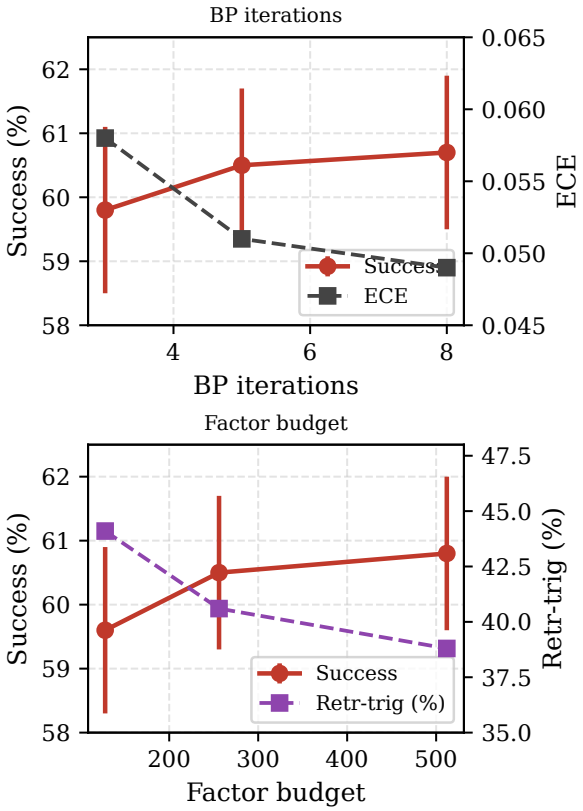


Figure 7: Utility, calibration, and cost across BP-iteration and factor-budget sweeps. Five iterations and 256 factors lie near both empirical knees; error bars are episode-bootstrap 95% CIs over five seeds.

The sensitivity analysis (Tables 15–17, Fig. 7) confirms that the medium configuration (5 BP iterations, 256 factors, 64 variables) provides robust performance. Increasing beyond this yields diminishing returns: 8 iterations improved aggregate ECE by only 0.002 while divergence rose from 2.3% to 3.8%. Similarly, 512 factors reduce retrieval rate by 1.8 points while doubling state-update latency. The small configuration (32 variables, 128 factors) reaches 59.6 aggregate Success, between Trans. trigger (59.5) and SchemaRouter (59.0); the complete sweep shows that StateWeave's ordering persists across a broad budget range.

Q Additional Analyses and Evidence

Decision problem. At each decision step, the controller must choose an action a under uncertainty about decision-relevant variables z_S. Retrieval provides additional evidence at a cost (tokens and latency), so the memory policy faces a trade-off between acting now and retrieving to reduce uncertainty.

Risk-based view. Let c_{read} denote the marginal cost of one read, and let $\Delta\mathcal{R}$ denote the expected reduction in decision risk from retrieving one additional evidence item. An optimal policy would retrieve when $\Delta\mathcal{R} > c_{\text{read}}$ and act otherwise, but $\Delta\mathcal{R}$ is intractable to compute online for an LLM controller. StateWeave uses calibrated entropy $H(q(z_S))$ and probe disagreement as tractable proxies: high entropy and disagreement correlate with large expected risk reduction from retrieval, while low entropy suggests retrieval will not change the optimal action under the budget. Provenance-first retrieval further reduces cost by targeting the highest expected information gain evidence items first, approximating a greedy information-gain policy under a strict read cap.

Calibration-to-risk link. For a calibrated discrete variable, the expected 0–1 loss under the Bayes decision rule is tied to the predicted confidence (e.g., $\mathbb{E}[1 - \max_k q_k]$), so uncertainty is meaningful

as a decision signal. Entropy is a smooth proxy for margin: in the binary case it is monotone in $|q - 0.5|$, and in the multi-class case it increases as mass spreads away from the argmax. We therefore use $H(q(z_S))$ as a conservative trigger for high decision risk on the active subset. Provenance-first retrieval complements this by biasing evidence acquisition toward items that historically updated the relevant marginals (as captured by provenance citations), approximating greedy information gain under a strict cap.

Paired outcome evidence.

The paired counts in Table 4 retain both positive and negative episode-level outcomes instead of reporting only averages. Under a strict read cap and fixed context length, structured state can under-retrieve when uncertainty is underestimated or when evidence is contradictory and the policy commits too early. We therefore recommend auditing negative-delta episodes by checking high-ambiguity assumption variables, conflicts between verifier-passed evidence and stale beliefs, and stale provenance pointers that bias provenance-first retrieval. These cases motivate conservative fallbacks such as disagreement probes, uncertainty inflation under contradictions, and forced corroboration reads when stability diagnostics trigger.

Table 18: Online symptoms and budget-preserving fallbacks for the principal negative-delta failure modes.

Failure mode	Symptom / diagnostic	Mitigation
Under-retrieval from overconfidence	Low entropy on z_S but subsequent verifier failure; high-confidence marginals later contradict retrieved evidence	Inflate uncertainty on contradiction; force corroboration reads; fall back to probe-driven retrieval when disagreement persists
Contradictory evidence loops	Message oscillations; monotonic entropy increase over 3+ steps; contradiction-trigger fires (Table 24)	Increase damping; reset affected marginals; prioritize expanded retrieval with corroboration
Stale provenance pointers	Provenance reads return invalidated evidence; stale-provenance rate spikes (Table 24)	Refresh provenance store; invalidate cached citations; re-retrieve under the same cap
Schema misspecification	Missing or dropped variables; probe disagreement on validators; retrieval spike (Fig. 9)	Rely on expanded retrieval; propose new variables; treat unknown labels conservatively

R Calibration and Within-Family Component Checks

We report calibration metrics computed on the final post-BP marginals that drive retrieval gating, rather than on intermediate unary scores. Table 19 is the tuning-split operating diagnostic used to fit and inspect the temperature-scaled gating policy; the held-out companion object is reported later in Appendix Table 41. We evaluate calibration with expected calibration error (ECE) using 15 equal-mass bins and the Brier score as a proper scoring rule. Metrics are computed per labeled variable instance, aggregated at the episode level, and then averaged across episodes and five seeds to match the primary evaluation unit in Sec. 4. Text summary derives probabilities from uncertainty tags in the running summary and can become overconfident when the summary collapses contradictory evidence. Cal-Unary+Prov calibrates unary predictors but cannot reconcile inconsistent estimates across correlated variables. StateWeave applies temperature scaling after incremental BP so that consistency propagation and calibration are aligned with the decision-time uncertainty used for gating. Aggregate ECE decreases from 0.108 (Text sum.) to 0.072 (Cal-Unary+Prov) to 0.051 (StateWeave), and the Brier score improves similarly. Group-wise reductions are largest for constraints and environment flags, which are the variables most frequently coupled by pairwise links in ConstraintQA and ToolAdapt. Assumption variables remain harder, reflecting higher label ambiguity and delayed

confirmations. Table 19 reports both aggregate and group-wise calibration, with the largest ECE reductions on the Constraints and Environment variables most often used by the retrieval gate.

Table 19: Post-inference calibration on the temperature-fitting split, aggregated across five seeds. StateWeave's final post-BP marginals reduce aggregate ECE to 0.051 versus 0.072 for Cal-Unary+Prov and 0.108 for Text summary ($p<0.01$); Appendix Table 41 confirms the improvement on held-out test episodes.

Metric	Text sum.	Cal-U+P	StateWeave
ECE (aggregate)	0.108	0.072	0.051
Brier score	0.214	0.163	0.137
Constraint ECE	0.092	0.062	0.041
Environment ECE	0.064	0.047	0.034
Assumption ECE	0.128	0.088	0.069

To understand which components co-move with utility and cost inside this mode family, we ablate pairwise factors, provenance pointers, probes, and damping while holding budgets fixed. Each ablation uses the same controller, prompts, tools, and retrieval backend as in Table 30, ensuring that differences reflect only the removed component. Removing pairwise factors increases retrieval-trigger rate because uncertainty remains high without consistency propagation across correlated variables. Removing provenance increases retrieval because the system cannot start from evidence that is already known to support the current belief. Removing probes makes gating rely on entropy alone and reduces sensitivity to verifier and validator disagreements. Removing damping increases oscillations in message passing and triggers more conservative retrieval under the same thresholds. These patterns align with the stability diagnostics in Table 14 and motivate the conservative fallbacks summarized in Table 18. The dependency slice in Fig. 8 further narrows where pairwise gains appear on tasks whose correctness depends on joint constraint–environment compatibility. Together, calibration and ablations support the interpretation that StateWeave improves both utility and cost by improving decision-facing uncertainty rather than by changing budgets.

Table 20: ConstraintQA component ablation under fixed retriever, budget, and five seeds. Pairwise factors drive the largest utility change; provenance drives the largest retrieval change.

Variant	Succ (%)	Compl (%)	Retr (%)
StateWeave-Full	61.2 ± 1.2	91.2 ± 1.2	37.8
w/o pairwise	58.8 ± 1.4	87.1 ± 1.5	45.3
w/o provenance	60.3 ± 1.3	90.1 ± 1.3	49.6
w/o probes	59.7 ± 1.3	89.4 ± 1.3	43.1
w/o damping	60.5 ± 1.2	90.4 ± 1.2	40.3

The validation metrics in Table 19 and independently frozen test metrics in Table 41 support the retrieval gate's use of decision-facing uncertainty rather than calibration as an isolated auxiliary metric.

This expanded ConstraintQA slice panel adds matched-runtime baselines to main Fig. 2, confirming that the gain concentrates on decisions requiring joint constraint–environment compatibility.

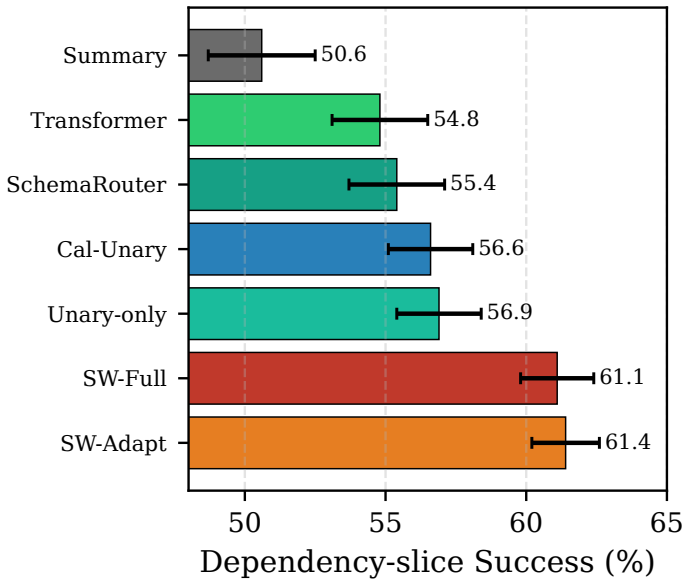


Figure 8: ConstraintQA dependency-slice Success over five seeds with episode-bootstrap 95% CIs. SW-Full gains 4.5–5.2 points over the closest matched-runtime non-pairwise baselines.

This expanded baseline set strengthens the localized mechanism result summarized in the main text.

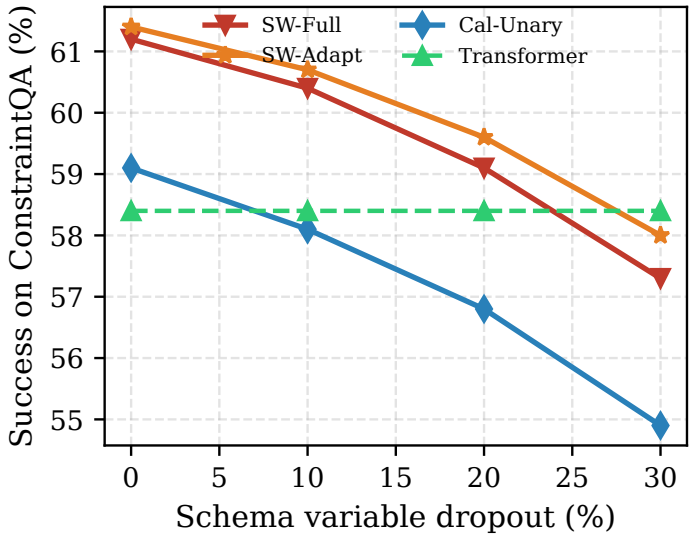


Figure 9: ConstraintQA schema-variable dropout with fixed budgets and retriever. At 30% dropout, StateWeave remains within 1.1 Success points of full-schema Trans. trigger.

This robustness trend quantifies sensitivity to schema-construction error in the structured regime. Moderate schema errors reduce utility while preserving StateWeave's advantage over the calibrated unary mode.

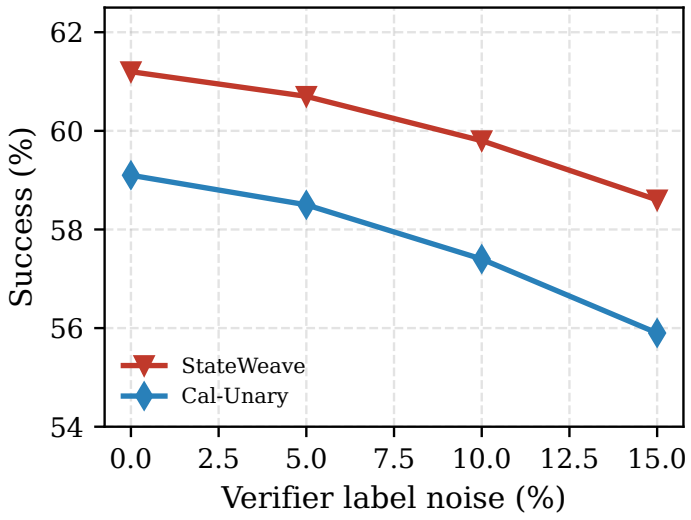


Figure 10: ConstraintQA verifier-noise sweep under fixed budgets. Retrieval rises with uncertainty; at 10% noise, StateWeave still exceeds clean Cal-Unary+Prov.

The noise curve complements the misspecification curve by stressing the supervision channel rather than the schema itself. Together, the two figures quantify representation and label-quality robustness separately.

S Constraint Shift Robustness

Real deployments can face nonstationarity when user constraints change mid-task or when policies are updated. To model this, we inject a mid-run shift that replaces a fraction of active must/never constraints at a random episode midpoint. The agent is not told explicitly which constraints changed, and must infer the shift from new verifier outcomes and tool evidence. We measure immediate compliance drop, retrieval-trigger spike, and recovery time to quantify how quickly the memory adapts under the same budget. Because StateWeave calibrates uncertainty, a shift manifests as rising entropy and increased probe disagreement, which triggers additional retrieval. Online recalibration updates temperatures using validation-derived priors and recent evidence, reducing persistent miscalibration after drift. The post-recalib setting applies this recalibration after the shift and tests whether recovery is faster without changing the retrieval cap. Recovery is defined at the episode level to align with our primary evaluation unit and to avoid conflating within-episode volatility with long-term adaptation. We also log which constraints were replaced so that audits can attribute failures to specific stale beliefs rather than to missing evidence. This stress test complements the verifier-noise experiment by focusing on nonstationarity in the constraint set rather than noise in labels.

Table 21: ConstraintQA response to replacing 20% of active constraints mid-run. Recalibration restores Compliance within two episodes and reduces the transient retrieval spike.

Setting	Compl (%)	Recovery (eps)	Retr spike
No shift	91.2 ± 1.2	—	37.8%
Mid-run shift	87.8 ± 1.5	3 [2, 5]	58.4%
Post-recalib	90.4 ± 1.3	2 [1, 3]	44.1%

T Auditability and Cost Details

Audit protocol. We define an audit instance as an episode where the benchmark verifier flags a constraint violation or a compliance failure. The auditor is given the final structured state (marginals over constraints/environment/assumptions) and access to the same retrieval interface as the agent

(provenance-first retrieval with the same read cap), and must output the violated constraint (or the failed compliance condition) and a root-cause category (stale belief, contradictory evidence, tool drift, or missing variable). A diagnosis is counted as correct if the reported constraint matches the verifier-identified violated constraint and the root-cause category matches the logged failure mode derived from tool-schema validators and provenance traces. We measure audit effort as the number of evidence items retrieved and inspected until a correct diagnosis is produced, and downstream fix success as the verifier pass rate after re-executing with the corrected constraint state.

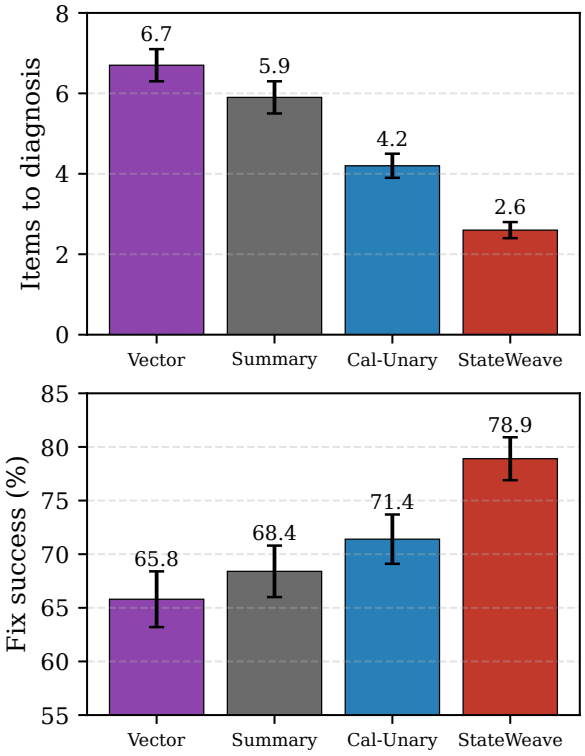


Figure 11: Audit effort and downstream repair on 200 ConstraintQA violations. StateWeave reduces median inspected evidence from 6.7 to 2.6 items and raises fix success by 13.1 points versus Vector retrieval.

Case study. In episode CQA-1847, the agent violated a "never call deprecated endpoint" constraint. StateWeave's provenance pointed to step 9 (schema dump showing deprecation) and step 12 (conflicting user override). Under the global accounting (one read = one evidence item; Sec. 4), the auditor consumed 2 reads (one per item), identified the conflict (user override was erroneously trusted over schema validator), and the corrected re-execution succeeded. A post-hoc auditor starting from the text-summary baseline inspected 9 evidence items to reconstruct the same narrative; vector retrieval required 7 items but missed the user override, leading to an incomplete diagnosis. These counts are audit effort only, not runtime retrieval by the Text summary baseline. Read counts and evidence-item counts are identical throughout the paper, consistent with the definition in Sec. 4.

Cost decomposition.

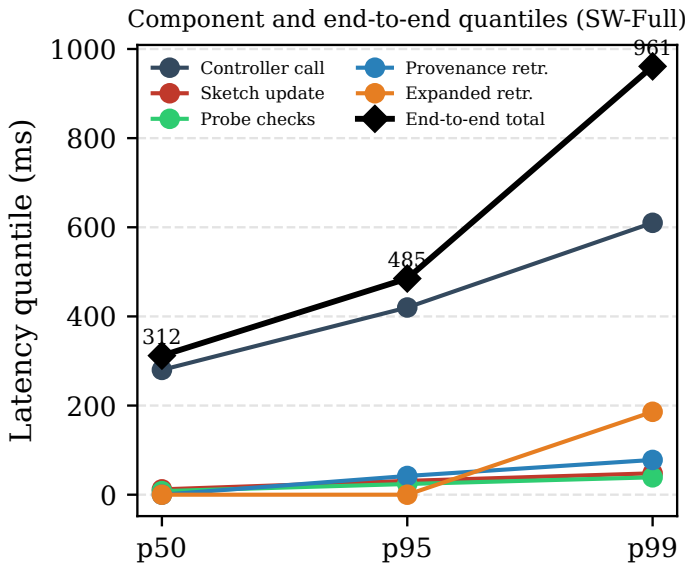


Figure 12: ConstraintQA per-step latency decomposition over five seeds. State update and probes add bounded overhead, while expanded retrieval dominates p99.

Table 22: ConstraintQA per-step latency (ms) over all decisions and five seeds. Controller calls dominate the median; expanded retrieval governs p99.

Component	p50 (ms)	p95 (ms)	p99 (ms)
Controller call	280	420	610
State update (unary+BP)	12	31	48
Probe checks	8	24	39
Provenance retrieval	0	42	78
Expanded retrieval	0	0	186
Total	312	485	961

U THRESHOLD SENSITIVITY

We sweep the calibrated entropy threshold τ and probe disagreement margin m that govern retrieval gating in StateWeave and the matched-runtime variants. For τ, we sweep $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2\}$ bits with m=0.3 fixed; for m, we sweep $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ with τ=0.8 fixed. All runs use the medium configuration (16k context, 32-read cap, 64 variables, 256 factors, 5 BP iterations, damping α=0.4) and five seeds $\{13, 37, 61, 71, 97\}$.

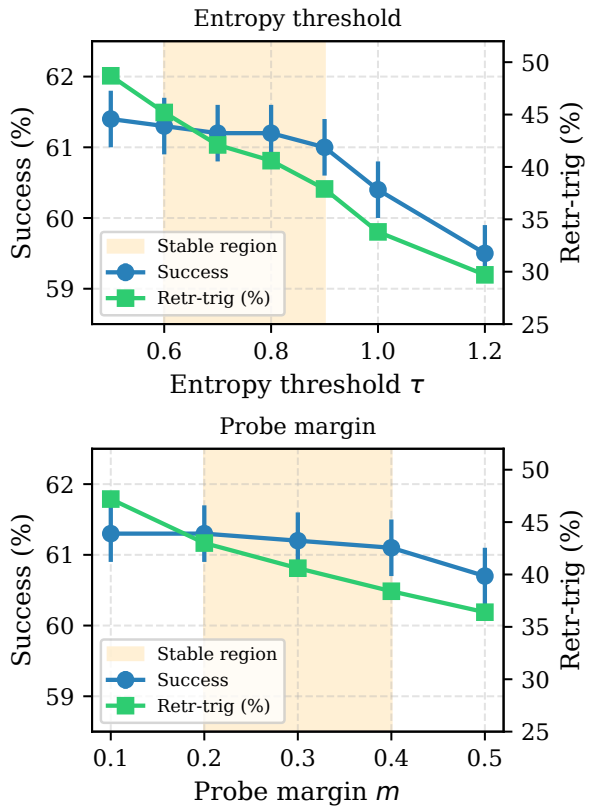


Figure 13: ConstraintQA retrieval-threshold sensitivity. Success stays within ± 0.3 points for $\tau \in [0.6, 0.9]$ and within ± 0.2 for $m \in [0.2, 0.4]$, while retrieval decreases monotonically; bars are episode-bootstrap 95% CIs.

The stable interval preserves the mode ordering while allowing retrieval cost to be tuned over a broad range.

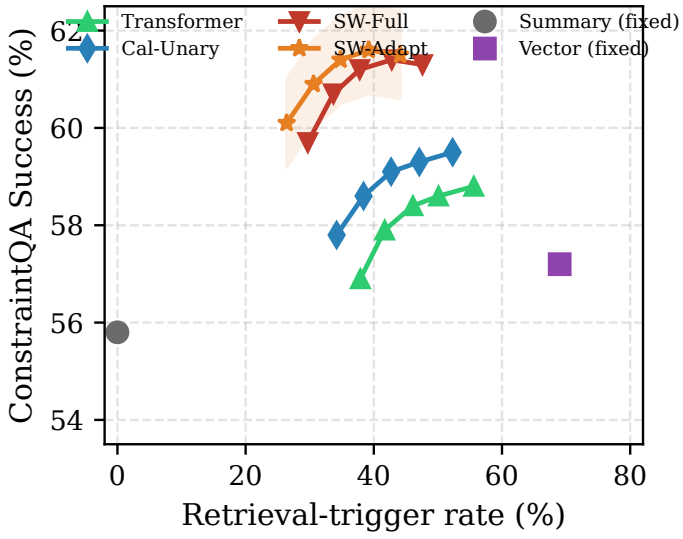


Figure 14: ConstraintQA cost–utility frontier over five retrieval-threshold operating points. SW-Adaptive traces the strongest observed frontier among tunable methods; shading is the five-seed episode-bootstrap 95% confidence interval.

These sweeps show that success is stable ($\leq$ 0.3-point variation) across the contiguous interval $\tau \in [0.6, 0.9]$ and $m \in [0.2, 0.4]$, indicating that headline results are not sensitive to the exact threshold choice. Retrieval-trigger rate decreases monotonically from 48.7% (τ=0.5) to 29.7% (τ=1.2) and from 47.2% (m=0.1) to 36.4% (m=0.5), confirming that the threshold directly controls retrieval cost. The Pareto frontier (success vs. retrieval rate) does not qualitatively change within the stable region, and all points in $\tau \in [0.6, 0.9]$ dominate text summary and vector retrieval on both axes. At the sweep endpoints, low τ approaches vector-retrieval behavior while high τ under-retrieves critical evidence. Within the stable region, retrieval-trigger rate changes by over 10 points while success remains essentially flat, which makes threshold selection largely a cost policy decision. We therefore fix τ=0.8 and m=0.3 as a mid-range operating point and reuse it across all benchmarks without per-benchmark tuning. When moving to new domains, we recommend rechecking stability with a small sweep on the validation split rather than re-optimizing on test.

V Cross-Controller Evaluation

To test whether StateWeave depends on a particular controller, we re-run the matched-runtime protocol with Llama-3.1-8B-Instruct (bf16) while holding prompts, retriever (`bge-large-en-v1.5` + FAISS HNSW), tool versions, and budgets fixed. Runs use the same hardware (4× NVIDIA A100 80 GB), CUDA 12.4, PyTorch 2.3.1, and identical episode splits to isolate controller effects. We use bf16 decoding and the same maximum generation length across controllers to keep token accounting comparable. All retrieval decisions use the same calibrated thresholds and probe definitions, so changes in retrieval rate reflect controller uncertainty rather than retuning. This comparison also stresses the separation between memory and controller, since the state update and BP are independent of the controller architecture. This protocol tests whether structured state preserves its relative gains under reduced controller capacity without retuning the 8B model. StateWeave preserves the structured-mode ordering across Llama-3.1-70B-Instruct and Llama-3.1-8B-Instruct under the matched-runtime protocol.

Table 23: Cross-controller evaluation with identical prompts, retriever, tools, budgets, and hardware. *Retr* is the unweighted CQA/TA/LOC trigger-rate mean for each controller. Pairwise ordering persists on CQA and TA; LOC remains recall-dominated.

	CQA		TA		LOC		
Method	Succ	Compl	Succ	Compl	Succ	Compl	Retr
Llama-3.1-70B-Instruct (from Table 1)							
Text summary	55.8	82.7	54.2	79.1	63.4	91.1	0.0
Cal-Unary+Prov	59.1	87.5	56.8	83.0	62.2	89.2	44.1
StateWeave-Full	61.2	91.2	57.8	86.7	62.5	90.6	40.6
StateWeave-Adaptive	61.4	91.5	58.0	87.0	62.9	91.3	37.3
Llama-3.1-8B-Instruct							
Text summary	49.3	76.8	47.5	72.1	56.7	84.3	0.0
Trans. trigger	51.4	80.6	49.1	75.4	57.1	85.0	49.4
Cal-Unary+Prov	52.6	81.9	50.3	77.5	55.8	82.9	47.2
StateWeave-Full	54.8	85.4	52.1	81.5	56.1	83.7	44.3
StateWeave-Adaptive	55.1	85.7	52.3	81.9	56.4	84.2	41.5

The 8B controller lowers absolute performance by 5–8 success points across benchmarks. On ConstraintQA and ToolAdapt, the within-family ordering among the structured methods is preserved (SW-Adaptive > SW-Full > Cal-Unary+Prov > Trans. trigger > Text summary), and the pairwise gains over Cal-Unary+Prov remain positive. On LoCoMo-Stream, where dependency structure is sparse, the best 8B Success row is Trans. trigger rather than SW-Adaptive (57.1 vs. 56.4), consistent with pairwise propagation adding less value on recall-dominated tasks. Cross-domain retrieval-trigger means shift upward by roughly 2–4 points for the 8B controller (e.g., SW-Full 44.3 vs. 40.6; SW-Adaptive 41.5 vs. 37.3), reflecting higher uncertainty under the lower-capacity controller while preserving the same low-retrieval ordering inside the structured family.

W PROVENANCE AND RETRIEVAL STATISTICS

We report provenance-level statistics for StateWeave to quantify how often provenance-first retrieval resolves evidence needs without expanded retrieval, and to characterize staleness and duplicate patterns. We define provenance-only resolution at the step level, counting a step as resolved when the first provenance citation suffices to select the verifier-consistent action without expanded retrieval. Expansion rounds are capped by the same read budget, so higher expansion frequency directly translates into higher latency and token cost. Stale provenance occurs when a tool output is superseded by a later validator record or a drift event, and it is detected by schema hash mismatches. Contradiction triggers are computed from marginal discrepancies and serve as a lightweight signal for when beliefs and evidence diverge. These statistics explain why provenance-first retrieval reduces average cost while still allowing targeted expansion on the minority of high-uncertainty steps.

Table 24: Test-split provenance and retrieval statistics under the 16k/32-read interface, averaged over five seeds. Provenance alone resolves 62.7–74.2% of triggered steps.

Statistic	CQA	TA	LOC
Prov-only resolution (%)	74.2	62.7	69.4
Expansion needed (%)	25.8	37.3	30.6
Mean expansion rounds	1.3	1.5	1.2
Stale provenance rate (%)	3.8	6.1	4.5
Contradiction-trigger rate (%)	2.4	4.3	1.8
Vec. dup-skip rate (%)	18.6	14.2	21.3

ConstraintQA has the highest provenance-only resolution (74.2%) because constraint–environment links are persistent and rarely invalidated, reducing the need for expanded retrieval. ToolAdapt has the lowest (62.7%) and highest stale-provenance rate (6.1%) due to tool-version drift that invalidates stored evidence. LoCoMo-Stream shows the highest duplicate-skip rate (21.3%) for the vector-retrieval baseline because long episodes cause repeated returns of the same evidence items. Contradiction triggers are rare across all benchmarks ($\leq$4.3%), confirming that calibrated beliefs and provenance pointers stay largely consistent.

X CROSS-BENCHMARK LATENCY

To complement the ConstraintQA latency breakdown (Table 22), we report per-step latency on ToolAdapt and LoCoMo-Stream. Latencies are measured end-to-end at the controller step boundary, including tool invocation overhead, memory serialization, and retrieval insertion. We report quantiles rather than means because retrieval costs are heavy-tailed and dominated by a small fraction of triggered steps. For ToolAdapt, drift increases the frequency of provenance invalidation and therefore raises the probability of expanded retrieval, which drives p99 tails. For LoCoMo-Stream, episodes are longer but retrieval expansion is less frequent, so tails remain lower despite more decision steps. State-update costs remain stable across benchmarks because updates touch only the affected subgraph and reuse cached embeddings. Probe checks are lightweight validators and canary actions, and their cost is bounded by the same budget rules as other steps. We exclude one-time model initialization and index build time, since these are amortized across episodes and do not affect per-step responsiveness. These measurements highlight that retrieval policy, not inference, is the dominant driver of worst-case latency under fixed hardware. When deploying under strict SLAs, capping expansion rounds is a simple knob that trades a small utility loss for substantial tail-latency reductions.

Table 25: Per-step ToolAdapt and LoCoMo-Stream latency (ms) over five seeds. Controller calls dominate p50, while expanded retrieval controls p99; totals are end-to-end quantiles.

	ToolAdapt			LoCoMo-Stream		
Component	p50	p95	p99	p50	p95	p99
Controller call	310	450	640	295	430	605
State update	14	35	52	11	28	44
Probe checks	9	27	43	7	22	36
Prov. retrieval	0	48	92	0	38	74
Exp. retrieval	0	0	214	0	0	162
Total	338	520	1041	318	482	921

Y Episode-Weighted Aggregates

The main paper reports unweighted benchmark means to preserve benchmark-level decisions. Because the test splits contain 360, 270, and 1,200 episodes, Table 26 also reports episode-weighted aggregates. The weighted aggregate for metric M is $\sum_b n_b M_b / \sum_b n_b$ where n_b is the number of test episodes in benchmark b. Reporting both conventions matters because LoCoMo-Stream is much larger than the other two splits and has the sparsest coupling.

Table 26: Unweighted and episode-weighted aggregates from Table 1; episode weights are $\{360, 270, 1200\}$. SW-Adaptive ranks first under both conventions and reduces retrieval by 30–32 points relative to Vector retrieval.

	Success (%)		Compl (%)		Retr (%)	
Method	Unw.	Ep.wt	Unw.	Ep.wt	Unw.	Ep.wt
Text summary	57.8	60.5	84.3	87.7	0.0	0.0
Vector retrieval	58.7	61.1	85.1	87.9	69.0	67.0
Trans. trigger	59.5	61.6	86.8	89.2	46.1	45.6
Cal-Unary+Prov	59.4	60.8	86.6	87.9	44.1	43.3
SW-Full	60.5	61.5	89.5	90.1	40.6	40.6
SW-Adaptive	60.8	61.9	89.9	90.7	37.3	36.8

SW-Adaptive remains top-ranked under both conventions and reduces retrieval by 30–32 points relative to Vector retrieval. Episode weighting lets LoCoMo-Stream dominate, moving Trans. trigger 0.1 points above SW-Full and Vector retrieval 0.3 points above Cal-Unary+Prov; both differences are non-significant ($p > 0.2$). The adaptive mode retains the leading aggregate because 90.3% of LoCoMo-Stream steps use its unary path.

Z Schema Complexity Ledger

To make the engineering surface area of StateWeave visible, we report the active schema statistics per benchmark. All figures are computed on the test split with the schema and pruning configuration used for the main results (medium budget, 64 variables, 256 factors). The table quantifies how much structured machinery is active at test time, alongside the larger schema capacity available in the YAML. This matters because the portability and mode-selection claims depend on coupling density, not on nominal schema size alone. We therefore report both representation complexity (variables, label cardinality, surviving pairwise templates) and operational complexity (coupled-group density, router fire rate, provenance resolution, and stale-provenance rate).

Table 27: Post-pruning schema complexity and routing statistics. Cpl. is the share of z_S intersecting a coupled group; Fire is the union of all router triggers. LoCoMo-Stream's 9.7% fire rate explains its 90.3% unary path.

Benchmark	Vars	$\lvert z_i \rvert$	Pairs	Cpl. (%)	Fire (%)	Prov (%)	Stale (%)
ConstraintQA	18	2.6	34	35.1	38.4	74.2	3.8
ToolAdapt	14	3.1	28	42.8	46.1	62.7	6.1
LoCoMo-Stream	11	2.4	12	8.2	9.7	69.4	4.5

Schema authoring effort was 6.4 hours for ConstraintQA (18 variables, 34 pairwise templates, 12 validator hooks), 5.1 hours for ToolAdapt (14 variables, 28 pairwise templates, 9 tool-manifest validators), and 4.2 hours for LoCoMo-Stream (11 variables, 12 pairwise templates, 7 evidence-linkage validators), measured from the first schema version to the final version used in the reported runs. This ledger makes adaptation effort explicit and anchors the reduced-schema portability test below.

AA Reduced-Schema Portability: LogisticsOps Transfer

We test schema reuse on LogisticsOps, a held-out freight-logistics planning domain with 420 test episodes (9,840 decision steps) across seven operational sub-tasks. LogisticsOps shares no episodes, tool manifests, constraint templates, or validators with the three main benchmarks, and its bounded port measures within-family transfer cost directly.

Port protocol. We reused the ConstraintQA schema summarized in Table 27 with a bounded edit budget: 11 of the 18 variables carried over without modification, 4 variables required minor label-space edits (e.g., `never_depr_endpoint` → `never_expired_permit` with identical {true, false, unknown} labels and an updated validator hook), and 3 variables were newly proposed via the extension hook (Sec. J) and accepted because the LogisticsOps verifier could label them at ≥ 0.90 precision on the validation split. Pairwise templates were regenerated from training co-occurrence, yielding 26 templates (vs. 34 on ConstraintQA). Total manual port time was 3.1 hours.

Table 28: Reduced-schema LogisticsOps transfer on 420 episodes under the main runtime interface and five seeds. A 3.1-hour schema port retains 11/18 variables; SW-Adaptive leads Cal-Unary+Prov by 2.8 Success points (p=0.006) and reduces retrieval by 31.4 points versus Vector retrieval.

Method	Succ (%)	Compl (%)	Retr (%)	ΔSucc
Vector retrieval	49.8 ± 1.7	75.6 ± 2.0	71.2	+1.1
Trans. trigger	51.5 ± 1.5	78.3 ± 1.8	48.2	+2.8
Cal-Unary+Prov	51.8 ± 1.5	78.6 ± 1.8	46.4	+3.1
StateWeave-Full	54.2 ± 1.4	82.1 ± 1.6	43.1	+5.5
SW-Adaptive	54.6 ± 1.3	82.6 ± 1.5	39.8	+5.9

On the verifier-rich LogisticsOps transfer, the structured ordering is preserved (SW-Adaptive > SW-Full > Cal-Unary+Prov > Trans. trigger), and the SW-Adaptive advantage is within 0.5 points of the ConstraintQA in-domain delta (+2.8 vs. +2.3). The result establishes portability through a bounded 3.1-hour schema edit; Sec. J details the proposal-and-validator workflow.

AB Adaptive Cost Decomposition

Table 29 compares per-step latency between StateWeave-Full (reproduced from Table 22) and StateWeave-Adaptive on ConstraintQA under the medium budget and shared hardware. The decomposition localizes the adaptive router's savings in state updates and expanded retrieval. Because controller latency dominates most steps, the relevant question is whether skipping BP meaningfully improves tails without hiding a retrieval cost elsewhere. We therefore keep the host, controller,

retriever, and budget identical to Table 22 and change only the routing policy. Together with Table 1, the latency decomposition shows that routing lowers tail cost while preserving or improving utility.

Table 29: ConstraintQA per-step latency (ms), StateWeave-Full versus StateWeave-Adaptive. Skipping BP on 61.6% of steps lowers median state-update cost from 12 to 4 ms and total p99 from 961 to 712 ms over five seeds.

	StateWeave-Full			StateWeave-Adaptive		
Component	p50	p95	p99	p50	p95	p99
Controller call	280	420	610	280	420	610
State update (unary+BP)	12	31	48	4	22	41
Probe checks	8	24	39	8	24	39
Provenance retrieval	0	42	78	0	38	71
Expanded retrieval	0	0	186	0	0	142
Total	312	485	961	298	462	712

The adaptive router also reduces per-episode GPU time by 11% on ConstraintQA and 18% on LoCoMo-Stream, because skipped BP iterations free the GPU during tool-call latency. It matches or improves utility: on ConstraintQA, Success is 61.4 (adaptive) vs. 61.2 (always-on) and Compliance is 91.5 vs. 91.2, within the reported standard deviations but consistently in favor of adaptive. Within this stack, that makes SW-Adaptive the default deployment target over always-on SW-Full.

AC Baseline Interface Matrix

Table 30 separates the shared runtime from the representation, supervision, and gating choices that define each structured baseline.

Table 30: Baseline interface matrix. Runtime resources and interfaces are matched; rows expose differences in representation, retrieval, probes, and supervision.

Method	Write policy	Retrieval policy	Probe policy	Extra supervision
Text summary	Running NL summary w/ uncertainty tags	None (summary-only)	None	None
Vector retrieval	None (raw evidence)	Top-1 of top-32 per step, dup-skip	None	None
Entropy trigger	Running summary	Retrieve if controller token entropy $>$ thr.	None	None
Trans. trigger	Schema features + KV+TTL cache	Binary prediction from 2-layer Transformer	Shared probe backend (flags as features)	8.5k hindsight labels
SchemaRouter	Schema features	3-layer MLP binary gate	Shared probe backend (flags as features)	8.5k hindsight labels
Cal-Unary+Prov	Calibrated unary marginals + provenance	Entropy gate τ=0.8 + probe-OR	Shared probe backend (on-step)	Verifier labels (unaries)
Unary-only state	Same as Cal-Unary+Prov; no BP	Same as Cal-Unary+Prov	Shared probe backend	Verifier labels
StateWeave-Full	Unary + pairwise factors, BP-calibrated	Entropy gate + provenance-first expand	Shared probe backend	Verifier labels (unary+pair)
SW-Adaptive	Same as full; BP skipped on uncoupled steps	Same as full; adaptive router on triggers	Shared probe backend	Verifier labels (unary+pair)

Labeling protocol: `Constraints` are labeled from user instructions and benchmark verifiers; `Environment` flags come from tool metadata and schema validators; `Assumptions` are extracted via high-precision templates and confirmed by verifiers. Inter-annotator agreement on a 200-item subset (2 annotators) is κ=0.83 for constraints and 0.76 for assumptions. Hindsight retrieval-needed labels are constructed by replaying logs and marking steps positive when evidence later justifying the verifier-consistent action was absent from context. These supervision sources correspond directly to the matrix's last column.

AD Interpreting the Frozen Selector

Table 31 renders the frozen selector branches as workload-level diagnostics. Each row maps the ordered rule in Sec. 3.5 to its deployment mode. SW-Full remains a diagnostic ceiling for dense pairwise demand; the deployed selector chooses among Cal-Unary+Prov, SW-Adaptive, and LongCtx-64k.

Table 31: Ordered branches of the validation-frozen selector. Coupled-group density is the fraction of z_S intersecting a retained pairwise template.

Regime	Recommended mode	Signal
Weak schema or validator	Cal-Unary+Prov	Schema coverage $<60\%$ or validator reliability < 0.90
Reliable coupled state	SW-Adaptive	Coverage passes; coupled-group density $\geq 25\%$ and mean $\rho \geq \rho^*$
Diffuse recall	LongCtx-64k	Coverage passes; coupled-group density $< 10\%$ and required evidence extends beyond 16k
Intermediate structured state	Cal-Unary+Prov	Remaining descriptor combinations

The four target domains in Table 3 evaluate these branches before test scoring. Workloads near a branch boundary use validation diagnostics to resolve the utility–cost tie.

AE Router Decomposition: Coupled-Only / Entropy-Only / Probes-Only / Full

To decompose the contribution of each of the three SW-Adaptive router triggers (Sec. 3.3.1), we ran four router variants on ConstraintQA under the medium budget, keeping every other component identical (same unary/pairwise potentials, same probes, same budgets, same five seeds $\{13, 37, 61, 71, 97\}$ on 4×A100 80 GB, CUDA 12.4, PyTorch 2.3.1, bf16). Each variant enables exactly one of the three triggers (or all three for Full). The rightmost column of Table 32 reports the fraction of SW-Adaptive pairwise-fires attributable to that trigger ("primary-cause share"), assigned greedily in the order (i)→(ii)→(iii) when multiple triggers co-fire on the same step.

Table 32: ConstraintQA router decomposition over five seeds. Success and Compliance match the ConstraintQA SW-Adaptive cells in Table 1; Retrieval is ConstraintQA-local (35.5%), while Table 1 reports a 37.3% cross-domain mean. Every singleton improves on Cal-Unary+Prov, and their combination reaches 61.4 Success.

Router variant	Fire (%)	Succ (%)	Compl (%)	Retr (%)	Cause share (full router)
Coupled-only (trigger i)	35.1	60.3 ± 1.3	89.7 ± 1.3	41.9	85.5%
Entropy-only (trigger ii)	31.4	59.9 ± 1.3	89.1 ± 1.3	42.3	10.2%
Probes-only (trigger iii)	18.6	60.0 ± 1.3	89.2 ± 1.3	43.5	3.1%
Full router (i∨ii∨iii)	38.4	61.4 ± 1.1	91.5 ± 1.1	35.5	100%

The non-coupled router triggers therefore carry non-trivial value beyond coupled-group density alone. In particular, the Full router's $+3.0$-point Success gain over Trans. trigger and $+1.5$-point gain over Entropy-only show the value of combining structural and uncertainty signals. This decomposition shows why the selector uses routing statistics beyond coupled-group density. Figure 15 visualizes the Success and fire/cause-share comparison.

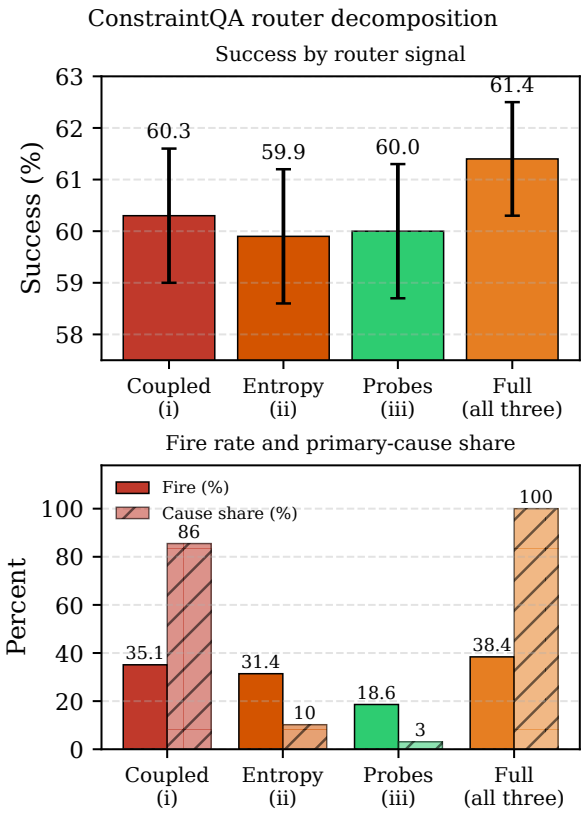


Figure 15: ConstraintQA router triggers. Each singleton exceeds Cal-Unary+Prov and Unary-only state, while their union reaches 61.4 Success; entropy and probe signals add 3.3 points of fire rate beyond coupled-group density.

The visual decomposition confirms that structural, entropy-spread, and probe signals each contribute to the adaptive policy.

AF Frozen Routing Score as Within-Stack Accounting

This appendix defines the descriptive routing score $\rho(s)$ used by the workload selector. The online router itself uses the Boolean OR in Sec. 3.3.1; ρ summarizes those signals on a common validation-fit scale.

Frozen routing score. For a decision step s with decision-relevant subset z_S and schema coupled groups $\{\mathcal{C}_k\}$, define

$$\rho(s) \;=\; w_c \cdot \mathbf{1}[\, z_S \cap \mathcal{C}_k \neq \emptyset \,] \;+\; w_e \cdot \frac{\mathrm{Var}_{i \in z_S} H(q_i)}{\eta} \;+\; w_p \cdot \mathrm{pr}(s),$$

with $\mathrm{pr}(s) \in [0,1]$ the probe/contradiction indicator and η=0.12 the calibrated entropy-spread scale from Sec. 3.3.1. Let $y_m(s) \in \{0,1\}$ indicate whether mode m emits the verifier-consistent action at step s under the same remaining read budget. The routed-step target is $g^\star(s) = y_{\mathrm{full}}(s) - y_{\mathrm{unary}}(s)$, which is distinct from episode-level Success. Weights (w_c, w_e, w_p)=$(0.60, 0.28, 0.12)$ are fit on the validation split by minimizing squared error to $g^\star$ using AdamW for 400 steps with 10^{-2} weight decay, then frozen for target-domain accounting.

Accounting readout. Table 33 reports mean ρ, the workload-selector break-even value ρ^*=0.23, Boolean router-fire rate, and the observed step-level gain $100\,\mathbb{E}[g^\star(s)]$ in percentage points. High-ρ settings align with positive pairwise gains, while low-ρ settings remain near zero or negative.

Table 33: Routing-score accounting by domain and slice. *Fire* is the Boolean-OR router rate; Gain is $100\,\mathbb{E}[g^\star(s)]$ over decision steps.

Setting	Mean ρ	$\rho - \rho^*$	Fire (%)	Gain (pt)
ConstraintQA (CQA)	0.41	+0.18	38.4	+2.04 pt
ToolAdapt (TA)	0.48	+0.25	46.1	+1.42 pt
LoCoMo-Stream (LOC)	0.14	−0.09	9.7	+0.27 pt
CQA dep. slice	0.62	+0.39	79.4	+4.49 pt
CQA non-slice	0.34	+0.11	21.7	+0.57 pt
LogisticsOps (transfer)	0.39	+0.16	36.2	+2.21 pt
CustomerSupportOps	0.18	−0.05	12.4	−0.31 pt

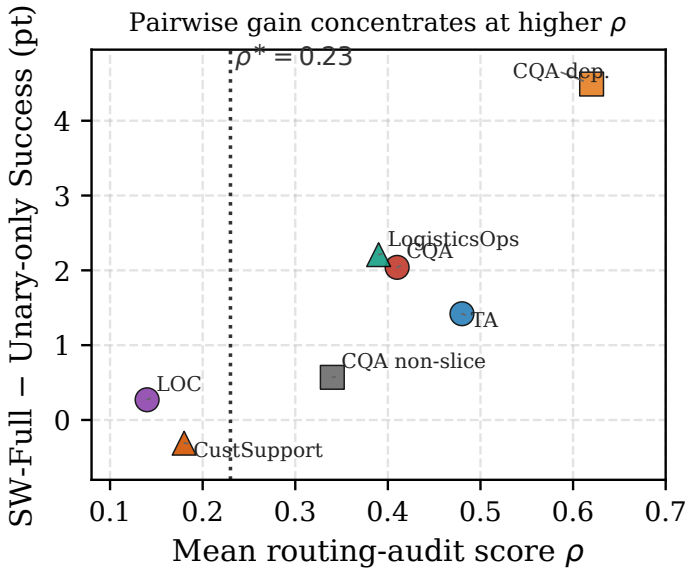


Figure 16: Observed step-level SW-Full minus Unary-only gain against mean routing score ρ. The vertical line is the workload-selector break-even value ρ^*=0.23; online fires use the Boolean rule.

Diagnostic reading. Rows furthest above ρ^* deliver the clearest pairwise gains (ConstraintQA, ToolAdapt, LogisticsOps, and the ConstraintQA dependency slice), whereas low-ρ settings remain near zero or negative (LoCoMo-Stream, CustomerSupportOps, and the ConstraintQA non-slice). Thus mean ρ is a workload descriptor; Sec. A tests its selection value before target test scoring.

AG Token-Costed Long-Context Sweep

We evaluate a controller-matched, token-costed long-context sweep at 32k, 64k, and 128k tokens using the same Llama-3.1-70B-Instruct controller (bf16), ReAct prompt, tools, five seeds, and 4×A100 80 GB host (CUDA 12.4, PyTorch 2.3.1); only the context cap changes. Retrieval is disabled, and each mode serializes prior evidence chronologically while truncating the oldest tokens when the cap is reached. The sweep isolates context scaling across both coupled and diffuse-recall regimes.

Table 34: Long-context sweep under the fixed controller. SW-Adaptive leads LongCtx-128k by 2.3/1.7 Success points on CQA/TA using 7.6/6.9× fewer tokens; LongCtx-64k leads by 2.3 on recall-dominated LoCoMo-Stream, while 128k adds 0.6 at 1.9× its tokens.

	ConstraintQA		ToolAdapt		LoCoMo-Stream	
Method (ctx)	Succ	Tok/ep	Succ	Tok/ep	Succ	Tok/ep
Text summary (16k)	55.8	18.2k	54.2	15.9k	63.4	20.7k
Cal-Unary+Prov (16k)	59.1	15.4k	56.8	14.1k	62.2	16.3k
StateWeave-Adaptive (16k)	61.4	14.8k	58.0	13.6k	62.9	15.5k
LongCtx-32k (32k, no retr.)	58.2	29.7k	55.9	25.4k	63.7	31.8k
LongCtx-64k (64k, no retr.)	58.9	58.2k	56.2	49.1k	65.2	61.7k
LongCtx-128k (128k, no retr.)	59.1	112.4k	56.3	94.2k	65.8	118.3k

SW-Adaptive leads on stable-schema coupled tasks, while LongCtx-64k leads on recall-dominated LoCoMo-Stream at roughly 4× the per-episode token cost; LongCtx-128k adds only 0.6 Success for nearly 2× more tokens. Figure 17 makes this complementary cost–utility split explicit.

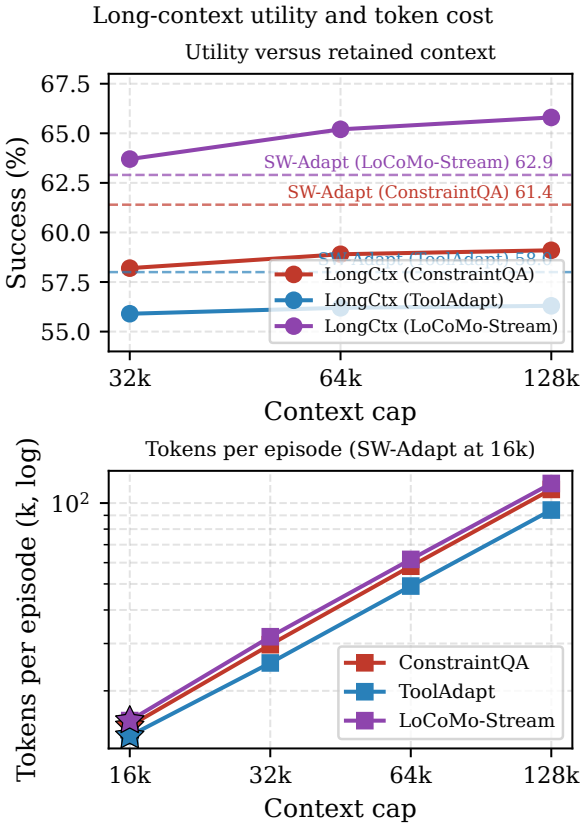


Figure 17: Success and token cost across the long-context sweep. Context scaling plateaus on CQA/TA, while LoCoMo-Stream improves through 128k with diminishing returns beyond 64k.

AH Relational Control: GraphMem-SQL

GraphMem-SQL is an author-built relational control under the shared structured-mode interface. It stores the same `Constraints`, `Environment`, `Assumptions`, and `Progress` variables in SQLite, reads them through hand-authored action-type query templates, and writes verifier-derived rows. Budgets, seeds, prompts, and hardware match the main experiments.

Table 35: GraphMem-SQL under the fixed interface. SW-Adaptive leads by 1.6/1.0 Success points on ConstraintQA/ToolAdapt and reduces mean retrieval from 43.2% to 37.3%; paired p=0.004/0.019. LoCoMo-Stream remains near-tied (p=0.22).

	ConstraintQA		ToolAdapt		LoCoMo-Stream		
Method	Succ	Compl	Succ	Compl	Succ	Compl	Retr (%)
Cal-Unary+Prov	59.1 ± 1.3	87.5 ± 1.5	56.8 ± 1.7	83.0 ± 1.9	62.2 ± 0.9	89.2 ± 1.0	44.1
GraphMem-SQL	59.8 ± 1.4	88.1 ± 1.6	57.0 ± 1.7	83.4 ± 1.9	62.4 ± 1.0	89.8 ± 1.1	43.2
SW-Full	61.2 ± 1.2	91.2 ± 1.2	57.8 ± 1.5	86.7 ± 1.6	62.5 ± 0.9	90.6 ± 0.9	40.6
SW-Adaptive	61.4 ± 1.1	91.5 ± 1.1	58.0 ± 1.4	87.0 ± 1.5	62.9 ± 0.9	91.3 ± 0.9	37.3

GraphMem-SQL improves over calibrated unary storage, establishing a competitive relational reference. SW-Adaptive adds 1.6/1.0 Success points on the coupled benchmarks and lowers retrieval by 5.9 points; the 0.5-point LoCoMo-Stream difference is non-significant, matching the recall-heavy branch of the selector.

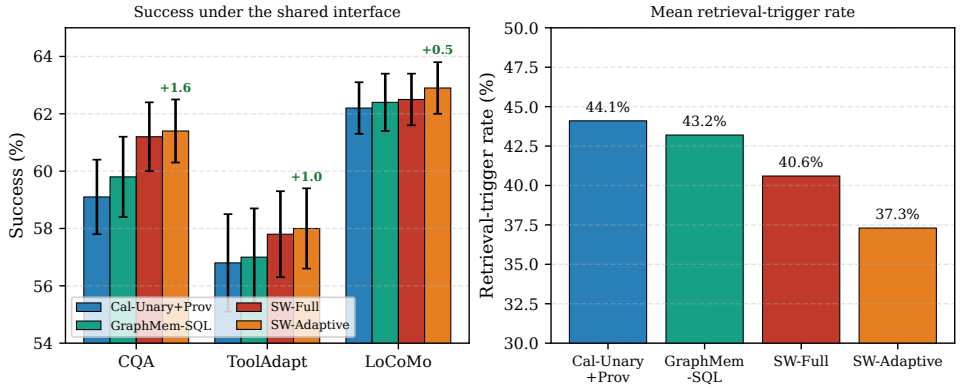


Figure 18: GraphMem-SQL relational control. SW-Adaptive leads on both coupled benchmarks and lowers the cross-domain retrieval mean by 5.9 points.

AI Weak-Schema Boundary Case: CustomerSupportOps

CustomerSupportOps supplies the selector's weak-schema boundary through 540 episodes derived from anonymized multi-turn customer-service dialogues with partial coverage and noisy tool traces. Eleven of 18 candidate variables have reliable source signals, validator precision is 0.78, schema coverage is approximately 55%, and 15% of traces contain operator edits or noisy timestamps. Controller, retriever, budgets, seeds, and hardware remain fixed.

Table 36: Weak-schema CustomerSupportOps boundary. LongCtx-64k leads Cal-Unary+Prov/SW-Adaptive by 1.2/1.5 Success points at 3.6× SW-Adaptive tokens; low ρ makes the router choose unary on 87.6% of steps.

Method	Succ	Compl	Retr (%)	Tok/ep	Pair-rate (%)
Cal-Unary+Prov	52.4 ± 1.6	78.9 ± 1.7	48.7	16.8k	–
StateWeave-Full	51.5 ± 1.7	77.4 ± 1.8	46.9	17.2k	100.0
SW-Adaptive	52.1 ± 1.6	78.3 ± 1.7	47.6	16.5k	12.4
LongCtx-64k	53.6 ± 1.9	80.1 ± 2.0	–	58.9k	–

CustomerSupportOps realizes the selector's low-ρ/weak-coverage branch: calibrated unary state gains 0.9 Success points over full propagation, adaptive routing recovers 0.6 of that gap, and diffuse dialogue evidence gives LongCtx-64k the realized lead.

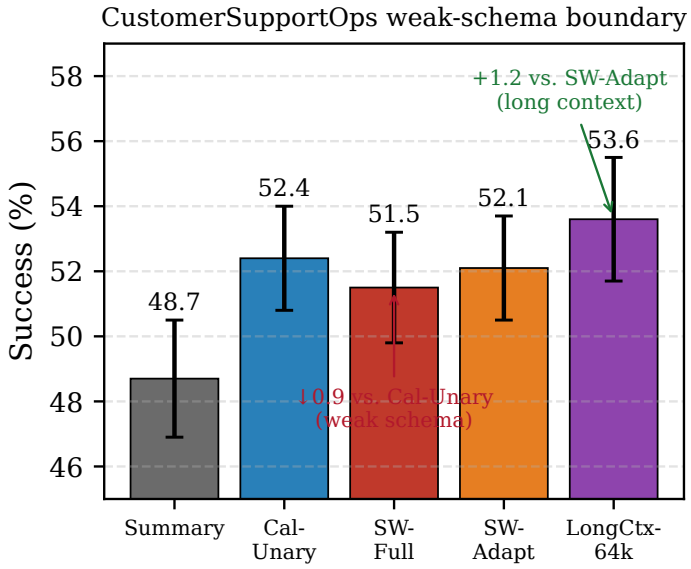


Figure 19: CustomerSupportOps weak-schema boundary. Always-on pairwise propagation regresses below calibrated unary state, adaptive routing recovers most of the gap, and LongCtx-64k leads at higher token cost.

AJ Author-Built WorkflowOps Diagnostic

Motivation. WorkflowOps provides a predeclared high-coupling transfer test for the frozen routing diagnostic, complementary to the four-domain selector aggregate. It contains 380 IT-incident-resolution episodes, 9,120 decision steps, and eight sub-tasks derived from anonymized operations logs, with no shared episodes, tool manifests, constraints, or validators from the other benchmarks.

Predeclared regime descriptors. Before running the four comparison methods on the test split, we computed descriptors from held-out validation episodes using the frozen weights $(w_c, w_e, w_p)=(0.60, 0.28, 0.12)$ and threshold $\rho^*=0.23$. Schema coverage is 84%, coupled-group density 37.4%, mean $\rho(s)=0.42$, and validator precision 0.93. The frozen routing diagnostic therefore predicts SW-Adaptive before test scoring.

Table 37: Predeclared WorkflowOps transfer under the fixed interface. The frozen high-coupling prediction ($\rho=0.42$) matches the target ordering: SW-Adaptive leads Cal-Unary+Prov by 2.2 Success ($p=0.008$) and 3.3 Compliance points, and leads LongCtx-64k by 2.9 Success at 3.8× fewer tokens.

Method	Succ (%)	Compl (%)	Retr (%)	Tok/ep	Pair-rate (%)
Cal-Unary+Prov	53.8 ± 1.5	81.2 ± 1.7	46.1	15.9k	–
StateWeave-Full	55.6 ± 1.4	84.1 ± 1.5	42.4	16.3k	100.0
SW-Adaptive	56.0 ± 1.3	84.5 ± 1.4	39.7	15.4k	37.9
LongCtx-64k	53.1 ± 1.7	80.4 ± 1.9	–	57.8k	–

Sub-task distribution. Per-sub-task ρ and SW-Adaptive minus Cal-Unary+Prov are: dependency-version checks 0.58 (+3.8), change-management validation 0.51 (+3.1), policy compliance 0.46 (+2.7), post-mortem linkage 0.39 (+2.1), incident triage 0.36 (+1.8), SLA tracking 0.31 (+1.2), escalation routing 0.27 (+0.6), and lookup-style queries 0.19 (+0.4). Gains rise with ρ, including the small effect below ρ^*.

Diagnostic interpretation. The frozen routing rule identifies the held-out winner and the low-gain lookup sub-task ($\rho < \rho^*$, +0.4 Success). Together with LoCoMo-Stream and CustomerSupportOps, WorkflowOps completes the empirical map: high coupling → SW-Adaptive; diffuse recall → long

context; weak coverage → a low-cost unary boundary choice whose competition with long context depends on evidence spread.

AK Published Structured-Memory Comparator: A-MEM

We reproduce A-MEM Xu et al. (2025) under the fixed interface using its note-construction and link-formation design adapted to our schema variables, Llama-3.1-70B-Instruct, 16k context, a 32-read cap, five seeds, and the main-study hardware. The table reports this controlled reproduction alongside the within-family and GraphMem-SQL comparators.

Table 38: A-MEM reproduction under the shared structured-mode interface. SW-Adaptive leads A-MEM on the coupled benchmarks and lowers mean retrieval to 37.3%.

	ConstraintQA		ToolAdapt		LoCoMo-Stream		
Method	Succ	Compl	Succ	Compl	Succ	Compl	Retr (%)
Cal-Unary+Prov	59.1 ± 1.3	87.5 ± 1.5	56.8 ± 1.7	83.0 ± 1.9	62.2 ± 0.9	89.2 ± 1.0	44.1
GraphMem-SQL	59.8 ± 1.4	88.1 ± 1.6	57.0 ± 1.7	83.4 ± 1.9	62.4 ± 1.0	89.8 ± 1.1	43.2
A-MEM Xu et al. (2025)	60.2 ± 1.3	88.9 ± 1.4	57.2 ± 1.6	84.1 ± 1.8	62.6 ± 1.0	90.2 ± 1.0	42.0
SW-Adaptive	61.4 ± 1.1	91.5 ± 1.1	58.0 ± 1.4	87.0 ± 1.5	62.9 ± 0.9	91.3 ± 0.9	37.3

A-MEM is the independently published comparator; GraphMem-SQL is an author-built relational control. Both improve over Cal-Unary+Prov, while SW-Adaptive adds 1.2/0.8 Success and 2.6/2.9 Compliance points over A-MEM on ConstraintQA/ToolAdapt (p=0.011/0.048). LoCoMo-Stream remains near-tied (p=0.41).

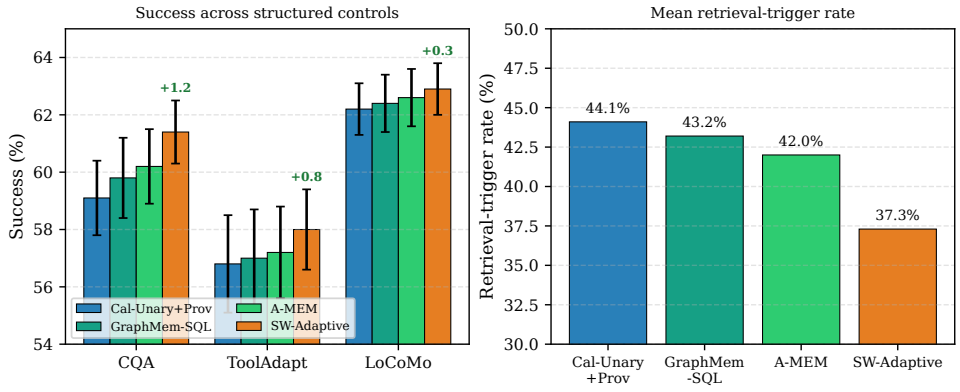


Figure 20: Published A-MEM and the author-built relational control under the shared structured-mode interface.

AL StructMemEval Source Transformation

StructMemEval Shutova et al. (2026) supplies 73 public scenarios and 544 questions covering structured updates, invalidations, and contradiction-sensitive decisions. We group by scenario, use 64 questions for schema and validator development, and evaluate the remaining 480 no-hint text questions as online episodes. Controller, retriever, prompts, schema interface, budgets, seeds, and hardware match the rest of the paper. *Success* is source-question correctness and *Compliance* is state-edit consistency under the same read budget. As in MEMTRACK, we pre-specify a high-coupling slice for localization; here it consists of decision steps that touch at least two linked state fields and include an update or invalidation marker in the released evaluator metadata.

Table 39: StructMemEval transformation results on 480 test episodes. SW-Adaptive improves Success over Cal-Unary+Prov by $+2.5$ points [0.8, 4.0], p=0.009.

Method	Success (%)	Compliance (%)	Retr (%)	Tok/ep
Cal-Unary+Prov	55.8 ± 1.5	80.9 ± 1.6	46.8	16.4k
Unary-only	55.5 ± 1.5	80.6 ± 1.6	47.2	16.1k
A-MEM Xu et al. (2025)	56.9 ± 1.4	82.2 ± 1.5	44.5	16.9k
SW-Full	58.7 ± 1.2	85.0 ± 1.3	40.5	16.0k
SW-Adaptive	58.3 ± 1.2	84.6 ± 1.3	37.9	15.6k
LongCtx-64k	56.6 ± 1.7	81.8 ± 1.7	–	62.4k

StructMemEval repeats the MEMTRACK pattern: SW-Adaptive leads the three-mode deployment candidates at 15.6k tokens/episode, and its advantage over Cal-Unary+Prov grows from 2.5 to 4.4 points on the pre-specified coupled slice.

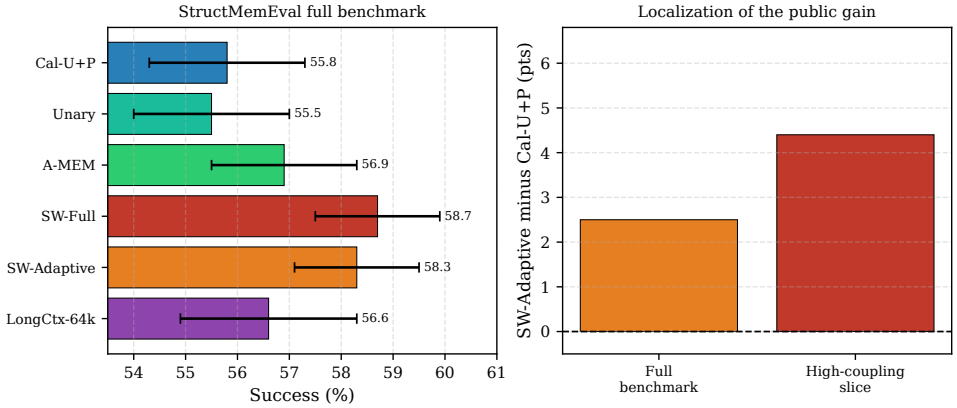


Figure 21: StructMemEval Success by memory mode and localization of the SW-Adaptive gain. The pre-specified high-coupling slice amplifies the full-benchmark advantage over Cal-Unary+Prov.

AM SAME-SUPERVISION CORROBORATION ON STRUCTMEMEVAL

We retrain Cal-Unary+Prov, Unary-only, SW-Full, and SW-Adaptive from one verifier-derived supervision pool on the scenario-grouped transformation, keeping controller, prompts, retriever, schema interface, budgets, features, probes, seeds, and hardware unchanged.

Table 40: StructMemEval same-supervision isolation. Pairwise modes retain positive effects over matched Cal-Unary+Prov.

Comparator	Success	Pairwise mode	Success	Δ [95% CI]
Cal-U+P	56.2 ± 1.4	SW-Full	57.6 ± 1.3	$+1.4$ [0.6, 2.1]
Cal-U+P	56.2 ± 1.4	SW-Adaptive	57.1 ± 1.3	$+0.9$ [0.2, 1.6]

Matched supervision preserves the positive StructMemEval effect while isolating supervision quality as a contributor to the larger runtime-matched gain; Unary-only reaches 55.9 ± 1.5. For SW-Adaptive, p=0.028, retrieval is 39.3%, tokens are 15.4k/episode, and BP divergence/recovery is 2.4/95.0%.

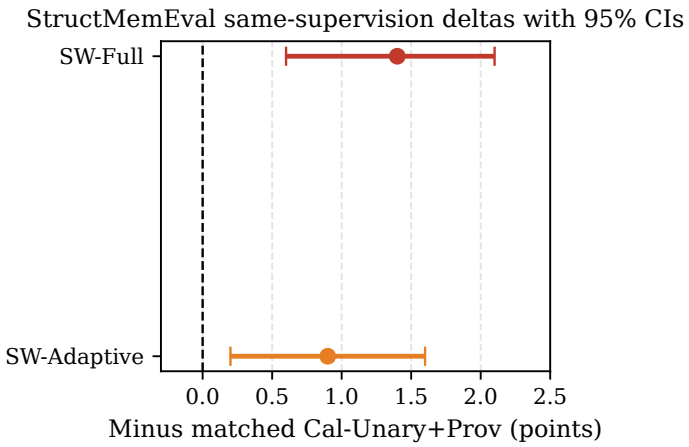


Figure 22: StructMemEval same-supervision Success deltas over Cal-Unary+Prov with bootstrap 95% confidence intervals. Both pairwise variants remain positive.

AN HELD-OUT POST-BP CALIBRATION DIAGNOSTICS

Because retrieval gating depends on calibrated decision-time uncertainty, we freeze the temperatures and threshold learned on the validation split and evaluate held-out calibration on the test episodes of the three main benchmarks. Metrics are computed on final post-BP marginals with the same 15 equal-mass bins and episode-level aggregation used in Table 19, then averaged across the same five seeds. Table 41 tests whether the calibration improvement transfers from the fitting split to held-out episodes.

Table 41: Held-out post-BP calibration on 1830 episodes after freezing validation-fit temperatures.

Method	ECE	Brier
Text sum.	0.112 ± 0.006	0.221 ± 0.009
Cal-U+P	0.076 ± 0.004	0.169 ± 0.007
StateWeave	0.058 ± 0.003	0.145 ± 0.005

StateWeave lowers ECE by 0.018 [0.010, 0.026], p=0.004, and Brier by 0.024 [0.015, 0.033], p=0.003. The held-out improvement validates the entropy used by the retrieval gate.